INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2021IPPAT035

Thèse de doctorat

TELECOM Paris

IP PARIS

# Expected smoothness for stochastic variance reduced methods and sketch-and-project methods for structured linear systems

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°574 mathématiques Hadamard (EDMH)
Spécialité de doctorat : Mathématiques appliquées

Thèse présentée et soutenue à Palaiseau, le 6 décembre 2021, par

Nidham GAZAGNADOU

Composition du Jury :

| | |
|---|---|
| Pascal BIANCHI<br>Professeur, Télécom Paris | Président |
| François GLINEUR<br>Professeur, Université catholique de Louvain | Rapporteur & Examinateur |
| Francis BACH<br>Directeur de recherche, Inria & ENS | Examinateur |
| Claire BOYER<br>Maîtresse de conférences, Sorbonne Université | Examinatrice |
| Martin JAGGI<br>Assistant Professor, École polytechnique fédérale de Lausanne | Examinateur |
| Joseph SALMON<br>Professeur, Université de Montpellier | Directeur de thèse |
| Robert M. GOWER<br>Maître de conférences, Télécom Paris | Invité, Co-encadrant de thèse |

Rapporteurs :

| | |
|---|---|
| François GLINEUR<br>Professeur, Université catholique de Louvain | Rapporteur & Examinateur |
| Michael MAHONEY<br>Associate Adjunct Professor, University of California Berkeley | Rapporteur |



Projet soutenu par la Région Ile-de-France dans le cadre du DIM Math Innov. This work was supported by the Paris Ile-de-France Region via the DIM Math Innov program.

*À ma famille, d'ici et d'ailleurs*

# Remerciements

Bien évidemment, mon projet doctoral est avant tout une aventure scientifique, mais il n'aurait pas pu voir le jour dans d'aussi bonnes conditions sans mes collègues, sans mes amis, sans ma famille.

Mes premiers remerciements vont à mon directeur de thèse Robert Gower et à mon codirecteur Joseph Salmon. Robert, thank you very much for everything you taught me and for all the time you dedicated to me. I now realize how lucky I was to have an advisor as available as you were to discuss about science, projects and broader subjects. You have always been attentive to my academic success while caring about my working conditions and my well-being. I am grateful for that. Joseph, merci beaucoup pour ton soutien pendant cette thèse, pour ce goût pour le détail que tu as conforté chez moi et surtout pour toutes les discussions que nous avons pu avoir sur les "à-côtés" de la recherche. Ces échanges m'ont beaucoup aidé lorsque j'ai eu à faire des choix importants.

I want to thank François Glineur and Michael Mahoney for their careful review of my thesis. François, merci beaucoup pour tes commentaires qui m'ont permis d'améliorer ce manuscrit. I am also grateful to Francis Bach, Pascal Bianchi, Claire Boyer and Martin Jaggi for accepting to be members of my jury. Francis, je tiens à te remercier tout particulièrement pour nos échanges en fin de master. À cette époque, j'hésitais à me lancer dans un projet doctoral et tu as su m'orienter vers un choix que je suis aujourd'hui loin de regretter.

Many thanks to my collaborators: Othmane Sebbouh and Mark Ibrahim. Othmane, j'ai beaucoup admiré ta ténacité face aux preuves récalcitrantes. Sache que c'est avec le sourire que je repense parfois à notre séjour berlinois. Dear Mark, I really want to express how grateful I am regarding all the software skills you taught me. It was a pleasure to work with you and your permanent good mood.

Je salue mes collègues de Télécom Paris qui ont rendu ce projet doctoral encore plus agréable en me partageant leur expérience lors de colloques, de la préparation de TDs ou de déjeuners passés ensemble. Je remercie tout particulièrement Pascal, Umut, Olivier, Jean-Charles, François et Alexandre de m'avoir transmis leur goût de l'enseignement et de la recherche.

Un grand merci à tous les doctorants et post-doctorants du LTCI. Merci pour cette bonne humeur constante qui y régnait ainsi que pour votre aide. Plus spécifiquement merci pour votre soutien lors de la rentrée scolaire quand nous affrontions les deux plus grandes épreuves du doctorant que sont la réinscription et la demande de cumul d'enseignement. Je suis heureux d'avoir partagé avec vous ces verres autour desquels nous échappions à la relative solitude de la recherche.

Je suis aussi très reconnaissant envers le personnel administratif du département IDS et de la formation doctorale de Télécom Paris, ainsi qu'envers l'École Doctorale de Mathématiques Hadamard qui m'a accompagné dans divers projets comme mon apprentissage du persan à l'INALCO. Je remercie également

le DIM Math'Innov, qui a financé ce projet doctoral ainsi que les bibliothécaires des salles R et S de la Bibliothèque Nationale de France. Plus globalement, je tiens à exprimer ma gratitude envers le système éducatif français qui m'a permis de m'épanouir dans mes études. J'ai une pensée particulière pour les différents professeurs qui m'ont inspiré, conseillé et pris sous leur aile.

Je salue bien évidemment mes amis de lycée et de l'ENSTA Paris. Votre compagnie m'a énormément porté pendant ces trois années, mais vous citer tous serait aussi fastidieux que périlleux. Vous m'avez souvent permis de relativiser les problèmes rencontrés lors de mes recherches et m'avez encouragé à prendre soin de moi malgré les efforts nécessaires.

Je remercie tout particulièrement Louis et Grégoire qui ont toujours été à l'écoute et dont l'amitié m'est si précieuse. J'ai hâte de repartir en randonnée avec vous et d'enfin rattraper mon retard dans le TGTG game. Je salue mon groupe d'amis du Mayflower: un grand merci à Jules, Quentin et Mathieu grâce à qui j'ai pu affûter mes talents de chanteur de karaoké avant mon départ pour le pays du Soleil-Levant. Jules, encore merci pour tes petites attentions depuis toutes ces années. Choisir entre tous ces beaux manteaux sera un exercice plus que difficile. Nos échanges réguliers sur nos expériences de thèse m'ont beaucoup aidé pendant ce doctorat. J'ai aussi une pensée pour vous, Amaury et Kimia. Quelle chance d'être arrivé dans le "B412 et son annexe"! Je suis heureux d'être votre ami et toujours partant pour écouter les anecdotes de Jam sessions d'Amaury autour d'une IPA au Diamant.

Enfin, je remercie ma famille du fond du cœur. Elle m'a toujours soutenu dans mes études et c'est sans nul doute grâce à son engagement dans mon parcours scolaire que j'ai pu en arriver là.

Je tiens à remercier mon père, qui bien qu'étranger à tout concept mathématique, m'a transmis la curiosité des sciences humaines, le goût des podcasts de France Culture ainsi qu'un certain nombre de valeurs que j'espère avoir réussi à intégrer. Mes prochains remerciements vont à ma belle-mère, Marjan. Je te dois tellement. Je te dois une nouvelle culture, une nouvelle langue, une nouvelle famille. Cette thèse est le produit d'un attrait pour la connaissance que vous avez tous les deux su faire naître en moi, mais elle n'aurait pas pu voir le jour sans le cadre de vie prospère que vous m'avez assuré. Mina, ma sœur, ta joie de vivre m'a porté toutes ces années. En grandissant et sans le savoir, c'est toi qui m'apprends énormément.

Mes derniers mots seront pour ma mère. Maman, jamais je ne pourrai être assez reconnaissant de l'énergie que tu m'as consacrée. Ton soutien indéfectible, mêlé à ta franchise, m'a permis de me construire. Il serait impudique de m'étendre plus sur la gratitude que j'éprouve envers toi. J'espère simplement que tu es fière de moi, de ce que je suis devenu.

# Table of Contents

**Notation** **11**

**1 Introduction** **13**
1.1 Stochastic optimization for supervised learning . . . 13
1.1.1 Empirical risk minimization . . . 13
1.1.2 Gradient descent method . . . 14
1.1.3 Stochastic gradient algorithms for high dimensional problems . . . 16
1.1.4 Variance reduced stochastic gradients . . . 18
1.1.5 Extrapolating between stochastic and deterministic gradients methods . . . 21
1.2 The sketch-and-project method for solving structured linear system . . . 22
1.2.1 The sketch-and-project method . . . 23
1.2.2 SGD interpretation of sketch-and-project . . . 25
1.2.3 Randomization of ADI methods through sketch-and-project . . . 25
1.3 Contributions . . . 26
1.4 Publications . . . 27

**Part I Extended analysis of variance reduced gradient methods through expected smoothness and residual** **29**

**2 Optimal mini-batch and step sizes for SAGA** **31**
2.1 Introduction . . . 31
2.2 Background . . . 33
2.2.1 Controlled stochastic reformulation and JacSketch . . . 33
2.2.2 The expected smoothness constant . . . 34
2.2.3 Mini-batch without replacement: $b$–nice sampling . . . 35
2.3 Upper bounds on the expected smoothness . . . 36
2.3.1 Assumptions and notation . . . 36
2.3.2 Path to the optimal mini-batch size . . . 38
2.3.3 Estimating the expected smoothness . . . 39
2.4 Optimal mini-batch sizes . . . 48
2.5 Numerical study . . . 49
2.5.1 Upper-bounds of the expected smoothness constant . . . 50
2.5.2 Related step size estimation . . . 51
2.5.3 Comparison with previous SAGA settings . . . 51
2.5.4 Optimality of our mini-batch size . . . 51
2.6 Conclusions . . . 52
2.7 Technical tools and proofs . . . 52
2.7.1 Linear algebra tools . . . 52
2.7.2 Matrix Bernstein inequality for sampling without replacement . . . 53
2.8 Additional experiments . . . 58

**3 Towards closing the gap between the theory and practice of SVRG** **73**
3.1 Introduction . . . 73

3.2 Background and contributions . . . 74
3.3 Assumptions and sampling . . . 76
3.4 Free-SVRG: freeing up the inner loop size . . . 77
3.4.1 Convergence analysis . . . 77
3.4.2 Total complexity for $b$–nice sampling . . . 87
3.5 L-SVRG-D: a decreasing step size approach . . . 88
3.6 Optimal parameter settings: loop, mini-batch and step sizes . . . 93
3.6.1 Optimal loop size for Free-SVRG . . . 93
3.6.2 Optimal mini-batch size for L-SVRG-D . . . 94
3.6.3 Optimal mini-batch and step sizes . . . 96
3.7 Experiments . . . 99
3.8 Additional experiments . . . 100
3.8.1 Comparison of theoretical variants of SVRG . . . 100
3.8.2 Optimality of our theoretical parameters . . . 109

**Part II Extension of sketch-and-project methods for ridge regression and ADI model problem 115**

**4 `RidgeSketch`: a fast sketching based solver for large scale ridge regression 117**
4.1 Introduction . . . 117
4.2 Background and contributions . . . 118
4.2.1 Linear system formulation . . . 120
4.2.2 Using a kernel . . . 121
4.3 The sketch-and-project method . . . 122
4.4 Sketching methods and matrices . . . 123
4.4.1 Classical sketches . . . 123
4.4.2 Count and SubCount sketch . . . 124
4.4.3 Subsampled randomized Hadamard transform . . . 125
4.5 Convergence theory . . . 125
4.5.1 Convergence of the iterates . . . 126
4.5.2 Convergence of the residuals with step size $0 < \gamma < 1$ . . . 126
4.6 Momentum . . . 128
4.6.1 Iterate averaging viewpoint . . . 129
4.6.2 Convergence theorem . . . 130
4.6.3 Implementation . . . 131
4.7 Specialized convergence theory for single column sketches . . . 132
4.7.1 Complexity of coordinate descent with and without momentum . . . 133
4.8 RidgeSketch momentum experiments . . . 135
4.8.1 Experiment 1: comparison of different momentum settings . . . 135
4.8.2 Experiment 2: comparison of different types of sketches . . . 137
4.8.3 Experiment 3: comparison against direct solver and conjugate gradients . . . 138
4.9 Acceleration . . . 138
4.9.1 From theory to practical implementation of acceleration . . . 138
4.9.2 Experiments setting the acceleration parameters . . . 140
4.10 `RidgeSketch` package . . . 141
4.10.1 Solving the ridge regression problem . . . 141
4.10.2 Adding sketching methods . . . 141

**5 Randomized ADI methods 143**
5.1 Introduction . . . 143
5.1.1 The Peaceman-Rachford method . . . 143
5.1.2 ADI methods for matrix equations . . . 144
5.2 Sketched Peaceman-Rachford method . . . 146
5.2.1 A randomized Peaceman-Rachford method . . . 147
5.2.2 Error recurrence . . . 148

5.2.3 Particular sketches . . . 149
5.3 Convergence analysis of the SPR method . . . 152
5.3.1 Convergence in expectation for single row sketching . . . 153
5.3.2 Sketch of convergence in $\ell^2$–norm . . . 155
5.4 Sketched ADI method for Sylvester matrix equations . . . 156
5.4.1 Solution to the Sylvester equation and ADI error . . . 157
5.4.2 Sketch-and-project ADI method . . . 157
5.4.3 SADI with block subsampling sketching . . . 159
5.5 Numerical experiments . . . 160
5.5.1 Solving Peaceman-Rachford problem . . . 160
5.5.2 Solving Sylvester matrix equations . . . 162
5.6 Conclusion and future work . . . 166
5.7 Linear algebra tools . . . 167

**Part III Conclusion** **169**

**6 Conclusion and Perspectives** **171**

**Bibliography** **179**

**Part IV Introduction en français** **181**

**7 Introduction** **183**
7.1 Optimization stochastique pour l'apprentissage supervisé . . . 183
7.1.1 Minimisation du risque empirique . . . 183
7.1.2 Méthode de la descente de gradient . . . 184
7.1.3 Algorithmes du gradient stochastique pour les problèmes en grande dimension . . . 186
7.1.4 Algorithmes du gradient stochastique à variance réduite . . . 188
7.1.5 Extrapolation entre les méthodes de gradients stochastiques et déterministes . . . 191
7.2 La méthode de sketch-and-project pour la résolution de systèmes linéaires structurés . . . 192
7.2.1 La méthode de sketch-and-project . . . 193
7.2.2 Interprétation de sketch-and-project comme une méthode SGD . . . 195
7.2.3 Randomisation des méthodes ADI par sketch-and-project . . . 195
7.3 Contributions . . . 197
7.4 Publications . . . 198

# Notation

| | |
|---|---|
| $[n]$ | Shorthand notation for the set $\{1, \dots, n\}$ |
| $\binom{n}{k}$ | Binomial coefficient for integers $n \geq k \geq 0$, $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ |
| $\langle x, y \rangle$ | Standard Euclidean inner product of $x, y \in \mathbb{R}^n$, $\langle x, y \rangle = \sum_{i=1}^n x_i y_i$ |
| $\|x\|_2$ | Standard Euclidean norm of $x \in \mathbb{R}^n$, $\|x\|_2 = \sqrt{\langle x, x \rangle}$ |
| $\|x\|_{\mathbf{W}}$ | Norm defined by symmetric positive definite $\mathbf{W} \in \mathbb{R}^{n \times n}$, $\|x\|_{\mathbf{W}} = \sqrt{\langle \mathbf{W}x, x \rangle}$ |
| $e^i$ | $i$-th canonical basis vector in $\mathbb{R}^n$ |
| $\mathrm{Null}\,(\mathbf{M})$ | null space of $M$, $\mathrm{Null}\,(\mathbf{M}) = \{x \,\vert\, \mathbf{M}x = 0\}$ |
| $\mathrm{Range}\,(\mathbf{M})$ | range space of $\mathbf{M}$, e.g. if $\mathbf{M} \in \mathbb{R}^{m \times n}$ then $\mathrm{Range}\,(\mathbf{M}) = \{\mathbf{M}x \,\vert\, x \in \mathbb{R}^n\}$ |
| $\sigma\,(\mathbf{M})$ | Spectrum of $\mathbf{M}$ |
| $\lambda_{\max}(\mathbf{M})$ | Largest eigenvalue of $\mathbf{M}$ |
| $\lambda_{\min}(\mathbf{M})$ | Smallest eigenvalue of $\mathbf{M}$ |
| $\mathbf{M}^\dagger$ | Moore-Penrose pseudoinverse of $\mathbf{M}$. |
| $\mathbf{Tr}\,(\mathbf{M})$ | Trace of $\mathbf{M}$, e.g. if $\mathbf{M} \in \mathbb{R}^{n \times n}$ then $\mathbf{Tr}\,(\mathbf{M}) = \sum_{i=1}^n \mathbf{M}_{ii}$ |
| $\|\mathbf{M}\|_F$ | Frobenius norm of $\mathbf{M}$, $\|\mathbf{M}\|_F = \sqrt{\mathbf{Tr}\,(\mathbf{M}^\top \mathbf{M})}$ |
| $\|\mathbf{M}\|_2$ | Spectral norm of $\mathbf{M}$, $\|\mathbf{M}\|_2 = \max_{\|v\|_2 = 1} \|\mathbf{M}v\|_2 = \sqrt{\lambda_{\max}(\mathbf{M}^\top \mathbf{M})}$ |
| $\|\mathbf{M}\|_{\mathbf{W}}$ | Weighted matrix norm with symmetric positive definite $\mathbf{W} \in \mathbb{R}^{n \times n}$, $\|\mathbf{M}\|_{\mathbf{W}} = \left\|\mathbf{W}^{1/2}\mathbf{M}\mathbf{W}^{-1/2}\right\|_2 = \max_{\|v\|_{\mathbf{W}} = 1} \|\mathbf{M}v\|_{\mathbf{W}}$ |
| $\mathbf{A} \otimes \mathbf{B}$ | Kronecker product of $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times q}$, $\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \dots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \dots & a_{mn}\mathbf{B} \end{bmatrix} \in \mathbb{R}^{pm \times qn}$ |
| $\mathrm{vec}\,(\mathbf{A})$ | vectorization of $\mathbf{A}$ stacking its columns, if $\mathbf{A} \in \mathbb{R}^{m \times n}$ then $\mathrm{vec}\,(\mathbf{A}) = \begin{bmatrix} \mathbf{A}_{:1} \\ \vdots \\ \mathbf{A}_{:n} \end{bmatrix} \in \mathbb{R}^{mn}$ |
| $\mathbb{E}\,[\cdot]$ | Expectation operator |
| $\mathbb{E}_k\,[\cdot]$ | Expectation operator conditional on the iterations up to $k$ |
| $\nabla^2$ | Laplace operator |

CHAPTER 1

# Introduction

Ça a débuté comme ça. Moi, j'avais jamais rien dit. Rien.

L.-F. Céline (1932). Voyage au bout de la nuit

## 1.1 Stochastic optimization for supervised learning

### 1.1.1 Empirical risk minimization

In many industrial, economical or scientific applications where a significant amount of data is available, machine learning is used to accomplish a specific task it was *trained* for. The corresponding learning problem is called *supervised* when the training task is performed on a dataset containing both inputs/feature vectors, *e.g.,* measurements made on patients, and outputs/targets, *e.g.,* a variable indicating the progression of their diabetes[1]. Assuming this dataset contains $n \in \mathbb{N}$ samples and that each one belongs to a $d \in \mathbb{N}$ dimensional space, we denote the dataset by $\{(x_1, y_1), ..., (x_n, y_n)\}$ where $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$. In this thesis, the model is represented by a weight vector $w \in \mathbb{R}^d$.

**Empirical Risk Minimization.** In order to fit the model, practitioners define (possibly regularized) loss functions $f_i : \mathbb{R}^d \to \mathbb{R}$ based on pairs $(x_i, y_i)$ for $i = 1, \dots, n$, that quantify the quality of the prediction. The worst the prediction is, the larger the loss. This is how, most of the time solving the training problem boils down to an optimization one: minimizing the loss over available data. We call this resulting optimization problem is called *Empirical Risk Minimization (ERM)*:

$$w^* \in \arg\min_{w \in \mathbb{R}^d} \left( f(w) := \frac{1}{n} \sum_{i=1}^{n} f_i(w) \right) , \tag{1.1}$$

where we assume there exists an optimal weight vector $w^*$. We underline that in machine learning applications, the goal is not to estimate the minimum value $f(w^*)$ but finding a model close to $w^*$ that can be used for prediction on unseen data. Obviously, if $f$ is regular enough, approximating $w^*$ implies approximating $f(w^*)$. See [136, 44] for a broader introduction to supervised learning and notions of learning theory, testing and model selection that we omit in this manuscript.

**Solving the ERM problem.** For some losses the optimization problem has a closed-form solution. For instance, let us consider the quadratic loss $f_i(w) = \frac{1}{2}(x_i^\top w - y_i)^2$. The ERM problem now becomes the famous Ordinary Least-Square (OLS) problem and can be reformulated by

$$w^*_{\text{OLS}} \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{X}w - y\|_2^2 , \tag{1.2}$$

[1] See the `diabetes` dataset available at `https://scikit-learn.org/stable/datasets/toy_dataset.html`.

where $\mathbf{X} \overset{\text{def}}{=} [x_1, \dots, x_n]^\top \in \mathbb{R}^{n \times d}$ denotes the design matrix, $y \overset{\text{def}}{=} [y_1, \dots, y_n] \in \mathbb{R}^n$ the targets vector. By looking at the first-order optimality condition, we get that if $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{d \times d}$ is nonsingular, the solution to (1.2) is

$$w^*_{\text{OLS}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top y \ , \tag{1.3}$$

**Necessity of iterative methods.** In most cases one cannot solve exactly and directly the ERM problem. Either because it is impossible to solve directly the optimality conditions, *e.g.,* logistic regression does not admit a closed-form solution. Or simply because direct solvers are too costly, for instance in (1.3), forming $\mathbf{X}^\top \mathbf{X}$ costs $\mathcal{O}\left(nd^2\right)$ which can be impractical in the big data regime where $n, d \gg 1$. This regime is typically met in many machine learning applications dealing with huge amounts of high dimensional data, like in web advertisement datasets with millions of samples and hundred of thousands of features (some of them are freely accessible on LIBSVM [26][2]. In such cases, practitioners instead apply iterative methods that produce a sequence of $\left(w^k\right)_k$ which gets closer to a solution across iterations. One could legitimately wonder if solving approximately (1.1) is satisfactory enough. In fact, in statistical learning, the optimization error does not need to reach machine precision since it is also balanced by the estimation and approximation errors. See [19] for more details about this error decomposition framework. The canonical iterative methods to use are the first-order/gradient methods.

### 1.1.2 Gradient descent method

In order to apply first-order algorithms to solve the ERM problem, some structure on the optimization problem is required. The assumptions we will now make on the problem ensure that it admits at least a solution and allow to measure efficiency of methods through there convergence speed.

**General assumptions.** We assume differentiability, $L$–smoothness and $\mu$-strong convexity (which is typically the case of regularized problems) of $f$, such that (1.1) admits a unique solution $w^*$:

**Assumption 1.1** (Smoothness)**.** *There exist $L \geq 0$ such that for all $x, y \in \mathbb{R}^d$,*

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|_2^2 \ . \tag{1.4}$$

*We say that $f$ is $L$–smooth.*

**Assumption 1.2** (Strong convexity)**.** *There exist $\mu \geq 0$ such that for all $x, y \in \mathbb{R}^d$,*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \|y - x\|_2^2 \ . \tag{1.5}$$

*We say that $f$ is $\mu$–strongly convex.*

**Gradient Descent.** The Gradient Descent (GD) algorithm starts from an initial point $w^0 \in \mathbb{R}^d$ and minimizes a differentiable and convex function by iteratively taking steps as follows:

$$w^{k+1} = w^k - \gamma \nabla f(w^k) \ , \tag{1.6}$$

where $\gamma > 0$ is a constant step size. The intuition behind GD is that at each iterate $w^k$, the negative first derivative indicates a local decrease direction in function values. In Figure 1.1, we show the the level sets and the first 100 iterates $(w^k)$ of GD with a step size $\gamma = 1/L$ on a $\ell^2$–regularized logistic regression problem with synthetic data ($n = 100000$, $d = 2$, and regularization parameter $\lambda = 1/\sqrt{n}$)[3]. We observe that GD is a proper *descent method* as it decreases the loss at every iteration. GD and its variants (proximal, projected, accelerated, etc.) are called *full batch* methods, since they require to access all data points at each iteration. Indeed, this method requires the computation of the gradient of the entire loss, *i.e.,* $\nabla f(w^k) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(w^k)$, which costs $n$ single gradient computations. We call

[2] Available at `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

[3] Corresponding code is available at: `https://github.com/ngazagna/plotOpt`.

**Algorithm 1** Gradient Descent

**Input:** step size $\gamma > 0$
**Initialize:** $w^0 \in \mathbb{R}^d$
**for** $k = 0, 1, \ldots, K-1$ **do**
  $w^{k+1} = w^k - \gamma \nabla f(w^k)$ ▷ $\mathcal{O}(d)$
**Output:** $w^K$

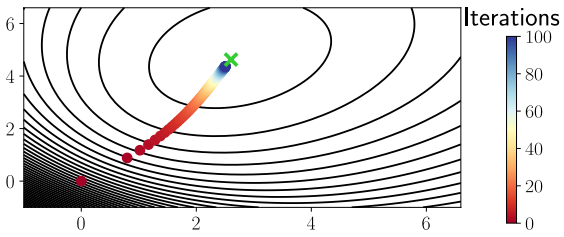



**Figure 1.1:** *GD iterates with $\gamma = 1/L$ plotted on level sets of $f$.*

this quantity the *per-iteration complexity* or *iteration cost*. For an exhaustive presentation of full batch first-order methods, we refer to [13].

In Algorithm 1, we provide the implementation of GD algorithm with the cost of full gradient computation. Note that in this manuscript, we often present iterative algorithms using for loops, thus omitting the stopping criterion which usually depends on the norm of the gradient at the current iterate $\left\|\nabla f(w^k)\right\|_2$ or on loss evaluation $f(w^k)$. As the objective function $f$ is the average of $n$ loss functions, its evaluation tends to be very costly when $n \gg 1$. This is why practitioners avoid computing $f$ throughout iterations in order not to slow down convergence.

**Convergence and complexity.** To compare optimization methods we can measure how fast they approaches the solution. There are two classical ways of estimating if an iterate is close to a solution. Firstly, by computing its *suboptimality* $f(w^k) - f(w^*)$, which is by definition a nonnegative quantity. Secondly, by measuring its *distance to the minimizer* $\left\|w^k - w^*\right\|_2^2$ or to the set of minimizers if uniqueness is not met. In the convex or non-convex settings, looking at suboptimality is more relevant as there might exist several solutions of the ERM problem (1.1). In this thesis, since we assume strong convexity most of the time, there is a unique minimizer and we focus on the distance to the minimizer. Next we provide the classical result of convergence of gradient descent under Assumptions 1.1 and 1.2 which is given in many textbooks such as in Theorem 3.10 of [24], Theorem 5.1 of [9], Theorem 10.29 [13] in the proximal case or Section 9.3.1 of [22] for GD with line search.

**Theorem 1.3** (Convergence of GD in distance to minimizer)**.** *Let the initial point be $w^0 \in \mathbb{R}^d$. If $f$ is $L$–smooth and $\mu$–strongly convex and if the step size is set to $\gamma = 1/L$, then, the iterates generated by the gradient descent method converge according to*

$$\left\|w^k - w^*\right\|_2^2 \leq \left(1 - \frac{\mu}{L}\right)^k \left\|w^0 - w^*\right\|_2^2 \quad . \tag{1.7}$$

*Proof.* We first develop the error term at step $k \geq 1$ using the gradient descent update given in (1.6):

$$\begin{aligned}
\left\|w^k - w^*\right\|_2^2 &= \left\|w^{k-1} - w^* - \gamma \nabla f(w^{k-1})\right\|_2^2 \\
&= \left\|w^{k-1} - w^*\right\|_2^2 - 2\gamma \left\langle \nabla f(w^{k-1}), w^{k-1} - w^* \right\rangle + \gamma^2 \left\|\nabla f(w^{k-1})\right\|_2^2 \\
&\overset{(1.5)}{\leq} (1 - \gamma\mu) \left\|w^{k-1} - w^*\right\|_2^2 - 2\gamma(f(w^{k-1}) - f(w^*)) + \gamma^2 \left\|\nabla f(w^{k-1})\right\|_2^2 \\
&\overset{(1.4)}{\leq} (1 - \gamma\mu) \left\|w^{k-1} - w^*\right\|_2^2 - 2\gamma(1 - \gamma L)(f(w^{k-1} - f(w^*)) \quad .
\end{aligned}$$

By setting the step size to $\gamma = \frac{1}{L}$, we get that for every $k \geq 1$

$$\left\|w^k - w^*\right\|_2^2 \leq (1 - \gamma\mu) \left\|w^{k-1} - w^*\right\|_2^2 \quad .$$

□

The type of convergence given in Theorem 1.3 is known as *linear convergence*. In order to reach an $\epsilon$–solution, that is to reach a $w^k$ such that the relative error $\left\|w^k - w^*\right\|_2^2 / \left\|w^0 - w^*\right\|_2^2$ is smaller than $\epsilon > 0$, (1.7) implies that one must run at least

$$\frac{L}{\mu} \log\left(\frac{1}{\epsilon}\right) \tag{1.8}$$

iterations. From this *iteration complexity* result one can see that the larger the condition number $\kappa \overset{\text{def}}{=} L/\mu$, the slower the gradient descent. In fact, $\kappa$ appears in all convergence analysis and quantifies the difficulty to solve the ERM problem in (1.1). However, $\mathcal{O}\left(\frac{L}{\mu}\log\left(\frac{1}{\epsilon}\right)\right)$, where the $\mathcal{O}$ notation hides constants terms, is not the best complexity that first-order methods can achieve. A lower bound result [106] showed that the optimal complexity of gradient methods is $\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\epsilon}\right)\right)$, which is actually achieved by Nesterov's Accelerated Gradient Descent method [108, 24].

**Remark 1.4.** *Note that the result in Theorem 1.3 is not tight. It is just given as a pedagogical example. Theorems and 2.1 and 3.1 in [145] show that the convergence rate in* (1.7) *can be replaced by* $\left(1 - \frac{\mu}{L}\right)^{2k}$.

**Impractical in large scale settings.** However, focusing on the iteration complexity can be misleading since it does not take into account the cost of a single iteration. We then introduce the *total complexity*: the per-iteration cost times the iteration complexity, which equals $\mathcal{O}\left(n\kappa\log\left(\frac{1}{\epsilon}\right)\right)$ for GD. Thus, in the large scale regime, that is when the number of samples $n$ is very large, one observes that applying GD to solve the ERM problem becomes a hard and time-consuming task. In this regime, stochastic algorithms with cheap updates are preferred to gradient descent as we will see in next section.

### 1.1.3 Stochastic gradient algorithms for high dimensional problems

The considerable increase in the number of data samples $n$ of recent machine learning applications (such as in web advertisement and bioinformatics) rules out the use of *full batch* gradient methods. Indeed, solving the ERM problem becomes too time consuming since these algorithms need to scan through all the data points at each iteration, which make them not competitive. Furthermore, the goal in machine learning applications is not just solve exactly the training problem but finding a model which will perform well on unseen data. In other words, our goal is to solve the ERM such that the optimization error reaches the statistical error, *i.e.,* the inherent error due to the choice of the statistical model. As described in [19], stochastic first-order optimization methods are particularly well suited for objectives which are averages of losses and lead to faster convergence than full batch methods while generalizing better on unseen data. These past years, stochastic gradients methods have been popularized by the Deep Learning community with their use in training efficiently large neural networks [85].

These reasons have popularized the Stochastic Gradient Descent (SGD) method, introduced in [126], to solve incrementally the ERM problem. This method only require a unbiased estimate of the full gradient at each iteration, which is easily accessible through the gradient of a single loss $f_i$ where $i$ is sampled randomly in $[n] \overset{\text{def}}{=} \{1, \dots, n\}$:

$$w^{k+1} = w^k - \gamma_k \nabla f_i(w^k) \ , \tag{1.9}$$

where $\gamma_k > 0$ is the step size at iteration $k$. The SGD algorithm is given in Algorithm 2 and its first 1000 iterates with constant step size on the same logistic regression problem are displayed in Figure 1.2. An important characteristic to underline, visible in Figure 1.2, is that Stochastic Gradient "Descent" is a misnomer because this method does not meet any descent property, unlike GD for which at step $k$

$$f(w^{k+1}) \overset{(1.6)}{=} f(w^k - \gamma\nabla f(w^k)) \overset{(1.4)}{\leq} f(w^k) + \gamma\left(\gamma\frac{L}{2} - 1\right)\left\|\nabla f(w^k)\right\|_2^2 \leq f(w^k)$$

as soon as $0 < \gamma \leq 1/(2L)$.

**Simple convergence of SGD.** SGD appears to be an alternative to full-batch methods because it enjoys a very cheap per-iteration complexity: $\mathcal{O}(1)$ as opposed to $\mathcal{O}(n)$ for GD. But in order to

**Algorithm 2** Stochastic Gradient Descent

**Input:** sequence of positive step sizes $(\gamma_k)_k$
**Initialize:** $w^0 \in \mathbb{R}^d$
**for** $k = 0, 1, \ldots, K-1$ **do**
  Sample $i \sim [n]$
  $w^{k+1} = w^k - \gamma_k \nabla f_i(w^k)$ ▷ $\mathcal{O}(1)$
**Output:** $w^K$

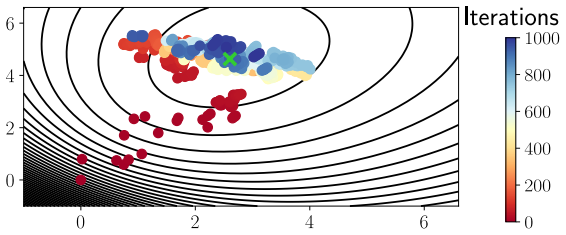



**Figure 1.2:** *SGD iterates with $\gamma = 1/L$ plotted on level sets of $f$.*

compare these two methods, we need more than the cost of an iteration: we need to compare their total complexities. Thus, let us give a similar convergence result for SGD with constant step size than in Theorem 1.3.

**Theorem 1.5** (Convergence of SGD in distance to minimizer)**.** *Let the initial point be $w^0 \in \mathbb{R}^d$. Let us assume that we have uniformly bounded stochastic gradient on the path of SGD* [4]*:*

$$\mathbb{E}_k \left[ \left\| \nabla f_i(w^k) \right\|_2^2 \right] \leq B^2, \quad \forall i \in [n], \forall k \in \mathbb{N}^* \tag{1.10}$$

*where $(w^k)_k$ are the iterates generated by SGD and $B > 0$. If $f$ $\mu$–strongly convex and if the step size is set such that $0 < \gamma < 1/\mu$, then, the iterates generated by the gradient descent method converge according to*

$$\mathbb{E}\left[ \left\| w^k - w^* \right\|_2^2 \right] \leq (1 - \gamma\mu)^k \left\| w^0 - w^* \right\|_2^2 + \frac{\gamma}{\mu} B^2 \ . \tag{1.11}$$

*Proof.* The proof is similar to the one of Theorem 1.3 and is given in [51]. □

**A convergence tradeoff.** We recover the left-hand side term which is the same as in (1.7) for GD (with $\gamma = 1/L$) and decreases exponentially with $k$. In the stochastic case we get a second remaining *residual or noise* term $\gamma B^2/\mu$ which we cannot force to go to zero. Indeed, (1.11) proves the convergence of SGD to a neighborhood of the solution and highlights a tradeoff in the choice of the step size $\gamma$. On the one hand, one should set $\gamma$ closed to $1/\mu$, which can be very large as often $\mu$ is near to zero, to make the left-hand side term go rapidly to zero. But on the other hand, $\gamma$ should be close to 0 to cancel the noise term. This tradeoff is often known as the opposition between *forgetting the initial conditions*, *i.e.,* escaping fast from $w^0$ to approach $w^*$, and *cancelling the noise*, *i.e.,* reducing the radius of the ball around the solution that SGD reaches.

**Complexity comparison with GD.** More advanced results [10, 103, 58] show the sublinear convergence of SGD with constant step size under strongly-convex assumption on the average loss $f$ and $L_i$-smoothness assumption on the $f_i$'s. Let $L_{\max} \overset{\text{def}}{=} \max_{i=1,\ldots,n} L_i$. This implies that the iteration complexity of SGD, which equals its total complexity since only a single gradient is computed at each iteration, is

$$\mathcal{O}\left( \left( \frac{L_{\max}}{\mu} + \frac{\sigma^2}{\mu^2 \epsilon} \right) \log\left( \frac{1}{\epsilon} \right) \right) \ , \tag{1.12}$$

where $\sigma^2 \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \left\| \nabla f_i(w^*) \right\|_2^2$ is the *gradient noise*. We recall from (1.8) that GD enjoys a linear convergence rate leading to a total complexity of $\mathcal{O}\left( n \frac{L}{\mu} \log\left( \frac{1}{\epsilon} \right) \right)$ as we have seen in the previous section. Hence, this comparison suggests to apply SGD to get a low accuracy solution, but for high accuracy results when $\epsilon$ is small, GD method is to be preferred.

[4] It is worth noting that if this assumption is made at all points $w$ it conflicts with the following strong convexity assumption on $f$, see [110, 20].

**Enforcing convergence of SGD.** SGD with a constant step size still suffers from not converging exactly to the solution. For example, if one initialize Algorithm 2 at $w^0 = w^*$ the solution of the ERM problem (1.1), the next iterates will move away from there and oscillate around it can be seen as for the last iterates in Figure 1.2. This is due to the fact that individual gradients $\nabla f_i(w^*)$ are not necessarily null at optimum[5] as contrary to the full batch gradient $\nabla f(w^k)$ which converge to 0 as $w^k \to w^*$.

Several options can correct this oscillating behaviour and enforce convergence. Instead of outputting last iterate, which is the most natural design of gradient methods, one can return an average of the iterates, *i.e.,* perform *Polyak-Ruppert averaging* [119, 130, 136]. Yet, this option is less practical since it requires storing in memory an average of previous iterates which is cumbersome for very large models like Deep Learning model where the number of weights can be extremely large. Another strategy consists in using decreasing step sizes $\gamma_k$, typically of the order $\mathcal{O}\left(1/\sqrt{k}\right)$ or $\mathcal{O}(1/k)$, to cancel gradient noise out when approaching the solution. However, this sequence of step sizes is often hard to tune and can be very time-consuming for practitioners. Note that recent results on SGD in [58], lead to a new step size policy, constant then decreasing in $\mathcal{O}(1/k)$, which achieve sublinear convergence while cancelling gradient noise.

### 1.1.4 Variance reduced stochastic gradients

In the past several years, part of the optimization community tried to get efficient linear convergence of full batch gradient descent while keeping the cheap iteration cost of SGD. Authors of [129, 133] succeeded "to get the best of both worlds" by designing and analyzing the Stochastic Average Gradient (SAG) method, a *stochastic variance reduced (VR) gradient* algorithm that solves (1.1) with linear convergence by exploiting its finite sum structure. They were then followed by many other methods such as SVRG [71], SAGA [33], the unbiased version of SAG, SDCA [139], FINITO/MISO [35, 95], SVRG++ [167]. We refer to the review paper [57] for more details on variance reduced methods for solving machine learning problems. Note that for variance reduced methods, the expected bounded stochastic gradients assumption (1.10) is not necessary anymore. This is preferable since assuming in advance a property on the iterates generated by an algorithm to prove the convergence of the latter is a kind of circular argument one seeks to avoid.

**Variance reduction via control covariates.** In all these methods, it is shown that the convergence of SGD can be faster if one replaces $\nabla f_i(w^k)$ by a better stochastic estimate $g^k \in \mathbb{R}^d$ of the full gradient. This estimate is typically chosen to be unbiased, *e.g.,* $\mathbb{E}\left[g^k\right] = \nabla f(w^k)$, and such that it has a smaller variance than $\nabla f_i(w^k)$ by leveraging stochastic gradient information of previous steps. Reducing the variance of the gradient estimate across iterations, *i.e.,* $\lim_{w^k \to w^*} \mathrm{Var}(g^k) = 0$, then allows to use constant step size, making stochastic VR gradient methods not only faster but also more practical than SGD, which requires decreasing ones.

Let us explain briefly how $\mathrm{Var}(g^k) \overset{\text{def}}{=} \mathbb{E}\left[\left\|g^k - \nabla f(w^k)\right\|_2^2\right]$ can be reduced. First, let $h_i : \mathbb{R}^d \to \mathbb{R}$, for $i = 1, \dots, n$, be $n$ that we call *covariate functions*. Let $i$ be sampled i.i.d. from $[n]$, then we can reformulate the objective function as

$$\frac{1}{n}\sum_{i=1}^{n} f_i(w) = \mathbb{E}\left[f_i(w)\right] = \mathbb{E}\left[f_i(w) - h_i(w) + \mathbb{E}\left[h_i(w)\right]\right] \quad . \tag{1.13}$$

Now at iteration $k$, instead of applying a SGD step of the ERM problem on the left of (1.13), let us apply it to the right of (1.13), that we call the *controlled stochastic reformulation* [56] of the ERM problem, to build a new estimate of the full gradient:

$$g^k \overset{\text{def}}{=} \nabla f_i(w^k) - \nabla h_i(w^k) + \mathbb{E}\left[\nabla h_i(w^k)\right] \quad .$$

By design, $g^k$ is an unbiased estimate of the gradient as $\mathbb{E}\left[g^k\right] = \nabla f(w^k)$, and its variance equals

$$\mathrm{Var}(g^k) = \mathrm{Var}(\nabla f_i(w^k)) - 2\,\mathrm{Cov}(\nabla f_i(w^k), \nabla h_i(w^k)) + \mathrm{Var}(\nabla h_i(w^k)) \quad .$$

[5] Except in so-called *over-parametrized models* that we do not study in this thesis.

**Algorithm 3** SAGA

**Input:** step size $\gamma_k > 0$
**Initialize:** $w^0 \in \mathbb{R}^d$,
$g_j = \nabla f_j(w^0)$ for $j = 1, \dots, n$ ▷ $\mathcal{O}(n)$
$G = \frac{1}{n}\sum_{j=1}^n g_j$
**for** $k = 0, 1, \dots, K-1$ **do**
  Sample $i \sim [n]$
  $g^k = \nabla f_i(w^k) - g_i + G$ ▷ $\mathcal{O}(1)$
  $w^{k+1} = w^k - \gamma g^k$
  $G = G + \frac{1}{n}(\nabla f_i(w^k) - g_i)$
  $g_i = \nabla f_i(w^k)$
**Output:** $w^K$

**Algorithm 4** SVRG

**Input:** step size $\gamma > 0$, inner-loop size $m$
**Initialize:** $z^0 \in \mathbb{R}^d$
**for** $k = 0, 1, \dots, K-1$ **do**
  $z = z^k$
  $G = \nabla f(z)$ ▷ $\mathcal{O}(n)$
  $w^0 = z$
  **for** $t = 0, 1, \dots, m-1$ **do**
    Sample $i \sim [n]$
    $g^k = \nabla f_i(w^k) - \nabla f_i(z) + G$ ▷ $\mathcal{O}(1)$
    $w^{k+1} = w^k - \gamma g^k$
  $z^{k+1} = w^m$
**Output:** $w^m$

Thus, the variance of $g^k$ is smaller the that of the stochastic gradient if $\nabla h_i(w^k)$ and $\nabla f_i(w^k)$ are sufficiently positively correlated. So, in VR methods we roughly want to have $\nabla h_i(w^k) \approx \nabla f_i(w^k)$ where $\nabla h_i(w)$ is based on past gradient computations. To fulfill these requirements, the covariates functions $h_i$ are often selected as linear approximations of the $f_i$'s, as we will see below for SAGA and SVRG. It is worth noting that even if we set $\nabla h_i(w^k) = \nabla f_i(w^k)$, then it takes us back to full batch GD as

$$g^k = \nabla f_i(w^k) - \nabla f_i(w^k) + \mathbb{E}\left[\nabla f_i(w^k)\right] = \nabla f(w^k) \ ,$$

which, certainly, has a null variance but that we want to avoid because of its expensive per-iteration cost. Finally, a key aspect of VR methods to keep in mind is that they introduce a notion of *memory*, information must be stored across iterations to be reused as control covariates.

**Example of SAGA and SVRG.** Let us now detail two famous VR methods studied in depth in Chapters 2 and 3. First, SAGA [33] is an algorithm which keeps in memory an $n \times d$ table of previously computed individual gradients $\nabla f_j$ for each sample $j \in [n]$. Then, at step $k$, SAGA's update uses the average of the stored gradients to correct the freshly computed stochastic gradient $\nabla f_i(w^k)$:

$$g^k = \nabla f_i(w^k) - \nabla f_i(z_i^k) + \frac{1}{n}\sum_{j=1}^n \nabla f_j(z_j^k) \ , \tag{1.14}$$

where $z_j^k \in \mathbb{R}^d$ represents the last point at which $\nabla f_j$ has been computed. Here, $h_i(w) = w^\top \nabla f_i(z_i^k)$ and hence we have $\nabla h_i(w^k) = \nabla f_i(z_i^k)$. Note that in practice, these points $z_j^k$ are never stored, only the corresponding $n$ gradients values $\nabla f_j(z_j^k) \in \mathbb{R}^d$ are kept in memory. Indeed, this is exactly how SAGA algorithm is written in Algorithm 3: these gradient are stored in the $g_j$ variables, forming a $d \times n$ table, and their average $G$ is computed on the fly. The main downside of SAGA is this extra memory cost of $\mathcal{O}(nd)$ which makes this method impractical in the case of very large scale problems[6] [34]. This last point is important because it can be a barrier to applying VR methods on large neural networks, for which $d \gg 1$, trained on millions of samples, *i.e.,* $n \gg 1$ too.

Regarding SVRG [71], as shown in Algorithm 4, it has a memory cost of $\mathcal{O}(d)$ since it stores a *snapshot* point $z \in R^d$ and the full gradient at this point $\nabla f(z) \in \mathbb{R}^d$, also called the *reference gradient*. Every $m$ iterations, where $m$ is called the *inner loop size*, the snapshot $z$ is reset to the current iterate $w^k$ and the reference full gradient gets computed for this new snapshot. The gradient estimate for SVRG equals

$$g^k = \nabla f_i(w^k) - \nabla f_i(z) + \nabla f(z) \tag{1.15}$$

Here, the covariate functions equal $h_i(w) = w^\top \nabla f_i(z)$ such that $\nabla h_i(w^k) = \nabla f_i(z)$. The weakness of

[6] For General Linear Models (GLM), a storing trick allows to reduce this memory cost to $\mathcal{O}(n)$.

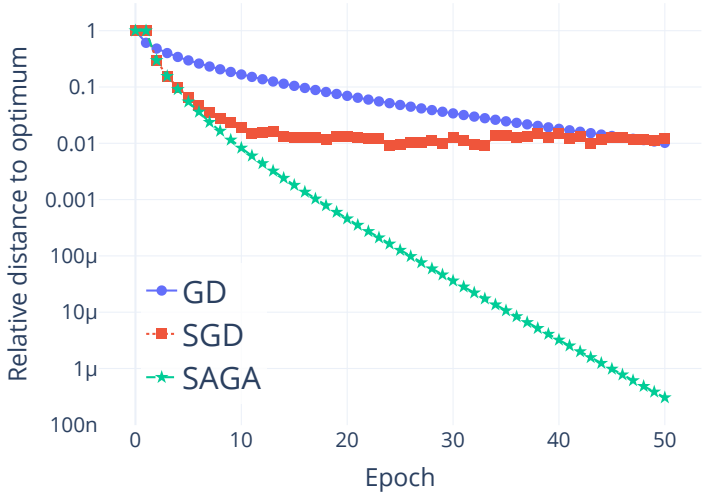



**Figure 1.3:** *Comparison between GD, SGD and SAGA with theoretical step sizes (respectively* $1/L$ *for GD and* $1/L_{\max}$ *for the others) on a synthetic* $\ell^2$*–regularized logistic regression problem* ($n = 1000$, $d = 200$, $\lambda = 1/\sqrt{n}$).

SVRG is that every $m$ inner steps it computes a full batch gradient $\nabla f(z)$ which can be cumbersome as soon as $m$ is small or $n \gg 1$. Note that in (1.15) two stochastic gradients are computed at each step whereas only one is required in SAGA update in (1.14). As exposed in Algorithm 4, SVRG is classically described with an inner and an outer loop even if it is not implemented this way in practice[7]. There exist also a randomized version of SVRG: Loopless SVRG (L-SVRG) [68, 81] which replaces the outer loop by a coin-tossing procedure and update the snapshot $z$ with probability $p$ (equivalent to $1/m$).

**Complexity of VR gradient methods.** Assuming that $f$ is $\mu$-strongly convex and that the $f_i$'s are convex and $L_i$-smooth and recalling that $L_{\max} = \max_{i=1,\dots,n} L_i$. SAGA, SAG, SVRG and many other VR gradient methods enjoy a *fast linear convergence* which is equivalent to a total complexity of

$$\mathcal{O}\left(\left(n + \frac{L_{\max}}{\mu}\right)\log\left(\frac{1}{\epsilon}\right)\right) \quad . \tag{1.16}$$

In comparison to the complexity of GD in (1.8), it means that VR methods are faster than GD as soon as

$$L \geq \mu + \frac{L_{\max}}{n} \quad .$$

Lemma 2.9 shows that $L \geq L_{\max}/n$, so that this condition is often met in machine learning applications where $\mu$ is negligible with respect to $L_{\max}/n$. Figure 1.3 illustrates the efficient linear convergence of SAGA versus SGD, which oscillates around the solution, and GD, which is much slower because of its per-iteration cost. Note that during the first epochs, SGD and SAGA have the same behavior as we did not initialize the $(g_j)_{j=1,\dots,n}$ gradients stored in memory. Throughout the epochs, as the $g_j$'s are updated, variance reduction accelerates the convergence of SAGA. Many variants (proximal, projected, streaming, etc.) of each of these methods have been derived, notably accelerated versions in the sense of Nesterov like Acc-SDCA [138], Point-SAGA [32], Katyusha [3] and Catalyst [88]. All these cited methods enjoy a faster linear convergence with complexity $\tilde{\mathcal{O}}\left(\left(n + \sqrt{n\frac{L_{\max}}{\mu}}\right)\log\left(\frac{1}{\epsilon}\right)\right)$ where the $\tilde{\mathcal{O}}$ notation hides constants and logarithmic terms.

[7]The inner loop can be easily replaced by an if clause with a modulo $m$ operator applied to the number of iterations indicating when to update the snapshot and its gradient.

### 1.1.5 Extrapolating between stochastic and deterministic gradients methods

Stochastic gradient methods have proved their efficiency in machine learning applications such as in Figure 1.3, especially to reach low precision solutions. Due to recent technological advances, especially in the parallelization of computations, another very popular way to accelerate SGD and its variants has been popularized: it is called *mini-batching* [49, 20].

**Mini-batching.** This technique consists in building an estimate the full gradient by computing stochastic gradients on a mini-batch of points $B \subseteq [n]$ rather than at a single one:

$$g^k = \nabla f_B(w) = \frac{1}{|B|} \sum_{i \in B} \nabla f_i(w) \ , \tag{1.17}$$

where $f_B(w) \overset{\text{def}}{=} \frac{1}{|B|} \sum_{i \in B} f_i(w)$ denotes the *subsampled function* for mini-batch $B$. At each iteration this mini-batch $B$, usually of fixed cardinality $b \in [n]$, can be sampled according to different procedures such as *partition sampling*, *i.e.,* data is separated into blocks of size $b$ which are sampled randomly or cyclically, *sampling without replacement* or *sampling with replacement/b-nice sampling* [8] [56] which is the most widespread, and *importance sampling* where most significant samples are sampled more often [120, 165]. In [16], authors were already using blocks of data to solve incrementally least squares problems and emphasis on their use in the training of neural networks exploiting the backpropagation technique. As in [137], mini-batch gradients can be parallelized to speed up the computations, but even in serial use this technique can accelerate the optimization. Questions regarding the impact of mini-batching on generalization performances, especially in deep learning settings [76, 67, 96], are beyond the scope of this thesis.

From (1.17), we remark that if $b = 1$, one recovers SGD (or one of its VR variants) and on the other extreme if $b = n$, one recovers GD. As we have seen, the step size and the convergence rate of GD depend on $L$ in (1.8) whereas $L_{\max}$ plays this role for stochastic VR gradient methods in (1.16). Mini-batching often leads to faster convergence as $g^k$ better approximates the full batch gradient $\nabla f(w^k)$ when $b$ gets closer to $n$. Moreover, one can take larger steps, as we will see in next paragraph, and thus speed up even more the convergence.

**Smoothness and step size.** An important thing to notice is the dependence in $L$ of the convergence of full batch gradient descent in (1.8) whereas $L_{\max}$ plays this role for stochastic VR methods in (1.16). By definition of $L_i$–smoothness of $f_i$ (1.4), we have that

$$f_i(y) \leq f_i(x) + \langle \nabla f_i(x), y - x \rangle + \frac{L_i}{2} \|y - x\|_2^2 , \quad \forall i = 1, \dots, n \ . \tag{1.18}$$

By averaging these $n$ inequalities we get

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\bar{L}}{2} \|y - x\|_2^2 \ , \tag{1.19}$$

where $\bar{L} \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n L_i \leq L_{\max}$. This last inequality (1.19) shows that

$$L \leq \bar{L} \leq L_{\max} \ , \tag{1.20}$$

since $L$ is assumed to be the *smallest Lipschitz constant* satisfying (1.4). We can interpret (1.20) by saying that the average objective loss $f$ we aim at minimizing in (1.1) is "smoother" than the most

[8] A confusion can easily arise when talking about mini-batching *with or without replacement* as it can either refer to a local sampling policy at step $k$ or the global sampling policy across different iterations. In the definition of [124], *b–nice sampling* refers to a local rule where at each iteration $b$ indices are chosen at random without replacement among $[n]$. Whereas in [140], *without-replacement sampling* refers to the action of sampling at each iterations points that have not been sampled in previous iterations. This type of global sampling rule introduces more difficulty in the analysis because functions processed at each iterations are not statistically independent.

irregular individual loss among the $f_i$'s. Noting that the constant step size $\gamma$ is proportional to the inverse of these smoothness constants, it implies that for GD can perform larger steps, $\gamma = 1/L$, than stochastic VR methods, for which $\gamma \propto 1/L_{\max}$. Let $B \subseteq [n]$ be a mini-batch of samples, we recall the subsampled function on $B$ is defined by $f_B(w) = \frac{1}{|B|}\sum_{i\in B} f_i(w)$. Intuitively, $f_B$ is more likely to be smoother than $f_i$ for $i \in [n]$ if $b$ is large enough. This explains why for mini-batch stochastic VR gradient methods, one can take larger steps than when sampling a single point at each iteration. Yet, as we have seen in Figure 1.3 larger mini-batch $b$ and step sizes $\gamma$ do not always mean faster minimization as GD converges much slower than SAGA. Thus, a natural question then arises:

*How does the convergence rate behave in this intermediate mini-batching setting ?*

**Expected smoothness.** To study the effect of general sampling techniques and in particular of mini-batching, recent work [56] introduced a key concept: *expected smoothness*. Instead of assuming that each individual loss $f_i$ is $L_i$–smooth or that $f$ is $L$–smooth like in (1.4) and (1.18), let us assume a more general form of smoothness which measure the regularity of the subsampled function $f_B$. In the case of mini-batching without replacement this assumption can be simplified as follows.

**Assumption 1.6** (Expected Smoothness constant)**.** *Let $B \subseteq [n]$ be a random set of $b$ points sampled without replacement. There exists $\mathcal{L}(b) > 0$ such that*

$$\mathbb{E}\left[\|\nabla f_B(w) - \nabla f_B(w^*)\|_2^2\right] \leq 2\mathcal{L}(b)\left(f(w) - f(w^*)\right) \quad . \tag{1.21}$$

We remark that this property of $\mathcal{L}$ is not exactly defining smoothness in the sense of having Lipschitz first-order derivatives as in (1.4). Instead, its is related to a property of convex and $L$–smooth $f$ functions minimized at $w^*$. Under these assumptions, eq. (2.1.7) in [106] gives us that the variance of the gradient is controlled by the suboptimality as follows

$$\mathbb{E}\left[\|\nabla f(w) - \nabla f(w^*)\|_2^2\right] \leq 2L\left(f(w) - f(w^*)\right) \quad .$$

In its general formulation with arbitrary sampling, $\mathcal{L}$ captures the effect of the entire sampling strategy, which encompasses the sampling probabilities and the mini-batch size. Our analyses in Chapters 2 and 3 leverages results of [56], where the authors proved that the expected smoothness constant $\mathcal{L}$ rules the iteration complexity and the step size of a large class of stochastic methods, including SAGA. Expected smoothness is the key tool allowing us to estimate sharply the total complexity of mini-batch versions of stochastic VR methods under very different sampling strategies.

## 1.2 The sketch-and-project method for solving structured linear system

In numerical analysis, linear systems are ubiquitous in many quantitative fields such as the industry, econometrics or physics and solving them efficiently is a common task. They are usually defined as solving in $w \in \mathbb{R}^m$

$$\mathbf{A}w = b \ , \tag{1.22}$$

where $\mathbf{A} \in \mathbb{R}^{m\times m}$ and $b \in \mathbb{R}^m$[9]. The study of linear system is at the core of linear algebra, and many problems from numerical analysis, computer science, optimization, machine learning, etc. boil down to their resolution. Moreover, solving linear systems is sometimes a subroutine of broader programs or algorithms, such as in Newton's methods in optimization or in Peaceman-Rachford method [113] studied in Chapter 5.

Solving efficiently (1.22) highly depends on the *structure* (sparsity, type of matrix $A$, storage format of $A$) of the problem. For instance, famous Python functions such as `numpy.linalg.solve`[10] [152] for

[9] In this section we will only consider square systems of order $m$.

[10] See `https://numpy.org/doc/stable/reference/generated/numpy.linalg.solve.html`.

dense problems or `scipy.linalg.solve`[11] [154] for sparse matrices (typically in the CSC, CSR or COO format[12]) run the most suited linear system solver (in these cases LAPACK routines [5]) depending whether $\mathbf{A}$ is positive definite, semi-positive definite, symmetric, etc. In some applications, the matrix $A$ can also be inaccessible and only its representation as a linear operator can be used via its matrix-vector product $A : w \mapsto Aw$.

The dimension of the problem, here $m$, also affects the choice of the solver. In small dimensions, direct solvers (often relying on Cholesky, QR or LU factorizations) [48] are preferred even if they enjoy a $\mathcal{O}(m^3)$ cost. However, in large scale settings, one might not be able to load the entire matrix $A$ in RAM and this cubic cost makes them terribly slow.

**Randomized iterative methods.** In order to scale the resolution of linear system to large matrices, recent advances proved the usefulness of *randomization*. In this sense, parallels can be drawn between the randomization of GD leading to SGD and the one of classical deterministic iterative methods, *e.g.,* Jacobi, Gauss-Seidel and SOR methods [48], into *randomized iterative solvers*, *e.g.,* randomized Kaczmarz (RK) and randomized Gauss-Seidel/Coordinate Descent (RCD) methods. Contrary to direct solvers, which must perform their entire process before outputting the solution to (1.22), randomized iterative solvers are more flexible since they build sequentially iterates $(w^k)_k$ which are getting closer and closer to the solution (if it is unique). Such methods stop when a stopping criterion is met (typically if the residual is below a given threshold), and output the last iterate as the approximative solution to the system. Relatively new works on RK [73, 144, 102, 103] and RCD [87, 123, 93] methods proved their efficiency, both theoretically and numerically, against deterministic classical iterative or direct solvers. These results have confirmed the need of randomization to scale classical algorithm as nowadays dealing with massive amounts of data is a standard practice. For completeness we mention the classic Conjugate Gradient (CG) method [64, 141], dating back from the 1950's, which is still seen as a high standard difficult to beat since it typically converges in $n$ iterations (see Chapter 5 of [112]).

### 1.2.1 The sketch-and-project method

This section presents a method called *sketch-and-project* [59, 60, 53, 52] which has been initially designed to solve classic linear systems like (1.22). In what follows, we present this method and present how it was used and extended to other linear problems leveraging their inherent structure in Chapters 4 and 5. Sketch-and-project is a randomized iterative algorithm that unifies a large variety of methods including both RK [144, 102, 103] and RCD [59], and all their variants with importance or block sampling. This method goes beyond simple row or column sampling techniques and allows many more general sketching techniques as presented hereafter.

To solve approximately a linear system, we can use a *sketching matrix* $\mathbf{S} \in \mathbb{R}^{m \times \tau}$ to reduce the number of rows of the linear system (1.22) to $\tau$ rows as follows

$$\mathbf{S}^\top \mathbf{A} w = \mathbf{S}^\top b \ . \tag{1.23}$$

**One-shot sketching.** As in [37, 94], if the sketching matrix is correctly chosen and especially if the number of sampled rows $\tau$ is large enough, then the solution to the *sketched* linear system (1.23) is close to the solution $w^*$ of the initial system (1.22) with high probability. We refer to such technique as *one-shot* sketching, as opposed to *iterative sketching* methods such as sketch-and-project or Iteration Hessian Sketching (IHS) [117] methods, which involve a sketching process at each step. However, despite statistical guarantees given by the analysis of one-shot sketching it may be difficult to estimate the *sketch size* $\tau$ and if this quantity is too large, *i.e.,* close to the original system size $m$, computing the approximated solution will still be very slow due to the lack of dimension reduction. Moreover, the sketched system (1.23) can admit several solutions even if there is a unique one for the original one (1.22) and finally, with low probability, the solution to the sketched system can be far from $w^*$ which can be detrimental.

[11] See `https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.solve.html`.
[12] Scipy sparse formats are given in `https://docs.scipy.org/doc/scipy/reference/sparse.html`.

**Algorithm 5** SKETCH-AND-PROJECT

**Input:** distribution $\mathcal{D}$ over random $m \times \tau$ matrices

**Initialize:** $w^0 = 0_m$, $r^0 = \mathbf{A}w^0 - b = -b \in \mathbb{R}^m$

**for** $k = 0, 1, \ldots, K-1$ **do**

Sample an independent copy $\mathbf{S} \sim \mathcal{D}$

$w^{k+1} = w^k - \mathbf{A}^\top \mathbf{S} \left(\mathbf{S}^\top \mathbf{A}\mathbf{A}^\top \mathbf{S}\right)^\dagger \mathbf{S}^\top (\mathbf{A}w^k - b)$

**Output:** $w^K$

**Sketch-and-project.** In order to address these issues, iterative sketching methods exploite step by step new compressed information of the system and produce iterates $(w^k)_k$ that approach $w^*$. More precisely, in sketch-and-project methods we use an iterative projection scheme. At step $k$, a fresh sketching matrix $\mathbf{S}$ is sampled from $\mathcal{D}$, a predefined distribution over matrices of size $m \times \tau$, and the current iterate $w^k$ is projected onto the solution space of the sketched system $\mathbf{S}^\top \mathbf{A}w = \mathbf{S}^\top b$, that is

$$w^{k+1} = \underset{w \in \mathbb{R}^m}{\arg\min} \left\|w - w^k\right\|_2^2 \quad \text{subject to} \quad \mathbf{S}^\top \mathbf{A}w = \mathbf{S}^\top b \ . \tag{1.24}$$

In this section, we decided to use Euclidean projection, even if properly chosen weighted norms depending on features of $A$ can accelerate convergence rate [59]. Problem in (1.24) is known as the *sketching viewpoint* of sketch-and-project since $w^{k+1}$ is defined as the nearest point to $w^k$ which solves the sketch system. Its closed form solution is given by

$$w^{k+1} = w^k - \mathbf{A}^\top \mathbf{S} \left(\mathbf{S}^\top \mathbf{A}\mathbf{A}^\top \mathbf{S}\right)^\dagger \mathbf{S}^\top (\mathbf{A}w^k - b) \ , \tag{1.25}$$

where $^\dagger$ denotes the Moore–Penrose pseudoinverse.

A naive implementation of this method is available in Algorithm 5. Discussions about how to improve the implementation to accelerate the convergence are available in Chapter 4. In Algorithm 5, one can see that the only "parameter" is the distribution $\mathcal{D}$ from which the sketching matrice $(\mathbf{S})_k$ are sampled. It allows for a much wider compression techniques, not only row/column (block) and importance samplings, *e.g.,* fast Johnson-Lindenstrauss (JL) transforms [2] or any sketching technique mixing rows/columns information in a linear fashion such as *CountSketch* [27].

**Application to regression models.** As we have seen in Section 1.1, some ERM problems arising in machine learning can be easily written as the resolution of a linear system or approximated by a least-square formulation. This is for instance the case of Ordinary Least-Square (1.2) and ridge regression [132, 47] problems. As presented in (1.2), the solution of the OLS problem $w^*_{\text{OLS}}$ is given by its optimality conditions, that is solve in $w$ the following system

$$\mathbf{X}^\top \mathbf{X}w = \mathbf{X}^\top y \ . \tag{1.26}$$

In Chapter 4, we focus on applying the sketch-and-project method to the ridge regression problem which has similar optimality conditions but admits a unique solution. This led to a new method called *RidgeSketch*. This project was an opportunity to release an eponymous open source Python package available at:

`https://github.com/facebookresearch/RidgeSketch`.

This package is fully documented, tested, with extensive coverage and is provided with two tutorial notebooks to encourage its future use by the research community.

### 1.2.2 SGD interpretation of sketch-and-project

In our analysis of *RidgeSketch* in Chapter 4, we succeeded to provide convergence guarantees for a momentum version of (1.25):

$$w^{k+1} = w^k - \mathbf{A}^\top \mathbf{S} \left(\mathbf{S}^\top \mathbf{A}\mathbf{A}^\top \mathbf{S}\right)^\dagger \mathbf{S}^\top (\mathbf{A}w^k - b) + \beta_k (w^k - w^{k-1}) \ , \tag{1.27}$$

with iteration dependent momentum parameters $\beta_k \geq 0$. This idea was introduced in [125], where the authors developed *stochastic reformulations* of linear systems and derived that sketch-and-project method could be viewed as SGD applied to this new formulation of the objective. The same reasoning was also recently used to give an online SGD interpretation of the *Newton-Raphson* method [166] and to establish a new global convergence theory for the latter. It was also used to prove the sublinear convergence of the average iterate of sketch and project with constant momentum [91]. This new viewpoint allowed us to provide a new proof (and simpler than the one in [91] and dealing with convergence of the last iterate) of the sublinear convergence of the momentum version of *RidgeSketch* in Chapter 4 by exploiting recent results [135] on the convergence of the Stochastic Heavy Ball (SHB) method.

Let $\|x\|_{\mathbf{W}}$ be the weighted norm defined by a symmetric positive definite $\mathbf{W} \in \mathbb{R}^{m\times m}$ such that $\|x\|_{\mathbf{W}} = \sqrt{\langle \mathbf{W}x, x\rangle}$, for all $x \in \mathbb{R}^m$. Let $\mathbf{S} \sim \mathcal{D}$ and let $\mathbf{H_S} \overset{\text{def}}{=} \mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{A}^\top \mathbf{S})^\dagger \mathbf{S}^\top \in \mathbb{R}^{m\times m}$. To establish this SGD viewpoint, let us first define the stochastic function

$$f_{\mathbf{S}}(w) \overset{\text{def}}{=} \frac{1}{2} \|\mathbf{A}w - b\|^2_{\mathbf{H_S}} \ ,$$

which is a weighted version of the residual. Then the linear system can be reformulated as the following stochastic minimization problem

$$\min_{w\in\mathbb{R}^d} f(w) \overset{\text{def}}{=} \frac{1}{2} \|\mathbf{A}w - b\|^2_{\mathbb{E}[\mathbf{H_S}]} = \mathbb{E}\left[\frac{1}{2} \|\mathbf{A}w - b\|^2_{\mathbf{H_S}}\right] =: \mathbb{E}\left[f_{\mathbf{S}}(w)\right] \ , \tag{1.28}$$

Solving (1.22) is now equivalent to solving (1.28) so long as $\text{Range}\left(\mathbf{A}\right) \perp \text{Null}\left(\mathbb{E}\left[\mathbf{H_S}\right]\right)$, for which one can apply SGD

$$w^{k+1} = w^k - \gamma_k \nabla f_{\mathbf{S}}(w^k) \ , \tag{1.29}$$

where $\mathbf{S} \sim \mathcal{D}$ is sampled i.i.d at each iteration, $\gamma_k > 0$ is the step size and the gradient is given by

$$\nabla f_{\mathbf{S}}(w^k) = \mathbf{A}^\top \mathbf{H_S} \left(\mathbf{A}w^k - b\right) = \mathbf{A}^\top \mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{A}^\top \mathbf{S})^\dagger \mathbf{S}^\top \left(\mathbf{A}w^k - b\right) \ . \tag{1.30}$$

If one sets the step size $\gamma_k$ to 1, we precisely recover in (1.29) the sketch-and-project update (1.25).

In Chapter 4, we were able to propose a new convergence proof of sketch-and-project with iteration dependent momentum by leveraging this stochastic reformulation of the linear systems. Technically speaking, we exploit useful properties of the stochastic function $f_{\mathbf{S}}$ such as its convexity, the unbiasedness of its gradient, *e.g.,* $\mathbb{E}\left[\nabla f_{\mathbf{S}}(w^k)\right] = \nabla f(w^k)$ and how it relates to the residual $\|\mathbf{A}w - b\|_2^2$ (see [125]).

### 1.2.3 Randomization of ADI methods through sketch-and-project

As we have seen, sketch-and-project can be applied to problems which require solving a linear system as a stochastic iterative method and can even be interpreted as SGD. Moreover, it can also be used to randomize subroutines of algorithms requiring inverting linear systems, like quasi-Newton updates in [60]. In Chapter 5, we will focus on linear equations formed by the sum of two operators, *i.e.,* $\mathbf{\Sigma} \overset{\text{def}}{=} \mathbf{H} + \mathbf{V}$. Our work aimed at randomizing *Alternating-Direction Implicit (ADI)* methods solving such problems, hoping that it would lead to faster convergence in large scale settings or on problems with complex spectrum properties.

ADI methods are typically used to solve Sylvester and Lyapunov matrix equations, but to illustrate our point we focus here on a simpler case: the *Peaceman-Rachford (PR)* method [113]. The PR method was initially introduced in 1955 to solve the 2D heat flow equation, a parabolic Partial Differential Equation

**Algorithm 6** Peaceman-Rachford method

**Input:** shift parameters $(p_k)_k, (q_k)_k \in \mathbb{R}$

**Initialize:** $u^0 \in \mathbb{R}^m$

**for** $k = 0, 1, \ldots, K-1$ **do**

$u^{k+1/2} \leftarrow$ solution in $u$ of $(\mathbf{H} + p_k \mathbf{I}_m)u = s + (p_k \mathbf{I}_m - \mathbf{V})u^k$

$u^{k+1} \leftarrow$ solution in $u$ of $(\mathbf{V} + q_k \mathbf{I}_m)u = s + (q_k \mathbf{I}_m - \mathbf{H})u^{k+1/2}$

**Output:** $u^K$

(PDE), on a square:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{\partial T}{\partial t} \ , \tag{1.31}$$

with Dirichlet boundary conditions

$$\begin{aligned} T(\pm 1, y, t) &= 0, \quad \forall y \in [-1, 1], \ \forall t > 0 \\ T(x, \pm 1, t) &= 0, \quad \forall x \in [-1, 1], \ \forall t > 0 \ , \end{aligned}$$

and initial condition $T(x, y, 0) = 1$ for all $x, y \in [-1, 1]$, where $T$ is the temperature, $x$ and $y$ are the coordinates of the points of the square in $[-1, 1]$ and $t$ is the time.

More generally, it is applied to solve in $u \in \mathbb{R}^m$ linear systems of the form

$$(\mathbf{H} + \mathbf{V})u = s \ , \tag{1.32}$$

where $\mathbf{H}, \mathbf{V} \in \mathbb{R}^{m \times m}$ are symmetric matrices. In the example of (1.31), operators $\mathbf{H}$ (resp. $\mathbf{V}$) represent second order discrete differentiation operators with respect to the $x$ (resp. $y$) axis and $s$ equals the zero vector. The PR method is particularly useful for linear systems such that the full matrix $\mathbf{\Sigma} = \mathbf{H} + \mathbf{V}$ is too difficult to invert directly [89, 41]. It falls within the family of *operator splitting methods* [40] which rely on a fundamental assumption: even though $(\mathbf{\Sigma} + r\mathbf{I}_m)$ is not easily invertible, for $r \in \mathbb{R}$, let us suppose that $(\mathbf{H} + p\mathbf{I}_m)$ and $(\mathbf{V} + q\mathbf{I}_m)$ are relatively easy to invert, for some $p, q \in \mathbb{R}$. Then, by combining cleverly *forward* steps, *i.e.,* apply $(\mathbf{H} + p\mathbf{I}_m)$ or $(\mathbf{V} + q\mathbf{I}_m)$, and *backward* steps, *i.e.,* apply $(\mathbf{H} + p\mathbf{I}_m)^{-1}$ or $(\mathbf{V} + q\mathbf{I}_m)^{-1}$, one will approach a direct application of $(\mathbf{\Sigma} + r\mathbf{I}_m)^{-1}$. Hence $\mathbf{H}$ and $\mathbf{V}$ are not simply summed together and neither can we apply the *proximal point algorithm* [128].

Next we directly give PR updates but a detailed explanation of the underlying intuition based on variable splitting is provided in the introduction of Chapter 5. The PR method solves (1.32) by initializing a vector $u^0 \in \mathbb{R}^m$ (often full of zeros) and performs two half-steps as described in Algorithm 6. In Chapter 5, we try to replace these two linear system inversions in PR (and more generally in ADI) updates hoping that randomization will fasten the convergence in large scale settings or when spectral information is inaccessible.

## 1.3 Contributions

This thesis is divided into two parts: the first part studies stochastic variance reduced gradient methods widely used to solve the empirical risk minimization problem and provide theoretical results for practical settings (like mini-batching) or variants. The second part is focused on extensions of the *sketch-and-project* algorithm to tackle large scale problems giving both theoretical proofs of convergence and extensive numerical evidence of efficiency.

In this manuscript, emphasis is on the practical side of the proposed methods. Thus, open source codes in Julia or Python as well as extensive numerical experiments are provided to allow reproducibility. Note that each chapter can be read independently.

The structure of Part I is as follows:

- The goal of Chapter 2 is to provide theoretical basis capturing the benefice of mini-batching for SAGA. Such an analysis is possible through the use of a recently introduced tool called *expected smoothness* which measures the smoothness of the stochastic loss sampled with a mini-batch size $b$. This quantity allows us to develop convergence results interpolating between the regime of SAGA with single element sampling and full gradient descent. In particular, we provide approximations of this expected smoothness constant, which allows to set much larger step sizes and results in a much faster convergence rate. Furthermore, we show that the total complexity of the SAGA algorithm decreases linearly in the mini-batch size up to a pre-defined value: the optimal mini-batch size. The suite of Julia codes developed for this project is available at: `https://github.com/gowerrobert/StochOpt.jl`.
- In Chapter 3 we provide a more general analysis of SVRG than what had been previously done by using arbitrary sampling, which allows us to analyse virtually all forms of mini-batching through a single theorem. Moreover, our analysis is focused on more practical variants of SVRG including a new variant of the loopless SVRG [68, 81, 83] and a variant of k-SVRG [121] where $m = n$ and where $n$ is the number of data points. Since our setup and analysis reflect what is done in practice, we are able to set the parameters such as the mini-batch size and step size using our theory in such a way that produces a more efficient algorithm in practice. Numerical experiments have also been performed with the above-mentioned Julia code suite `StochOpt.jl`.

Part II is divided as follows:

- In Chapter 4 we propose new variants of the sketch-and-project method for solving large scale ridge regression problems. Firstly, we propose a new momentum alternative and provide a theorem showing it can speed up the convergence of sketch-and-project, through a fast *sublinear* convergence rate. Secondly, we consider combining the sketch-and-project method with new modern sketching methods such as the Count sketch, SubCount sketch (a new method we propose), and subsampled Hadamard transforms. We show experimentally that when combined with the sketch-and-project method, the (Sub)Count sketch is very effective on sparse data and the standard Subsample sketch is effective on dense data. It is even competitive when compared to the Conjugate Gradient method. Corresponding open source software package can be found at: `https://github.com/facebookresearch/RidgeSketch`.
- In Chapter 5 we develop sketch-and-project versions of the Alternating-Direction Implicit (ADI) method. A particular example of these methods is the *Peaceman-Rachford (PR) method*, designed to solve squared linear systems of the form $(\mathbf{H} + \mathbf{V})u = s$ where we can only access solvers of the shifted systems $(\mathbf{H} + p\mathbf{I}_m)u = s$ and $(\mathbf{V} + q\mathbf{I}_m)u = s$, with $p, q \in \mathbb{R}$. These equations arise in numerical analysis, for instance when solving PDEs, but can easily be limited by the cost of system inversions for high dimensional problems. They are also commonly employed to solve Sylvester matrix equations of the form $\mathbf{AX} - \mathbf{XB} = \mathbf{F}$. Thus, we propose the *Sketched PR (SPR)* and *Sketched ADI (SADI)* methods to try to scale up these algorithms through randomization of both half-steps using the sketch-and-project. SPR succeeds to outperform classic PR with constant shifts on large scale problems, but SADI does not yet lead to convergence improvement (in time nor in epoch) over ADI. Corresponding codes and notebooks are available at: `https://github.com/ngazagna/sadi`.

## 1.4 Publications

This thesis is based on the following papers and ongoing work:

- **N. Gazagnadou**, R. Gower, and J. Salmon. "Optimal mini-batch and step sizes for SAGA". ICML, 2019
- O. Sebbouh, **N. Gazagnadou**, S. Jelassi, F. Bach, and R. Gower. "Towards closing the gap between the theory and practice of SVRG". NeurIPS, 2019

- **N. Gazagnadou**, M. Ibrahim, and R. M. Gower. "RidgeSketch: A Fast sketching based solver for large scale ridge regression". arXiv preprint arXiv:2105.05565, 2021 (Submitted to SIMAX)
- **N. Gazagnadou**. "Sketched ADI: a randomized iterative method for solving large scale Sylvester matrix equations". Incoming preprint

# Part I

# Extended analysis of variance reduced gradient methods through expected smoothness and residual

CHAPTER 2

# Optimal mini-batch and step sizes for SAGA

کار نیکو کردن از پر کردن است

Persian proverb

Recently it has been shown that the step sizes of a family of variance reduced gradient methods called the JacSketch methods [55] depend on the expected smoothness constant. In particular, if this expected smoothness constant could be calculated a priori, then one could safely set much larger step sizes which would result in a much faster convergence rate. We fill in this gap, and provide simple closed form expressions for the expected smoothness constant and careful numerical experiments verifying these bounds. Using these bounds, and since the SAGA algorithm is part of this JacSketch family, we suggest a new standard practice for setting the step and mini-batch sizes for SAGA [33] that are competitive with a numerical grid search. Furthermore, we can now show that the total complexity of the SAGA algorithm decreases linearly in the mini-batch size up to a pre-defined value: the optimal mini-batch size. This is a rare result in the stochastic variance reduced literature, only previously shown for the Katyusha algorithm [3]. Finally we conjecture that this is the case for many other stochastic variance reduced methods and that our bounds and analysis of the expected smoothness constant is key to extending these results.

## 2.1 Introduction

Consider the empirical risk minimization (ERM) problem:

$$w^* \in \arg\min_{w\in\mathbb{R}^d} \left( f(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w) \right) , \tag{2.1}$$

where each $f_i$ is $L_i$-smooth and $f$ is $\mu$-strongly convex. Each $f_i$ represents a regularized loss over a sampled data point. Solving the ERM problem is often time consuming for large number of samples $n$, so much so that algorithms scanning through all the data points at each iteration are not competitive. Gradient Descent (GD) falls into this category, and in practice its stochastic version is preferred.

Stochastic gradient descent (SGD), on the other hand, allows to solve the ERM incrementally by computing at each iteration an unbiased estimate of the full gradient, $\nabla f_i(w^k)$ for $i$ randomly sampled in $[n] \overset{\text{def}}{=} \{1, 2, \ldots, n\}$ [126]. On the downside, for SGD to converge one needs to tune a sequence of asymptotically vanishing step sizes, a cumbersome and time-consuming task for the user. Recent works have taken advantage of the sum structure in (2.1) to design stochastic variance reduced gradient algorithms [129, 71, 139, 33]. In the strongly convex setting, these methods lead to fast linear convergence

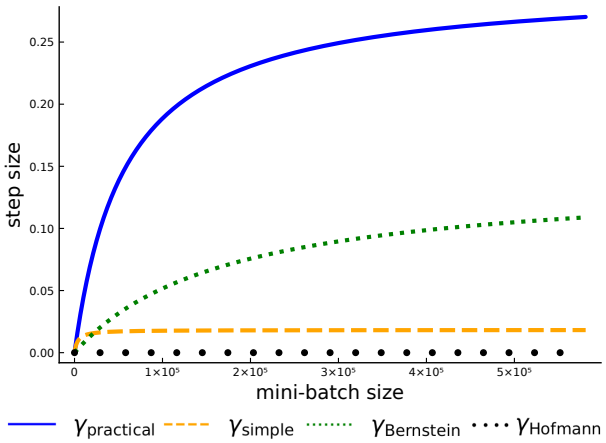



**Figure 2.1:** *Step size as a function of the mini-batch size for a regularized ($\lambda = 10^{-3}$) logistic regression problem applied to the feature-scaled covtype.binary dataset from LIBSVM.*

instead of the slow $\mathcal{O}(1/t)$ rate of SGD. Moreover, they only require a constant step size, informed by theory, instead of sequence of decreasing step sizes.

In practice, most variance reduced methods rely on a mini-batching strategy for better performance. Yet most convergence analysis (with the Katyusha algorithm of Allen-Zhu [3] being an exception) indicates that a mini-batch size of $b = 1$ gives the best overall complexity, disagreeing with practical findings, where larger mini-batch often gives better results. Here, we show both theoretically and numerically that $b = 1$ is not the optimal mini-batch size for the SAGA algorithm [33].

Our analysis leverages recent results in [55], where the authors prove that the iteration complexity and the step size of SAGA, and a larger family of methods called the JacSketch methods, depend on an expected smoothness constant. This constant governs the trade-off between the increased cost of an iteration as the mini-batch size is increased, and the decreased total complexity. Thus if this expected smoothness constant could be calculated a priori, then we could set the optimal mini-batch size and step size. We provide simple formulas for computing the expected smoothness constant when sampling mini-batches without replacement, and use them to calculate optimal mini-batches and significantly larger step sizes for SAGA.

In particular, we provide three bounds on the expected smoothness constant, each resulting in a particular step size formula. We first derive the *practical bound* which was originally conjectured in [45] and later proven to hold in [58]. Then, we present the *simple bound* and finally we develop a matrix concentration inequality to obtain the refined *Bernstein bound*. For illustration, we plot in Figure 2.1 the evolution of each resulting step size as the mini-batch size grows on a classification problem (Section 2.5 has more details on our experimental settings). Furthermore, our bounds provide new insight into the *total complexity*, denoted $K_{\text{total}}$ hereafter, of SAGA. For example, when using our *simple bound* we show for regularized Generalized Linear Models (GLM), with $\lambda > 0$ as in (2.11), that $K_{\text{total}}$ is piecewise linear in the mini-batch size $b$:

$$K_{\text{total}}(b) = \max\left\{n\frac{b-1}{n-1}\frac{4\bar{L}}{\mu} + \frac{n-b}{n-1}\frac{4L_{\max}}{\mu} + \frac{4b\lambda}{\mu}, n + \frac{n-b}{n-1}\frac{4(L_{\max}+\lambda)}{\mu}\right\}\log\left(\frac{1}{\epsilon}\right) \quad ,$$

with $L_{\max} \overset{\text{def}}{=} \max_{i\in[n]} L_i$, $\bar{L} \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} L_i$ and $\epsilon > 0$ is the desired precision. This complexity bound, and others presented in section 2.3.3 show that SAGA enjoys a linear speedup as we increase the mini-batch size until an optimal one (as illustrated in Figure 2.2). After this point, the total complexity increases. We use this observation to develop optimal and practical mini-batch sizes and step sizes.

The rest of the chapter is structured as follows. In Section 2.2 we first introduce variance reduction techniques after presenting our main assumption, the expected smoothness assumption. We highlight how this assumption is necessary to capture the improvement in iteration complexity, and conclude the

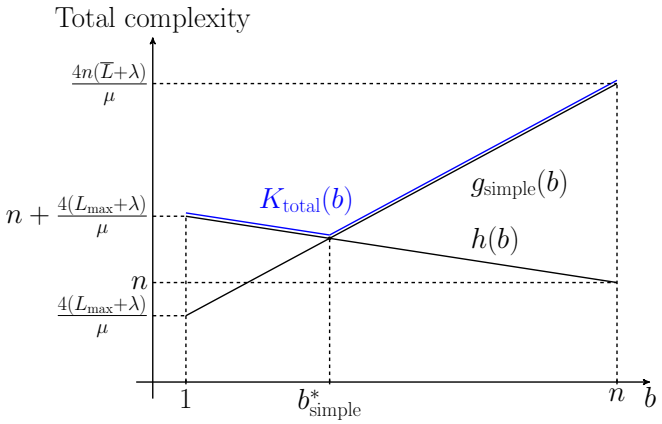



**Figure 2.2:** *Optimal mini-batch size $b^*_{simple}$ for the* simple bound*, where* $K_{\text{total}}(b) = \max\{g_{\text{simple}}(b), h(b)\}$.

section by showing that to calculate the expected smoothness constant we need evaluate an intractable expectation. Which brings us to Section 2.3 where we directly address this issue and provide several tractable upper-bounds of the expected smoothness constant. We then calculate optimal mini-batch sizes and step sizes by using our new bounds. Finally, we give numerical experiments in Section 2.5 that verify our theory on artificial and real datasets. We also show how these new settings for the mini-batch size and step size lead to practical performance gains.

## 2.2 Background

### 2.2.1 Controlled stochastic reformulation and JacSketch

We can introduce variance reduced versions of SGD in a principled manner by using a *sampling vector*.

**Definition 2.1.** *We say that a random vector $v \in \mathbb{R}^n$ with distribution $\mathcal{D}$ is a sampling vector if*

$$\mathbb{E}_{\mathcal{D}}\left[v_i\right] = 1 \ , \quad \text{for all } i \in [n] \ .$$

With a sampling vector we can rewrite (2.1) through the following stochastic reformulation

$$w^* = \arg\min_{w\in\mathbb{R}^d} \mathbb{E}_{\mathcal{D}}\left[f_v(w) \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^n f_i(w)\cdot v_i\right], \tag{2.2}$$

where $f_v(w)$ is called a *subsampled function*. The stochastic Problem (2.2) and our original Problem (2.1) are equivalent :

$$\mathbb{E}_{\mathcal{D}}\left[f_v(w)\right] = \frac{1}{n}\sum_{i=1}^n f_i(w)\cdot\mathbb{E}_{\mathcal{D}}\left[v_i\right] \overset{\text{Definition 2.1}}{=} \frac{1}{n}\sum_{i=1}^n f_i(w) \ .$$

Consequently the gradient $\nabla f_v(w)$ is an unbiased estimate of $\nabla f(w)$ and we could use the SGD method to solve (2.2). To tackle the variance of these stochastic gradients we can further modify (2.2) by introducing *control variates* which leads to the following *controlled stochastic reformulation*:

$$w^* \in \arg\min_{w\in\mathbb{R}^d} \mathbb{E}_{\mathcal{D}}\left[f_v(w) - z_v(w) + \mathbb{E}_{\mathcal{D}}\left[z_v(w)\right]\right] \ , \tag{2.3}$$

where $z_v(w) \in \mathbb{R}$ are the control variates. Clearly (2.3) is also equivalent to (2.1) since $-z_v(w)+\mathbb{E}_{\mathcal{D}}\left[z_v(w)\right]$ has zero expectation. Thus, we can solve (2.3) using an SGD algorithm where the stochastic gradients

are given by

$$g_v(w) \overset{\text{def}}{=} \nabla f_v(w) - \nabla z_v(w) + \mathbb{E}_{\mathcal{D}}\left[\nabla z_v(w)\right] \ . \tag{2.4}$$

That is, starting from a vector $w^0$, given a positive step size $\gamma$, we can iterate the steps

$$w^{k+1} = w^k - \gamma g_{v^k}(w^k) \ , \tag{2.5}$$

where $v^k \sim \mathcal{D}$ are i.i.d. samples at each iteration.

The JacSketch algorithm introduced by Gower, Richtárik, and Bach [55] fits this format (2.5) and uses a linear control $z_v(w) = \frac{1}{n}\left\langle \mathbf{J}^\top w, v\right\rangle$, where $\mathbf{J}$ is a $d \times n$ matrix of parameters. This matrix is updated at each iteration so as to decrease the variance of the resulting stochastic gradients. Carefully updating the covariates through $\mathbf{J}$ results in a method that has stochastic gradients with decreasing variance, *i.e.,* $\lim_{w^k \to w^*} \mathbb{E}\left[\|g_{v^k}(w^k) - \nabla f(w^k)\|_2^2\right] = 0$, which is why JacSketch is a stochastic variance reduced algorithm. This is also why the user can set a single constant step size a priori instead of tuning a sequence of decreasing ones. The SAGA algorithm, and all of its mini-batching variants, are instances of the JacSketch method.

### 2.2.2 The expected smoothness constant

In order to analyse stochastic variance reduced methods, some form of smoothness assumption needs to be made. The most common assumption is

$$\|\nabla f_i(w) - \nabla f_i(y)\| \leq L_{\max} \|w - y\| \,, \tag{2.6}$$

for each $i \in [n]$. That is each $f_i$ is uniformly smooth with smoothness constant $L_{\max}$, as is assumed in [33, 68, 121] for variants of SAGA[1]. In the analyses of these papers it was shown that the iteration complexity of SAGA is proportional to $L_{\max}$, and the step size is inversely proportional to $L_{\max}$.

But as was shown in [55], we can set a much larger step size by making use of the smoothness of the subsampled functions $f_v$. For this Gower, Richtárik, and Bach [55] introduced the notion of expected smoothness, which we extend here to all sampling vectors and control variates.

**Definition 2.2** (Expected smoothness constant)**.** *Consider a sampling vector $v$ with distribution $\mathcal{D}$. We say that the expected smoothness assumption holds with constant $\mathcal{L}$ if for every $w \in \mathbb{R}^d$ we have that*

$$\mathbb{E}_{\mathbf{S}\sim\mathcal{D}}\left[\|\nabla f_v(w) - \nabla f_v(w^*)\|_2^2\right] \leq 2\mathcal{L}(f(w) - f(w^*)) \ . \tag{2.7}$$

**Remark 2.3.** *Note that we refer to any positive constant $\mathcal{L}$ that satisfies* (2.7) *as an expected smoothness constant. Indeed $\mathcal{L} \to \infty$ is a valid constant in the extended reals, but as we will see, the smaller $\mathcal{L}$, the better for our complexity results.*

Gower, Richtárik, and Bach [55] show that the expected smoothness constant plays the same role that $L_{\max}$ does in the previously existing analysis of SAGA, namely that the step size is inversely proportional to $\mathcal{L}$ and the iteration complexity is proportional to $\mathcal{L}$ (see details in Theorem 2.10). Furthermore, by assuming that $f$ is $L$–smooth, the expected smoothness constant is bounded

$$L \quad \leq \quad \mathcal{L} \quad \leq \quad L_{\max} \ , \tag{2.8}$$

as was proven in Theorem 4.17 in [55]. Also, the bounds $L_{\max}$ and $L$ are attained when using a uniform single element sampling and a full batch, respectively. And as we will show, the constants $L_{\max}$ and $L$ can be orders of magnitude apart on large dimensional problems. Thus we could set much larger step sizes for larger mini-batch sizes if we could calculate $\mathcal{L}$. Though calculating $\mathcal{L}$ is not easy, as we see in the next lemma.

[1]The same assumption is made in proofs of SVRG [71], S2GD [80] and the SARAH algorithm [109].

**Lemma 2.4.** *Let $v$ be an unbiased sampling vector. Suppose that $f_v(w) = \frac{1}{n}\sum_{i=1}^n f_i(w)v_i$ is $L_v$-smooth and each $f_i$ is convex for $i = 1, \dots, n$. It follows that the expected smoothness constant holds with $\mathcal{L} = \max_{i\in[n]} \mathbb{E}[L_v v_i]$.*

*Proof of Lemma 2.4.* Since the $f_i$'s are convex, each realization of $f_v$ is convex, and it follows from equation 2.1.7 in [106] that

$$\|\nabla f_v(x) - \nabla f_v(y)\|_2^2 \quad \leq \quad 2L_v \left(f_v(x) - f_v(y) - \langle \nabla f_v(y), x - y\rangle\right). \tag{2.9}$$

Taking expectation over the sampling gives

$$\begin{aligned}
\mathbb{E}[\|\nabla f_v(x) - \nabla f_v(x^*)\|_2^2] &\leq 2\mathbb{E}\left[L_v \left(f_v(x) - f_v(x^*) - \langle \nabla f_v(x^*), x - x^*\rangle\right)\right] \\
&\overset{(2.9)}{=} \frac{2}{n}\mathbb{E}\left[\sum_{i=1}^n L_v v_i \left(f_i(x) - f_i(x^*) - \langle \nabla f_i(x^*), x - x^*\rangle\right)\right] \\
&= \frac{2}{n}\sum_{i=1}^n \mathbb{E}[L_v v_i]\left(f_i(x) - f_i(y) - \langle \nabla f_i(x^*), x - x^*\rangle\right) \\
&\leq 2\max_{i=1,\dots,n} \mathbb{E}[L_v v_i]\left(f(x) - f(x^*) - \langle \nabla f(x^*), x - x^*\rangle\right) \\
&= 2\max_{i=1,\dots,n} \mathbb{E}[L_v v_i]\left(f(x) - f(x^*)\right).
\end{aligned}$$

where in the last equality the full gradient vanishes because it is computed at optimality. The result now follows by comparing the above with the definition of expected smoothness in (2.7). □

If the sampling has a very large combinatorial number of possible realizations – for instance sampling mini-batches without replacement – then this expectation becomes intractable to calculate. This observation motivates the development of functional upper-bounds of the expected smoothness constant that can be efficiently evaluated.

### 2.2.3 Mini-batch without replacement: $b$–nice sampling

Now we will choose a distribution of the sampling vector $v$ based on a mini-batch sampling without replacement. We denote a mini-batch as $B \subseteq [n]$ and its size as $b = |B|$.

**Definition 2.5** (*b-nice sampling*)**.** *$S$ is a $b$-nice sampling if $S$ is a set valued map with a probability distribution given by*

$$\mathbb{P}[S = B] = \frac{1}{\binom{n}{b}}, \quad \forall B \subseteq [n] \quad : \quad |B| = b \ .$$

We can construct a sampling vector based on a $b$-nice sampling by setting $v = \frac{n}{b}\sum_{i\in S} e_i$, where $e_1, \dots, e_n$ is the canonical basis of $\mathbb{R}^n$. Indeed, $v$ is a sampling vector according to Definition 2.1 since for every $i \in [n]$ we have

$$v_i = \left(\frac{n}{b}\sum_{j\in S} e_j\right)_i = \frac{n}{b}\mathbb{1}_S(i) \ , \tag{2.10}$$

where $\mathbb{1}_S$ denotes the indicator function of the random set $S$. Now taking expectation in (2.10) gives

$$\mathbb{E}[v_i] = \frac{n}{b}\frac{1}{\binom{n}{b}}\sum_{B\subseteq[n]\,:\,|B|=b} \mathbb{1}_B(i) = \frac{n}{b\binom{n}{b}}\binom{n-1}{b-1} = 1,$$

using $|\{B \subseteq [n] : |B| = b \wedge i \in B\}| = \binom{n-1}{b-1}$.

Here we are interested in the mini-batch SAGA algorithm with $b$-nice sampling, which we refer to as the $b$-nice SAGA. In particular, $b$-nice SAGA is the result of using $b$-nice sampling, together with a linear model for the control variate $z_v(w)$. Different choices of the control variate $z_v(w)$ also recover

| Parameters | $v$ | $\nabla z_v(w)$ |
|---|---|---|
| GD | $e = e_1 + \cdots + e_n$ | $\nabla f_i(w)$ |
| SGD | $ne_i,\ i \sim \frac{1}{n}$ | $0$ |
| SAGA | $ne_i,\ i \sim \frac{1}{n}$ | $\mathbf{J}_{:i}$ |
| $b$-nice SAGA | $(n/b)\sum_{i\in S} e_i$ | $(1/b)\sum_{i\in S}\mathbf{J}_{:i}$ |

**Table 2.1:** *Algorithms covered by JacSketch and corresponding sampling vector $v$ and control variates $z_v$.*

popular algorithms such as gradient descent, SGD or the standard SAGA method (see Table 2.1 for some examples).

A naive implementation of $b$-nice SAGA based on the JacSketch algorithm is given in Algorithm 7[2].

**Algorithm 7** JacSketch version of $b$-nice SAGA

**Input:** mini-batch size $b$, step size $\gamma > 0$
**Initialize:** $w^0 \in \mathbb{R}^d$, $\mathbf{J}^0 \in \mathbb{R}^{d\times n}$
**for** $k = 0, 1, \ldots, K-1$ **do**
  Sample a fresh batch $B \subseteq [n]$ s.t. $|B| = b$
  $g^k = \frac{1}{n}\mathbf{J}^k e + \frac{1}{b}\sum_{i\in B}(\nabla f_i(w^k) - \mathbf{J}^k_{:i})$ ▷ update the gradient estimate
  $\mathbf{J}^{k+1}_{:i} = \begin{cases}\mathbf{J}^k_{:i}, & \text{if } i \notin B\\ \nabla f_i(w^k), & \text{if } i \in B.\end{cases}$ ▷ update Jacobian estimate
  $w^{k+1} = w^k - \gamma g^k$ ▷ take a step
**Output:** $w^K$ ▷ return weights vector

## 2.3 Upper bounds on the expected smoothness

To determine an optimal mini-batch size $b^*$ for $b$-nice SAGA, we first state our assumptions and provide bounds of the smoothness of the subsampled function. We then define $b^*$ as the mini-batch size that minimizes the total complexity of the considered algorithm, *i.e.,* the total number of stochastic gradients computed. Finally we provide upper-bounds on the expected smoothness constant $\mathcal{L}$, through which we can deduce optimal mini-batch sizes. Many proofs are deferred to the supplementary material.

### 2.3.1 Assumptions and notation

We consider that the objective function is a GLM with quadratic regularization controlled by a parameter $\lambda > 0$:

$$w^* \in \arg\min_{w\in\mathbb{R}^d} f(w) = \frac{1}{n}\sum_{i=1}^{n}\phi_i(x_i^\top w) + \frac{\lambda}{2}\|w\|_2^2 \ , \tag{2.11}$$

with $\|\cdot\|_2$ is the Euclidean norm, $\phi_1, \ldots, \phi_n$ are convex functions and $a_1, \ldots, a_n$ a sequence of observations in $\mathbb{R}^d$. This framework covers regularized logistic regression by setting $\phi_i(z) = \log(1+\exp(-y_i z))$ for some binary labels $y_i, \ldots, y_n$ in $\{\pm 1\}$, ridge regression if $\phi_i(z) = (z - y_i)^2/2$ for real observations $y_i, \ldots, y_n$, and conditional random fields for when the $y_i$'s are structured outputs.

We assume that the second derivative of each $\phi_i$ is uniformly bounded, which holds for our aforementioned examples.

**Assumption 2.6** (Bounded second derivatives)**.** *There exists $U \geq 0$ such that $\phi_i''(x) \leq U, \forall x \in \mathbb{R}, \forall i \in [n]$.*

[2]We also provide a more efficient implementation that we used for our experiments see Algorithm 8.

For a batch $B \subseteq [n]$, we rewrite the subsampled function as

$$f_B(w) \overset{\text{def}}{=} \frac{1}{|B|} \sum_{i \in B} \phi_i(x_i^\top w) + \frac{\lambda}{2} \|w\|_2^2 \ ,$$

and its second derivative is thus given by

$$\nabla^2 f_B(w) = \frac{1}{|B|} \sum_{i \in B} \phi_i''(x_i^\top w) x_i x_i^\top + \lambda \mathbf{I}_d \ , \tag{2.12}$$

where $\mathbf{I}_d$ denotes the identity matrix of size $d$.

For a symmetric matrix $\mathbf{M}$, we write $\lambda_{\max}(\mathbf{M})$ (resp. $\lambda_{\min}(\mathbf{M})$) for its largest (resp. smallest) eigenvalue. Assumption 2.6 directly implies the following.

**Lemma 2.7** (Subsample smoothness constant)**.** *Let $B \subset [n]$ of cardinal $b$, and let $\mathbf{X}_B \overset{def}{=} [x_i]_{i \in B}^\top \in \mathbb{R}^{b \times d}$ denote the row concatenation of the vectors $x_i$ with $i \in B$. The smoothness constant of the subsampled loss function $\frac{1}{|B|} \sum_{i \in B} \phi_i(x_i^\top w)$ is given by*

$$L_B \overset{def}{=} \frac{U}{|B|} \lambda_{\max} \left( \sum_{i \in B} x_i x_i^\top \right) = \frac{U}{|B|} \lambda_{\max} \left( \mathbf{X}_B^\top \mathbf{X}_B \right) \ . \tag{2.13}$$

*Proof.* The proof follows from Assumption 2.6 as

$$\begin{aligned} \frac{1}{|B|} \sum_{i \in B} \nabla^2 \phi_i(x_i^\top w) &= \frac{1}{|B|} \sum_{i \in B} \phi_i''(x_i^\top w) x_i x_i^\top \\ &\preceq \frac{U}{|B|} \mathbf{X}_B^\top \mathbf{X}_B \ . \end{aligned}$$

□

Combined with (2.12), we get that $f_B$ is $(L_B + \lambda)$-smooth.

Another key quantity in our analysis is the strong convexity parameter.

**Definition 2.8.** *The strong convexity parameter is given by*

$$\mu \overset{def}{=} \min_{w \in \mathbb{R}^d} \lambda_{\min} \left( \nabla^2 f(w) \right).$$

Since we have an explicit regularization term with $\lambda > 0$, $f$ is strongly convex and $\mu \geq \lambda > 0$.

We additionally define $L_i$, resp. $L$, as the smoothness constant of the individual function $\phi_i(x_i^\top w)$, resp. the whole function $\frac{1}{n} \sum_{i=1}^n \phi_i(x_i^\top w)$. We also recall the definitions of the maximum of the individual smoothness constants by $L_{\max} \overset{\text{def}}{=} \max_{i \in [n]} L_i$ and their average by $\bar{L} \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n L_i$.

The complexity results of Gower, Richtárik, and Bach [55] depends on all these smoothness constants. Here are some basic inequalities giving an idea of the order of those constants.

**Lemma 2.9.** *Let $\emptyset \neq B \subseteq [n] = \{1, \ldots, n\}$ a mini-batch set drawn randomly without replacement. The following inequalities hold*

*(i)*

$$L_i \leq L_{\max} \quad \forall i = 1, \ldots, n \ . \tag{2.14}$$

*(ii)*

$$L_B \leq \frac{1}{|B|} \sum_{i \in B} L_i \quad \forall i = 1, \ldots, n \ . \tag{2.15}$$

*(iii)*

$$L \overset{(a)}{\leq} \bar{L} \overset{(b)}{\leq} L_{\max} \overset{(c)}{\leq} nL \overset{(d)}{\leq} n\bar{L} \ . \tag{2.16}$$

*Proof.* (i) One directly gets that $L_i \leq \max_{j=1,\dots,n} L_j = L_{\max}$.

(ii) This inequality states that the smoothness constant $L_B$ of the averaged function $f_B$ is upper bounded by the average of the corresponding smoothness constants $L_i$, over the batch $B$. The proof consists in $|B|$ repetitive calls of Lemma 2.23.

(iii) ($a$) Direct implication of (ii) for $B = [n]$.

($b$) Direct calculation

$$\bar{L} = \frac{1}{n}\sum_{i=1}^{n} L_i \leq \frac{1}{n}\sum_{i=1}^{n} L_{\max} = L_{\max} \ .$$

($c$) Let us first recall the matrix formulation of our smoothness constants:

$$L = \frac{U}{n}\lambda_{\max}(\mathbf{X}^\top\mathbf{X}) = \frac{U}{n}\lambda_{\max}(\mathbf{X}\mathbf{X}^\top)$$

where let $\mathbf{X} \overset{\text{def}}{=} [x_1, \dots, x_n]^\top \in \mathbb{R}^{n\times d}$, and

$$L_{\max} = U \max_{i=1,\dots,n} e_i^\top \mathbf{X}\mathbf{X}^\top e_i \ .$$

Using the min-max theorem, we have that

$$\lambda_{\max}(\mathbf{X}\mathbf{X}^\top) = \max_{x\neq 0} \frac{x^\top \mathbf{X}\mathbf{X}^\top x}{\|x\|_2^2} \geq \max_{i=1,\dots,n} e_i^\top \mathbf{X}\mathbf{X}^\top e_i \ .$$

Dividing the above by $n$ on both sides gives

$$L \geq \frac{L_{\max}}{n} \ .$$

($d$) Direct consequence of ($a$).

□

### 2.3.2 Path to the optimal mini-batch size

Our starting point is the following theorem taken from combining Theorem 3.6 and Eq. (103) in [55][3].

**Theorem 2.10.** *Consider the iterates $w^k$ of Algorithm 7. Let the step size be given by*

$$\gamma = \frac{1}{4}\frac{1}{\max\left\{\mathcal{L}+\lambda, \frac{1}{b}\frac{n-b}{n-1}(L_{\max}+\lambda) + \frac{\mu}{4}\frac{n}{b}\right\}}. \tag{2.17}$$

*Given an $\epsilon > 0$, if $k \geq K_{\text{iter}}(b)$ where*

$$K_{\text{iter}}(b) \overset{def}{=} \left\{\frac{4(\mathcal{L}+\lambda)}{\mu}, \frac{n}{b} + \frac{n-b}{n-1}\frac{4(L_{\max}+\lambda)}{b\mu}\right\}\log\left(\frac{1}{\epsilon}\right), \tag{2.18}$$

[3]Note that $\lambda$ has been added to every smoothness constant since the analysis in Gower, Richtárik, and Bach [55] depends on the $(L+\lambda)$-smoothness of $f$ and the $(L_B+\lambda)$-smoothness of the subsampled functions $f_B$.

*then* $\mathbb{E}\left[\left\|w^k - w^*\right\|^2\right] \leq \epsilon\, C$, *where* $C > 0$ *is a constant* [4].

Through Theorem 2.10 we can now explicitly see how the expected smoothness constant $\mathcal{L}$ controls both the step size and the resulting iteration complexity. This is why we need bounds on $\mathcal{L}$ so that we can set the step size. In particular, we will show that the expected smoothness constant is a function of the mini-batch size $b$. Consequently so is the step size, the iteration complexity and the *total complexity*. We denote $K_{\text{total}}$ the total complexity defined as the number of stochastic gradients computed, hence with (2.18),

$$K_{\text{total}}(b) = bK_{\text{iter}}(b) = \max\left\{\frac{4b(\mathcal{L}+\lambda)}{\mu}, n + \frac{n-b}{n-1}\frac{4(L_{\max}+\lambda)}{\mu}\right\}\log\left(\frac{1}{\epsilon}\right). \tag{2.19}$$

Once we have determined $\mathcal{L}$ as a function of $b$, we will calculate the mini-batch size $b^*$ that optimizes the total complexity $b^* \in \arg\min_{b\in[n]} K_{\text{total}}(b)$.

As we have shown in Lemma 2.4, computing a precise bound on $\mathcal{L}$ can be computationally intractable. This is why we focus on finding upper bounds on $\mathcal{L}$ that can be computed, but also tight enough to be useful. To verify that our bounds are sufficiently tight, we will always have in mind the bounds $L \leq \mathcal{L} \leq L_{\max}$ given in (2.8). In particular, after expressing our bounds of $\mathcal{L} = \mathcal{L}(b)$ as a function of $b$,we would like the bounds (2.8) to be attained for $\mathcal{L}(1) = L_{\max}$ and $\mathcal{L}(n) = L$.

### 2.3.3 Estimating the expected smoothness

#### 2.3.3.1 Expected smoothness for $b$-nice sampling

All bounds we develop on $\mathcal{L}$ are based on the following lemma, which is a specialization of (2.4) for $b$-nice sampling.

**Proposition 2.11** (Expected smoothness constant)**.** *For the $b$-nice sampling, with $b \in [n]$, the expected smoothness constant is given by*

$$\mathcal{L} = \frac{1}{\binom{n-1}{b-1}}\max_{i=1,\dots,n}\left\{\sum_{B\subseteq[n]:|B|=b\wedge i\in B} L_B\right\}. \tag{2.20}$$

*Proof.* Let $S$ the $b$-nice sampling as defined in Definition 2.5 and let $v = \frac{n}{b}\sum_{j\in S} e_j$ be its corresponding sampling vector. Note that

$$f_v(w) = \frac{1}{n}\sum_{i\in[n]} f_i(w)v_i = \frac{1}{b}\sum_{i\in S} f_i(w) = f_S(w).$$

Finally from Lemma 2.4, we have that:

$$\begin{aligned}
\mathcal{L} = \mathbb{E}\left[L_v v_i\right] &\overset{(2.10)}{=} \mathrm{E}\left[L_S \frac{n}{b}\mathbb{1}_{\{i\in S\}}\right] \\
&= \frac{1}{\binom{n}{b}}\frac{n}{b}\sum_{B\subseteq[n]\,:\,|B|=b} L_B\mathbb{1}_{\{i\in B\}} \\
&= \frac{1}{\binom{n}{b}}\frac{n}{b}\sum_{\substack{B\subseteq[n]:\\ |B|=b\wedge i\in B}} L_B = \frac{1}{\binom{n-1}{b-1}}\sum_{\substack{B\subseteq[n]:\\ |B|=b\wedge i\in B}} L_B.
\end{aligned}$$

[4] Specifically, let $J^0 \in \mathbb{R}^{d\times n}$ be the initiated Jacobian of the $b$-nice SAGA Algorithm 7. Then this constant is

$$C \overset{\text{def}}{=} \left\|w^0 - w^*\right\|^2 + \frac{\gamma}{2L_{\max}}\sum_{i\in[n]}\left\|J^0_{:i} - \nabla f(w^*)\right\|^2.$$

Taking the maximum over all $i \in [n]$ gives the result. □

#### 2.3.3.2 Practical bound

Next we present the *practical bound* of $\mathcal{L}$ that is tight for both small and large mini-batch sizes. History of this bound is a bit special: at first, we conjectured in [45] that $\mathcal{L}_{\text{practical}}$ was a good approximation of the expected smoothness for nice sampling. Indeed, it was blatant when looking at the experiments, see Figure 2.3 and more examples in Section 2.8.1, that this practical was perfectly matching the expected smoothness in small dimensions. Yet, we could not prove that it was a proper upped bound of $\mathcal{L}$. One had to wait until the publication of [58] to get confirmation $\mathcal{L}_{\text{practical}}$ is a proper upper bound of $\mathcal{L}$. For the sake of completeness, we give its proof hereafter.

**Lemma 2.12** (Practical bound)**.** *For a b-nice sampling $S$, for $b \in [n]$, we have that*

$$\mathcal{L} \leq \mathcal{L}_{practical}(b) \overset{def}{=} \frac{n}{b}\frac{b-1}{n-1}L + \frac{1}{b}\frac{n-b}{n-1}L_{\max} \ . \tag{2.21}$$

Before giving the proof of this lemma, we need to introduce the following argument that we be needed in what follows. It is the double counting argument, allowing us to modifying the way of writing successive sums involving sets and elements of these sets.

**Lemma 2.13** (Double counting)**.** *Let $\alpha_{i,C} \in \mathbb{R}$ for $i = 1, \dots, n$ and $C \in \mathcal{C}$, where $\mathcal{C}$ is a collection of subsets of $[n]$. Then*

$$\sum_{C\in\mathcal{C}}\sum_{i\in C}\alpha_{i,C} \quad = \quad \sum_{i=1}^{n}\sum_{C\in\mathcal{C}\,:\,i\in C}\alpha_{i,C}. \tag{2.22}$$

*Proof of Lemma 2.12 adapted from the one of Proposition 3.8 in [58].* Let $w \in \mathbb{R}^d$. Using $b$-nice sampling, let us sample a mini-batch $B \subset [n]$ of cardinal $b$ and denote by $p_i = \frac{b}{n}$ the probability of $i$ being in the mini-batch B and $P_{i,j}$ the probability of $i, j \in B$. If $i \neq j$, $P_{i,j} = \frac{b(b-1)}{n(n-1)}$ else $P_{i,i} = p_i = \frac{b}{n}$. Finally, let us denote by $p_B = \binom{n}{b}$ the probability that a mini-batch $B$ gets sampled according to $b$-nice sampling.

Let us recall some results arising from the convexity and the $L$-smoothness (resp. $L_i$-smoothness) of $f$ (resp. $f_i$) detailed in Theorem 2.1.5 of [106]. For all $x, y \in \mathbb{R}^d$, for all $i \in [n]$, we have that

$$\|\nabla f(x) - \nabla f(y)\|^2 \leq 2L(f(x) - f(y) - \langle \nabla f(y), x - y\rangle) \ , \tag{2.23}$$

and

$$\|\nabla f_i(x) - \nabla f_i(y)\|^2 \leq 2L_i(f_i(x) - f_i(y) - \langle \nabla f_i(y), x - y\rangle) \ . \tag{2.24}$$

Starting back from (2.2) we have defined the subsampled function as $f_v(w) \overset{\text{def}}{=} \frac{1}{n}\sum_{i=1}^n f_i(w)\cdot v_i$. Thus, we can rewrite

$$f_v(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)v_i \overset{(2.10)}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(w)\frac{n}{b}\mathbb{1}_B(i) = \frac{1}{n}\sum_{i\in B}\frac{1}{p_i}f_i(w) \ . \tag{2.25}$$

We then get that

$$\begin{aligned}\|\nabla f_v(w) - \nabla f_v(w^*)\|^2 &\overset{(2.25)}{=} \left\langle \frac{1}{n}\sum_{i\in B}\frac{1}{p_i}(\nabla f_i(w) - \nabla f_i(w^*)), \frac{1}{n}\sum_{j\in B}\frac{1}{p_j}(\nabla f_j(w) - \nabla f_j(w^*))\right\rangle \\ &= \sum_{i,j\in B}\left\langle \frac{1}{np_i}(\nabla f_i(w) - \nabla f_i(w^*)), \frac{1}{np_j}(\nabla f_j(w) - \nabla f_j(w^*))\right\rangle \ .\end{aligned}$$

By taking expectation over the mini-batches $B$, we get

$$\begin{aligned}
& \mathbb{E}\left[\|\nabla f_v(w) - \nabla f_v(w^*)\|^2\right] \\
= & \sum_{B\subseteq[n]\,:|B|=b} p_B \left\langle \frac{1}{np_i}(\nabla f_i(w) - \nabla f_i(w^*)), \frac{1}{np_j}(\nabla f_j(w) - \nabla f_j(w^*)) \right\rangle \\
\overset{\text{Lemma 2.13}}{=} & \sum_{i,j=1}^{n} \sum_{\substack{B\subseteq[n]\,: \\ |B|=b\wedge i,j\in B}} p_B \left\langle \frac{1}{np_i}(\nabla f_i(w) - \nabla f_i(w^*)), \frac{1}{np_j}(\nabla f_j(w) - \nabla f_j(w^*)) \right\rangle \\
= & \sum_{i,j=1}^{n} \frac{1}{p_i p_j} \left\langle \frac{1}{n}(\nabla f_i(w) - \nabla f_i(w^*)), \frac{1}{n}(\nabla f_j(w) - \nabla f_j(w^*)) \right\rangle \sum_{\substack{B\subseteq[n]\,: \\ |B|=b\wedge i,j\in B}} p_B \\
= & \sum_{i,j=1}^{n} \frac{P_{i,j}}{p_i p_j} \left\langle \frac{1}{n}(\nabla f_i(w) - \nabla f_i(w^*)), \frac{1}{n}(\nabla f_j(w) - \nabla f_j(w^*)) \right\rangle
\end{aligned}$$

where in the second equality we used a double counting argument and in the fourth one we applied the law of total probability.

Now let us break this sum into two parts: $j$ different than $i$ and $j = i$ (in this case we recall that $P_{i,i} = p_i$). Using the values of the probabilities for $b$-nice sampling, that is $p_i = \frac{b}{n}$ and $P_{i,j}/(p_i p_j) = \frac{b(b-1)}{n(n-1)} \frac{n^2}{b^2} = \frac{n}{b}\frac{b-1}{n-1}$ for $i \neq j$, we get

$$\begin{aligned}
& \mathbb{E}\left[\|\nabla f_v(w) - \nabla f_v(w^*)\|^2\right] \\
= & \sum_{i,j=1\,:\,i\neq j}^{n} \frac{P_{i,j}}{p_i p_j} \left\langle \frac{1}{n}(\nabla f_i(w) - \nabla f_i(w^*)), \frac{1}{n}(\nabla f_j(w) - \nabla f_j(w^*)) \right\rangle \\
& + \sum_{i=1}^{n} \frac{1}{n^2}\frac{1}{p_i} \|\nabla f_i(w) - \nabla f_i(w^*)\|^2 \\
= & \frac{n}{b}\frac{b-1}{n-1} \sum_{i,j=1}^{n} \left\langle \frac{1}{n}(\nabla f_i(w) - \nabla f_i(w^*)), \frac{1}{n}(\nabla f_j(w) - \nabla f_j(w^*)) \right\rangle \\
& + \frac{1}{b}\left(1 - \frac{b-1}{n-1}\right) \sum_{i=1}^{n} \frac{1}{n} \|\nabla f_i(w) - \nabla f_i(w^*)\|^2 \\
= & \frac{n}{b}\frac{b-1}{n-1} \|\nabla f(w) - \nabla f(w^*)\|^2 + \frac{1}{b}\frac{n-b}{n-1} \sum_{i=1}^{n} \frac{1}{n} \|\nabla f_i(w) - \nabla f_i(w^*)\|^2 \\
\overset{(2.23)\,\&\,(2.24)}{\leq} & 2\frac{n}{b}\frac{b-1}{n-1} L(f(w) - f(w^*)) \\
& + 2\frac{1}{b}\frac{n-b}{n-1} \sum_{i=1}^{n} \frac{1}{n} L_i \left(f_i(w) - f_i(w^*) - \langle \nabla f_i(w^*), w - w^*\rangle\right) \\
\leq & 2\underbrace{\left(\frac{n}{b}\frac{b-1}{n-1} L + \frac{1}{b}\frac{n-b}{n-1} L_{\max}\right)}_{\mathcal{L}_{\text{practical}}(b)} (f(w) - f(w^*)) \ .
\end{aligned}$$

This concludes the proof showing that $\mathcal{L} \leq \mathcal{L}_{\text{practical}}(b)$ for $b$-nice sampling. □

We notice that $\mathcal{L}_{\text{practical}}(1) = L_{\max}$ and $\mathcal{L}_{\text{practical}}(n) = L$, achieving both limits of (2.8).

#### 2.3.3.3 Simple bound

The second bound, chronologically the first one we proved in [45], is technically the simplest to derive, which is why we refer to it as the *simple bound*.

**Theorem 2.14** (Simple bound)**.** *For a b-nice sampling $S$, for $b \in [n]$, we have that*

$$\mathcal{L} \leq \mathcal{L}_{\text{simple}}(b) \overset{def}{=} \frac{n}{b}\frac{b-1}{n-1}\bar{L} + \frac{1}{b}\frac{n-b}{n-1}L_{\max} \ , \tag{2.26}$$

*Proof of Theorem 2.14.* To derive this bound on $\mathcal{L}$ we use that

$$L_B \leq \frac{1}{b}\sum_{j\in B} L_j \ , \tag{2.27}$$

which follows from repeatedly applying Lemma 2.23. For $b \geq 2$, it follows from (2.20) and (2.27) that

$$\mathcal{L} \leq \frac{1}{b\binom{n-1}{b-1}} \max_{i=1,\dots,n} \left\{ \sum_{\substack{B\subseteq[n]:\\ |B|=b\wedge i\in B}} \sum_{j\in B} L_j \right\} . \tag{2.28}$$

Using a double counting argument we can show that

$$\begin{aligned}
\sum_{\substack{B\subseteq[n]:\\ |B|=b\wedge i\in B}} \sum_{j\in B} L_j &= \sum_{j=1}^{n} \sum_{\substack{B\subseteq[n]:\\ |B|=b\wedge i,j\in B}} L_j \\
&= \sum_{j\neq i} \sum_{\substack{B\subseteq[n]:\\ |B|=b\wedge i,j\in B}} L_j + \sum_{\substack{B\subseteq[n]:\\ |B|=b\wedge i\in B}} L_i \\
&= \sum_{j\neq i} \binom{n-2}{b-2} L_j + \binom{n-1}{b-1} L_i \\
&= \binom{n-2}{b-2}(n\bar{L} - L_i) + \binom{n-1}{b-1} L_i \ .
\end{aligned} \tag{2.29}$$

Inserting this into (2.28) gives

$$\begin{aligned}
\mathcal{L} &\leq \frac{1}{b\binom{n-1}{b-1}} \max_{i=1,\dots,n} \left\{ \binom{n-2}{b-2} n\bar{L} + \left( \binom{n-1}{b-1} - \binom{n-2}{b-2} \right) L_{\max} \right\} \\
&= \frac{n\binom{n-2}{b-2}}{b\binom{n-1}{b-1}}\bar{L} + \frac{\binom{n-1}{b-1} - \binom{n-2}{b-2}}{b\binom{n-1}{b-1}} L_{\max} \\
&= \frac{n}{b}\frac{b-1}{n-1}\bar{L} + \frac{1}{b}\frac{n-b}{n-1}L_{\max} \ .
\end{aligned} \tag{2.30}$$

We also verify that this bound is valid for 1-nice sampling. Indeed, we already have that in this case $\mathcal{L} = L_{\max}$. □

The previous bound interpolates, respectively for $b = 1$ and $b = n$, between $L_{\max}$ and $\bar{L}$, unlike the practical bound which goes from $L_{\max}$ and $L$. On the one hand, we have that $\mathcal{L}_{\text{simple}}(b)$ is a good bound for when $b$ is small, since $\mathcal{L}_{\text{simple}}(1) = L_{\max}$. Though $\mathcal{L}_{\text{simple}}(b)$ may not be a good bound for large $b$, since $\mathcal{L}_{\text{simple}}(n) = \overline{L} \quad \geq \quad L$, thanks to (2.16).Thus $\mathcal{L}_{\text{simple}}(b)$ does not achieve the left-hand side of (2.8). Indeed $\overline{L}$ can be far from $L$. For instance[5], if $f(w) = \frac{1}{n}\sum_{i\in[n]} \frac{1}{2}(x_i^\top w - y_i)^2$ is a quadratic function, then we have that $\overline{L} = \frac{1}{n}\mathbf{Tr}\left(\mathbf{X}^\top\mathbf{X}\right)$ and $L = \frac{1}{n}\lambda_{\max}(\mathbf{X}^\top\mathbf{X})$. Thus if the eigenvalues of $\mathbf{X}^\top\mathbf{X}$ are all equal then $\overline{L} = dL$. Alternatively, if one eigenvalue is significantly larger than the rest then $\overline{L} \approx L$.

[5] We numerically explore such extreme settings in Section 2.5.

#### 2.3.3.4 Bernstein bound

Due to this shortcoming of $\mathcal{L}_{\text{simple}}$, we then derived the *Bernstein bound*. This bound explicitly depends on $L$ instead of $\bar{L}$ but is looser than the practical. It is developed through a specialized variant of a matrix Bernstein inequality [148, 149] for sampling *without* replacement in section 2.7.2.

**Theorem 2.15** (Bernstein bound)**.** *The expected smoothness constant is upper bounded by*

$$\mathcal{L} \leq \mathcal{L}_{Bernstein}(b) \overset{def}{=} 2\frac{b-1}{b}\frac{n}{n-1}L + \frac{1}{b}\left(\frac{n-b}{n-1} + \frac{4}{3}\log d\right) L_{\max} \ . \tag{2.31}$$

Before starting the proof of Theorem 2.15, we need to rewrite the expected smoothness constant as the maximum over an expectation. Let $S^i$ be a $(b-1)$-nice sampling over $[n] \setminus \{i\}$. We can write

$$\begin{aligned} \mathcal{L} &= \frac{1}{\binom{n-1}{b-1}} \max_{i=1,\dots,n} \left\{ \sum_{\substack{B \subseteq [n]: \\ |B|=b \wedge i \in B}} L_B \right\} \\ &= \max_{i=1,\dots,n} \mathbb{E}\left[L_{S^i \cup \{i\}}\right] \\ &\overset{\text{Lemma 2.7}}{=} \max_{i=1,\dots,n} U\mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j \in S^i \cup \{i\}} x_j x_j^\top\right)\right] \ . \end{aligned} \tag{2.32}$$

One can come back to the definition of the subsample smoothness constant in (2.13) and interpret previous expression as an expectation of the largest eigenvalue of a sum of matrices. This insight allows us to apply a matrix Bernstein inequality, see Theorem 2.37, to bound $\mathcal{L}$.

For the proof of Theorem 2.15, we need Lemmas 2.16 to 2.18.

**Lemma 2.16.** *Let $x_j \in \mathbb{R}^d$, $i \in \{1, \dots, n\}$ and let $S^i$ be a $(b-1)$-nice sampling over the set $[n] \setminus \{i\}$. It follows that*

$$\mathbb{E}\left[\sum_{j \in S^i \cup \{i\}} x_j x_j^\top\right] = x_i x_i^\top + \frac{b-1}{n-1}\sum_{j=1, j \neq i}^{n} x_j x_j^\top \ . \tag{2.33}$$

*Proof of Lemma 2.16.* This results follows using a double-counting argument at the fourth line of the

computation.

$$\begin{aligned}
\mathbb{E}\left[\sum_{j\in S^i\cup\{i\}} x_j x_j^\top\right] &= \frac{1}{\binom{n-1}{b-1}} \sum_{\substack{B\subseteq[n]\setminus\{i\}:\\ |B|=b-1}} \sum_{j\in B\cup\{i\}} x_j x_j^\top \\
&= \frac{1}{\binom{n-1}{b-1}}\left(\binom{n-1}{b-1} x_i x_i^\top + \sum_{\substack{B\subseteq[n]\setminus\{i\}:\\ |B|=b-1}} \sum_{j\in B} x_j x_j^\top\right) \\
&= x_i x_i^\top + \frac{1}{\binom{n-1}{b-1}} \sum_{\substack{B\subseteq[n]\setminus\{i\}:\\ |B|=b-1}} \sum_{j\in B} x_j x_j^\top \\
&= x_i x_i^\top + \frac{1}{\binom{n-1}{b-1}} \sum_{j=1, j\neq i}^{n} \sum_{\substack{B\subseteq[n]\setminus\{i\}:\\ |B|=b-1\wedge j\in B}} x_j x_j^\top \\
&= x_i x_i^\top + \frac{\binom{n-2}{b-2}}{\binom{n-1}{b-1}} \sum_{j=1, j\neq i}^{n} x_j x_j^\top \\
&= x_i x_i^\top + \frac{b-1}{n-1} \sum_{j=1, j\neq i}^{n} x_j x_j^\top \ .
\end{aligned}$$

□

We then introduce another two lemmas which give a first intermediate bound.

**Lemma 2.17.** *Let $x_j \in \mathbb{R}^d$ for $j \in [n]$, let $i \in [n]$ and let $S^i$ be a $(b-1)$-nice sampling over $[n]\setminus\{i\}$. We have*

$$\begin{aligned}
U\mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j\in S^i\cup\{i\}} x_j x_j^\top\right)\right] \leq & \frac{1}{b(n-1)}\left((n-b)L_i + n(b-1)L\right) \\
& + U\mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j\in S^i} x_j x_j^\top - \frac{1}{b}\frac{b-1}{n-1}\sum_{j\in[n]\setminus\{i\}} x_j x_j^\top\right)\right] \ . \qquad (2.34)
\end{aligned}$$

*Proof of Lemma 2.17.* Expanding the expectation we have

$$\begin{aligned}
&\mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j\in S^i\cup\{i\}} x_j x_j^\top\right)\right] \\
&\leq \lambda_{\max}\left(\mathbb{E}\left[\frac{1}{b}\sum_{j\in S^i\cup\{i\}} x_j x_j^\top\right]\right) + \mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j\in S^i\cup\{i\}} x_j x_j^\top - \mathbb{E}\left[\sum_{j\in S^i\cup\{i\}} x_j x_j^\top\right]\right)\right] \\
&= \frac{1}{b}\lambda_{\max}\left(x_i x_i^\top + \frac{b-1}{n-1}\sum_{j=1,\, j\neq i}^{n} x_j x_j^\top\right) + \mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j\in S^i\cup\{i\}} x_j x_j^\top - \frac{1}{b}\left(x_i x_i^\top + \frac{b-1}{n-1}\sum_{j=1,\, j\neq i}^{n} x_j x_j^\top\right)\right)\right] \\
&= \frac{1}{b}\lambda_{\max}\left(\frac{1}{n-1}\left((n-b)x_i x_i^\top + (b-1)\sum_{j=1}^{n} x_j x_j^\top\right)\right) + \mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j\in S^i} x_j x_j^\top - \frac{1}{b}\frac{b-1}{n-1}\sum_{j\in[n]\setminus\{i\}} x_j x_j^\top\right)\right] \\
&\leq \frac{1}{b(n-1)}\left((n-b)\frac{L_i}{U} + n(b-1)\frac{L}{U}\right) + \mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j\in S^i} x_j x_j^\top - \frac{1}{b}\frac{b-1}{n-1}\sum_{j\in[n]\setminus\{i\}} x_j x_j^\top\right)\right] \ ,
\end{aligned}$$

where in the first inequality we add and remove the mean and then apply Lemma 2.23. In the second equality we explicit the mean with Lemma 2.16 and in the last inequality we use again Lemma 2.23 for the left-hand side term. Finally, we multiply by $U$ on both sides of the inequality. □

**Lemma 2.18.** *Let $x_j \in \mathbb{R}^d$ for $j \in [n]$ and let $S^i$ be a $(b-1)$-nice sampling over $[n] \setminus \{i\}$, for every $i \in [n]$. It follows that*

$$\mathcal{L} \leq \mathcal{L}_{practical}(b) + U \max_{i \in [n]} \mathbb{E}\left[\lambda_{\max}\left(\mathbf{N}_i\right)\right] \ , \tag{2.35}$$

*with $\mathbf{N}_i \overset{def}{=} \frac{1}{b}\sum_{j \in S^i} x_j x_j^\top - \frac{1}{b}\frac{b-1}{n-1}\sum_{j \in [n]\setminus\{i\}} x_j x_j^\top$ and where the* practical bound *is given by*

$$\mathcal{L}_{practical}(b) \overset{Lemma\ 2.12}{=} \frac{n}{b}\frac{b-1}{n-1}L + \frac{1}{b}\frac{n-b}{n-1}L_{\max} \ .$$

*Proof of Lemma 2.18.* The result comes from applying re-writing $\mathcal{L}$ as an expectation of the largest eigenvalue of a sum of matrices. Then we apply Lemma 2.17 and then taking the maximum over all $i \in [n]$. Thus, we have

$$\begin{aligned}
\mathcal{L} \overset{(2.32)}{=} & \max_{i=1,\dots,n} U\mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j \in S^i \cup \{i\}} x_j x_j^\top\right)\right] \\
\overset{\text{Lemma 2.17}}{\leq} & \max_{i=1,\dots,n}\left\{\frac{1}{b(n-1)}\left((n-b)L_i + n(b-1)L\right)\right. \\
& \left. + U\mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j \in S^i} x_j x_j^\top - \frac{1}{b}\frac{b-1}{n-1}\sum_{j \in [n]\setminus\{i\}} x_j x_j^\top\right)\right]\right\} \\
\leq & \frac{n}{b}\frac{b-1}{n-1}L + \frac{1}{b}\frac{n-b}{n-1}L_{\max} + \max_{i=1,\dots,n} U\mathbb{E}\left[\lambda_{\max}\left(\frac{1}{b}\sum_{j \in S^i} x_j x_j^\top - \frac{1}{b}\frac{b-1}{n-1}\sum_{j \in [n]\setminus\{i\}} x_j x_j^\top\right)\right].
\end{aligned}$$

□

*Proof of Theorem 2.15.* Applying the previous lemma we get

$$\mathcal{L} \overset{(2.35)}{\leq} \frac{n}{b}\frac{b-1}{n-1}L + \frac{1}{b}\frac{n-b}{n-1}L_{\max} + \max_{i=1,\dots,n} U\mathbb{E}\left[\lambda_{\max}\left(\mathbf{N}\right)\right] \ , \tag{2.36}$$

with $\mathbf{N} \overset{\text{def}}{=} \frac{1}{b}\sum_{j \in S^i} x_j x_j^\top - \frac{1}{b}\frac{b-1}{n-1}\sum_{j \in [n]\setminus\{i\}} x_j x_j^\top$.

To further our argument, we will encode different samplings using unit coordinate vectors. Let $e_1, \dots, e_n \in \mathbb{R}^n$ be the unit coordinate vectors. Let $S^i = \{S_1^i, \dots, S_b^i\}$ denote an arbitrary but fixed ordering of the elements of $S^i$. With this we can encode the sampling without replacement as

$$\sum_{j \in S^i} x_j x_j^\top = \sum_{k=1}^{b-1} \sum_{j \in [n]\setminus\{i\}} (e_j)_{S_k^i} x_j x_j^\top \ . \tag{2.37}$$

Using this notation, the matrix $\mathbf{N}$ which can be further decomposed as

$$\begin{aligned} N &= \frac{1}{b}\sum_{j\in S^i} x_j x_j^\top - \frac{1}{b}\frac{b-1}{n-1}\sum_{j\in[n]\setminus\{i\}} x_j x_j^\top \\ &= \frac{1}{b}\sum_{k=1}^{b-1}\sum_{j\in[n]\setminus\{i\}} (e_j)_{S_k^i} x_j x_j^\top - \frac{1}{b}\frac{b-1}{n-1}\sum_{j\in[n]\setminus\{i\}} x_j x_j^\top \\ &= \sum_{k=1}^{b-1}\frac{1}{b}\sum_{j\in[n]\setminus\{i\}}\left((e_j)_{S_k^i} - \frac{1}{n-1}\right) x_j x_j^\top \\ &\overset{\text{def}}{=} \sum_{k=1}^{b-1}\mathbf{M}_k \ . \end{aligned}$$

where we have encoded the sampling $S^i$ using unit coordinate vectors. The matrices $\mathbf{M}_1,\dots,\mathbf{M}_{b-1}$ are sampled *without* replacement from the set

$$\left\{\sum_{j\in[n]\setminus\{i\}}\frac{1}{b}\left(x_j - \frac{1}{n-1}\right)x_j x_j^\top \ : \ x\in\{e_1,\dots,e_{i-1},e_{i+1},\dots,e_n\}\right\} \ . \tag{2.38}$$

Now let $\mathbf{V}_1,\dots,\mathbf{V}_b$ be matrices sampled *with* replacement from (2.38) and let $\mathbf{V}_k \overset{\text{def}}{=} \frac{1}{b}\sum_{j\in[n]\setminus\{i\}}\left(z_j^k - \frac{1}{n-1}\right)x_j x_j^\top$ and $Y \overset{\text{def}}{=} \sum_{k=1}^{b-1}\mathbf{V}_k$ thus the vectors $z^k$ are sampled with replacement from $\{e_1,\dots,e_{i-1},e_{i+1},\dots,e_n\}$. Consequently

$$\mathbb{P}\left[z_j^k = 1\right] = \frac{1}{n-1} \ , \quad \forall j\in\{1,\dots,i-1,i+1,\dots,n\} \ .$$

We are now in a position to apply the Bernstein matrix inequality. To this end we have

- A sum of centered random matrices: $\mathbb{E}\left[\mathbf{V}_k\right] = 0$.

- Let $k^*$ be the unique index such that $z_{k^*}^k = 1$. We have a uniform bound of the largest eigenvalue of our $\mathbf{V}_k$

$$\begin{aligned} \lambda_{\max}(\mathbf{V}_k) &= \frac{1}{b}\lambda_{\max}\left(\sum_{j\in[n]\setminus\{i\}} z_j^k x_j x_j^\top - \frac{1}{n-1}\sum_{j\in[n]\setminus\{i\}} x_j x_j^\top\right) \\ &\le \frac{1}{b}\lambda_{\max}\left(\sum_{j\in[n]\setminus\{i\}} z_j^k x_j x_j^\top\right) \\ &= \frac{1}{b}\lambda_{\max}\left(x_{k^*}x_{k^*}^\top\right) \\ &\le \frac{1}{b}\frac{L_{\max}}{U} \ , \end{aligned} \tag{2.39}$$

  where we applied the Lemma 2.24 in the first inequality.

- And a bound on the variance too

$$\begin{aligned}
\mathbb{E}\left[\mathbf{V}_k^2\right] &= \mathbb{E}\left(\frac{1}{b}\sum_{j\in[n]\setminus\{i\}}\left(z_j^k - \frac{1}{n-1}\right)x_j x_j^\top\right)^2 \\
&= \frac{1}{b^2}\mathbb{E}\left(\sum_{j,p\in[n]\setminus\{i\}} z_j^k z_p^k x_j x_j^\top x_p x_p^\top - \frac{2}{n-1}\sum_{j,p\in[n]\setminus\{i\}} z_j^k x_j x_j^\top x_p x_p^\top \right. \\
&\qquad \left. + \frac{1}{(n-1)^2}\sum_{j,p\in[n]\setminus\{i\}} x_j x_j^\top x_p x_p^\top\right) \\
&= \frac{1}{b^2}\sum_{j,p\in[n]\setminus\{i\}}\left(\left(\mathbb{E}\left[z_j^k z_p^k\right] x_j x_j^\top x_p x_p^\top - \frac{2}{n-1}\mathbb{E}\left[z_j^k\right] x_j x_j^\top x_p x_p^\top + \frac{1}{(n-1)^2} x_j x_j^\top x_p x_p^\top\right)\right. \\
&= \frac{1}{b^2}\sum_{j,p\in[n]\setminus\{i\}}\left(\mathbb{E}\left[z_j^k z_p^k\right] x_j x_j^\top x_p x_p^\top - \frac{2}{(n-1)^2} x_j x_j^\top x_p x_p^\top + \frac{1}{(n-1)^2} x_j x_j^\top x_p x_p^\top\right) \\
&= \frac{1}{b^2}\left(\frac{1}{n-1}\sum_{j\in[n]\setminus\{i\}} x_j x_j^\top x_j x_j^\top - \frac{1}{(n-1)^2}\sum_{j,p\in[n]\setminus\{i\}} x_j x_j^\top x_p x_p^\top\right) \quad , \qquad (2.40)
\end{aligned}$$

where, in the last equality, we used that $z_j^k z_p^k = 0$ if $j \neq p$ and $\mathbb{E}\left[z_j^k z_j^k\right] = \mathbb{E}\left[z_j^k\right] = \frac{1}{n-1}$, so that

$$\begin{aligned}
\sum_{j,p\in[n]\setminus\{i\}} \mathbb{E}\left[z_j^k z_p^k\right] x_j x_j^\top x_p x_p^\top &= \mathbb{E}\left[\sum_{j,p\in[n]\setminus\{i\}} z_j^k z_p^k\right] x_j x_j^\top x_p x_p^\top \\
&= \sum_{j\in[n]\setminus\{i\}} \mathbb{E}\left[z_j^k z_j^k\right] x_j x_j^\top x_p x_p^\top \\
&= \frac{1}{n-1}\sum_{j\in[n]\setminus\{i\}} x_j x_j^\top x_p x_p^\top \quad .
\end{aligned}$$

Summing in (2.40), taking the largest eigenvalue and applying Lemma 2.24 results in

$$\begin{aligned}
\lambda_{\max}\left(\sum_{k=1}^{b-1}\mathbb{E}\left[\mathbf{V}_k^2\right]\right) &\leq \lambda_{\max}\left(\sum_{k=1}^{b-1}\frac{1}{b^2}\frac{1}{n-1}\sum_{j\in[n]\setminus\{i\}} x_j x_j^\top x_j x_j^\top\right) \\
&\leq \frac{b-1}{b^2}\left(\max_{j\in[n]\setminus\{i\}}\lambda_{\max}\left(x_j x_j^\top\right)\right)\cdot\lambda_{\max}\left(\frac{1}{n-1}\sum_{j\in[n]\setminus\{i\}} x_j x_j^\top\right) \\
&\leq \frac{b-1}{b^2}\frac{L_{\max}}{U^2}L_{[n]\setminus\{i\}} \quad . \qquad (2.41)
\end{aligned}$$

Considering (2.39) and (2.41) and applying the matrix Bernstein concentration inequality in Theorem 2.37 we get

$$U\mathbb{E}\left[\lambda_{\max}(\mathbf{N})\right] \leq \sqrt{2\frac{b-1}{b^2}L_{\max}L_{[n]\setminus\{i\}}\log d} + \frac{1}{3}\frac{L_{\max}}{b}\log d \quad .$$

Taking the maximum over $i$ and using $L_{[n]\setminus\{i\}} \leq \frac{n}{n-1}L$ we have that

$$\max_{i=1,\ldots,n} U\mathbb{E}\left[\lambda_{\max}(\mathbf{N})\right] \leq \sqrt{2\frac{b-1}{b^2}\frac{n}{n-1}L_{\max}L\log d} + \frac{1}{3}\frac{L_{\max}}{b}\log d \quad .$$

Combining the above result with (2.36) leads us to

$$\begin{aligned}\mathcal{L} &\leq \frac{(n-b)L_{\max}}{b(n-1)} + \frac{n(b-1)L}{b(n-1)} + \sqrt{2\left(\frac{b-1}{b}\frac{n}{n-1}L\right)\cdot\left(\frac{1}{b}L_{\max}\log d\right)} + \frac{1}{3}\frac{L_{\max}}{b}\log d \\ &\leq \frac{(n-b)L_{\max}}{b(n-1)} + \frac{n(b-1)L}{b(n-1)} + \frac{b-1}{b}\frac{n}{n-1}L + \frac{4}{3}\frac{L_{\max}}{b}\log(d) \\ &= 2\frac{b-1}{b}\frac{n}{n-1}L + \frac{1}{b}\left(\frac{4}{3}\log(d) + \frac{n-b}{n-1}\right)L_{\max}\ .\end{aligned}$$

where in the second inequality we used the inequality $\sqrt{2ab} \leq a+b$. This completes the proof of Theorem 2.15. □

Checking again the bounds of $\mathcal{L}_{\text{Bernstein}}(b)$, we have on the one hand that $\mathcal{L}_{\text{Bernstein}}(1) = \left(1+\frac{4}{3}\log d\right)L_{\max} \geq L_{\max}$, thus there is a little bit of slack for $b$ small. On the other hand, using $\frac{1}{n}L_{\max} \leq L$ (see Lemma 2.9), we have that

$$\mathcal{L}_{\text{Bernstein}}(n) = 2L + \frac{1}{n}\frac{4}{3}\log d\, L_{\max} \leq \left(2+\frac{4}{3}\log d\right)L\ ,$$

which depends only logarithmically on $d$. Thus we expect the *Bernstein bound* to be more useful in the large $d$ domains, as compared to the *simple bound*. We confirm this numerically in Section 2.5.1.

**Remark 2.19.** *The simple bound is relatively tight for $b$ small, while the Bernstein bound is better for large $b$ and large $d$. Fortunately, we can obtain a more refined bound by taking the minimum of the simple and the Bernstein bounds. This is highlighted numerically in Section 2.5.*

## 2.4 Optimal mini-batch sizes

Now that we have established the *simple* and the *Bernstein bounds*, we can minimize the total complexity (2.19) in the mini-batch size.

For instance for the *simple bound*, given $\epsilon > 0$ and plugging in (2.26) into (2.19) gives

$$K_{\text{total}}(b) \leq \max\left\{g_{\text{simple}}(b), h(b)\right\}\log\left(\frac{1}{\epsilon}\right)\ ,$$

where $g_{\text{simple}}(b) \overset{\text{def}}{=} \frac{4\left(n\bar{L}-L_{\max}+(n-1)\lambda\right)}{\mu(n-1)}b + \frac{4n\left(L_{\max}-\bar{L}\right)}{\mu(n-1)}$, and $h(b) \overset{\text{def}}{=} -\frac{4(L_{\max}+\lambda)}{\mu(n-1)}b + n\left(1+\frac{1}{n-1}\frac{4(L_{\max}+\lambda)}{\mu}\right)$.

**Remark 2.20.** *The right-hand side term $h(b)$ is common to all our bounds since it does not depend on $\mathcal{L}$. It linearly decreases from $h(1) = n + \frac{4(L_{\max}+\lambda)}{\mu}$ to $h(n) = n$.*

We note that $g_{\text{simple}}(b)$ is a linearly increasing function of $b$, because $L_{\max} \leq n\bar{L}$ (as proven in Lemma 2.9). One can easily verify that $g_{\text{simple}}(b)$ and $h(b)$ cross, as presented in Figure 2.2, by looking at initial and final values:

- At $b{=}1$, $g_{\text{simple}}(1) = \frac{4}{\mu}(L_{\max}+\lambda) = h(1) - n$. So, $g_{\text{simple}}(1) \leq h(1)$.
- At $b{=}n$, $g_{\text{simple}}(n) = \frac{4(\bar{L}+\lambda)}{\mu}n = \frac{4(\bar{L}+\lambda)}{\mu}h(n)$. Since $\bar{L} \geq \mu$, we get $g_{\text{simple}}(n) \geq h(n)$.

Consequently, solving $g_{\text{simple}}(b) = h(b)$ in $b$ gives the optimal mini-batch size

$$b^*_{\text{simple}} = \left\lfloor 1 + \frac{\mu(n-1)}{4(\bar{L}+\lambda)} \right\rfloor\ . \tag{2.42}$$

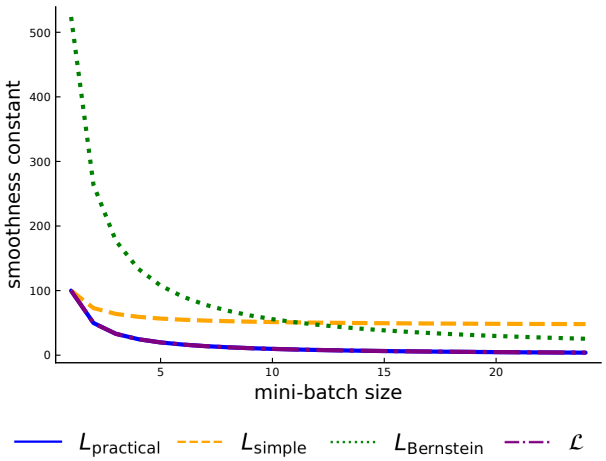



**Figure 2.3:** *Expected smoothness constant $\mathcal{L}$ and its upper-bounds as a function of the mini-batch size $b$ for ridge regression on the unscaled* staircase eigval *dataset ($\lambda = 10^{-3}$).*

For the *Bernstein bound*, plugging (2.31) into (2.19) leads to

$$K_{\text{total}}(b) \leq \max\left\{g_{\text{Bernstein}}(b), h(b)\right\} \log\left(\frac{1}{\epsilon}\right) \quad , \tag{2.43}$$

where

$$\begin{aligned} g_{\text{Bernstein}}(b) \overset{\text{def}}{=} & \frac{4}{\mu(n-1)}\left(2nL - L_{\max} + (n-1)\lambda\right) b \\ & + \frac{4n}{\mu(n-1)}\left(L_{\max} - 2L\right) + \frac{16}{3\mu}\log(d) L_{\max} \quad . \end{aligned}$$

The function $g_{\text{Bernstein}}$ is also linearly increasing in $b$ and its initial and final values are

- At $b = 1$, $g_{\text{Bernstein}}(1) = (1 + \frac{4}{3}\log d)\frac{4L_{\max}}{\mu} + \frac{4\lambda}{\mu}$.
- At $b = n$, $g_{\text{Bernstein}}(n) = n\frac{4(2L+\lambda)}{\mu} + \frac{16}{3\mu}(L_{\max} + \lambda)\log(d)$. Since $L \geq \mu$, we get $g_{\text{Bernstein}}(n) \geq h(n)$.

Yet, it is unclear whether $g_{\text{Bernstein}}(1)$ is dominated by $h(1)$. This is why we need to distinguish two cases to minimize the total complexity, which leads to the following solution

$$b^*_{\text{Bernstein}} = \begin{cases} \left\lfloor 1 + \frac{\mu(n-1)}{4(2L+\lambda)} - \frac{4}{3}\log d \frac{n-1}{n}\frac{L_{\max}}{2L+\lambda} \right\rfloor, & \text{if } \frac{4}{3}\frac{4L_{\max}}{\mu}\log d \leq n, \\ 1, & \text{otherwise} \quad . \end{cases}$$

In the first case, the problem is well-conditioned and $g_{\text{Bernstein}}$ and $h$ do cross at a mini-batch size between 1 and $n$. In the second case, the total complexity $K_{\text{total}}$ is governed by $g_{\text{Bernstein}}$ because $g_{\text{Bernstein}}(b) \geq h(b)$ for all $b \in [n]$, and the resulting optimal mini-batch size is $b = 1$.

## 2.5 Numerical study

All the experiments were run in Julia and the code is freely available on `https://github.com/gowerrobert/StochOpt.jl`. In our experiments we used the following implementation of the mini-batch SAGA algorithm which is more efficient in the computation of the gradient estimate than the naive Algorithm 7.

**Algorithm 8** JacSketch practical implementation of $b$-nice SAGA

**Parameters:** mini-batch size $b$, step size $\gamma > 0$
**Initialize:** $w^0 \in \mathbb{R}^d$, $\mathbf{J}^0 \in \mathbb{R}^{d\times n}$, $u^0 = \frac{1}{n}\mathbf{J}^0 e$
**for** $k = 0, 1, \ldots, K-1$ **do**
  Sample a fresh batch $B \subseteq [n]$ s.t. $|B| = b$
  $\text{aux} = \sum_{i\in B}(\nabla f_i(w^k) - \mathbf{J}^k_{:i})$ ▷ update the auxiliary vector
  $g^k = u^k + \frac{1}{b}\text{aux}$ ▷ update the unbiased gradient estimate
  $u^{k+1} = u^k + \frac{1}{n}\text{aux}$ ▷ update the biased gradient estimate
  $\mathbf{J}^{k+1}_{:i} = \begin{cases} \mathbf{J}^k_{:i} & i \notin B \\ \nabla f_i(w^k) & i \in B \end{cases}$ ▷ update the Jacobian estimate
  $w^{k+1} = w^k - \gamma g^k$ ▷ take a step
**Output:** $w^K$ ▷ return weights vector

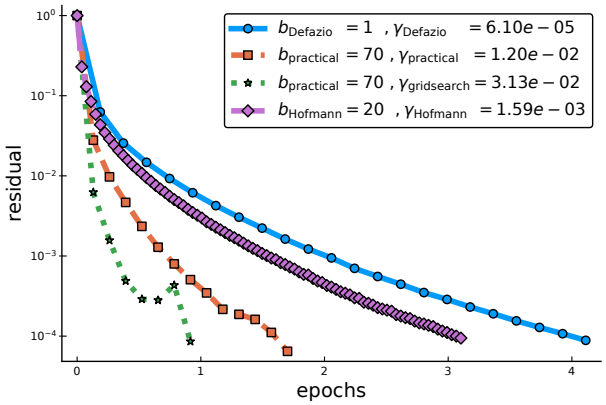



**Figure 2.4:** *Comparison of SAGA settings for ridge regression on the unscaled* slice *dataset* ($\lambda = 10^{-1}$).

### 2.5.1 Upper-bounds of the expected smoothness constant

First we experimentally verify that our upper-bounds hold and how much slack there is between them and $\mathcal{L}$ given in (2.20). For ridge regression applied to artificially generated small datasets, we compute (2.20) and compare it to our *practical*, *simple* and *Bernstein bounds*. Our data are matrices $\mathbf{X} \in \mathbb{R}^{n\times d}$ defined as follows

- *uniform* $(n = 24, d = 50) : [\mathbf{X}]_{ij} \sim \mathcal{U}([0, 1))$
- *alone eigval* $(n = d = 24) : \mathbf{X} = \text{diag}\,(1, \ldots, 1, 100)$
- *staircase eigval* $(n = d = 24) : \mathbf{X} = \text{diag}\left(1, 10\sqrt{1/n}, \ldots, 10\sqrt{(n-2)/n}, 10\right)$

In Figure 2.3 we see that $\mathcal{L}_{\text{practical}}$ is arbitrarily close to $\mathcal{L}$, making it hard to distinguish the two line plots. This was the case in many other experiments, which we defer to Section 2.8.1. For this reason, we use $\gamma_{\text{practical}}$ in our experiments with the SAGA method.

Furthermore, in accordance with our discussion in Section 2.3.3, we have that $\mathcal{L}_{\text{simple}}$ and $\mathcal{L}_{\text{Bernstein}}$ are close to $\mathcal{L}$ when $b$ is small and large, respectively. In Section 2.8.2 we show, by applying ridge and regularized logistic regression to publicly available datasets from LIBSVM[6] and the UCI repository[7], that the *simple bound* performs better than the *Bernstein bound* when $n \gg d$, and conversely for $d$

[6] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`
[7] `https://archive.ics.uci.edu/ml/datasets/`

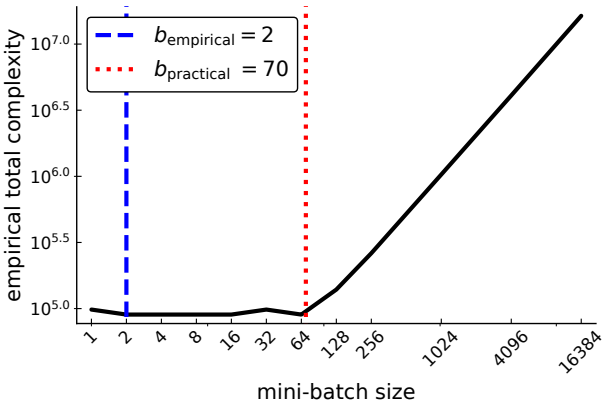



**Figure 2.5:** *Total complexity versus the mini-batch size for ridge regression on the unscaled* slice *dataset ($\lambda = 10^{-1}$).*

slightly smaller than $n$, larger than $n$ or when scaling the data.

### 2.5.2 Related step size estimation

Different bounds on $\mathcal{L}$ also give different step sizes (2.17). Plugging in our estimates $\mathcal{L}_{\text{simple}}$, $\mathcal{L}_{\text{Bernstein}}$ and $\mathcal{L}_{\text{practical}}$ into (2.17) gives the step sizes $\gamma_{\text{simple}}$, $\gamma_{\text{Bernstein}}$ and $\gamma_{\text{practical}}$, respectively. We compare our resulting step sizes to $\gamma_{\mathcal{L}}$ where $\mathcal{L}$ is given by (2.20) and to the step size given by Hofmann et al. [68], which is $\gamma_{\text{Hofmann}}(b) = \frac{K}{2L_{\max}(1+K+\sqrt{1+K^2})}$, where $K \overset{\text{def}}{=} \frac{4bL_{\max}}{n\mu}$. We can see in Figure 2.3, that for $b = 1$, all the step sizes are approximately the same, with the exceptions of the Bernstein step size. For $b > 5$, all of our step sizes are larger than $\gamma_{\text{Hofmann}}(b)$, in particular $\gamma_{\text{practical}}(b)$ is significantly larger. These observations are verified in other artificial and real data examples in Sections 2.8.3 and 2.8.4.

### 2.5.3 Comparison with previous SAGA settings

Here we compare the performance of SAGA when using the mini-batch size and step size $b = 1, \gamma_{\text{Defazio}} \overset{\text{def}}{=} 1/3(n\mu + L_{\max})$ given in [33], $b = 20$ and $\gamma_{\text{Hofmann}} = 20/n\mu$ given in Hofmann et al. [68], to our new practical mini-batch size $b_{\text{practical}} = \left\lfloor 1 + \frac{\mu(n-1)}{4(L+\lambda)} \right\rfloor$ and step size $\gamma_{\text{practical}}$. Our goal is to verify how much our parameter setting can improve practical performance. We also compare with a step size $\gamma_{\text{gridsearch}}$ obtained by grid search over odd powers of 2. These methods are run until they reach a relative error of $10^{-4}$.

We find in Figure 2.4 that our parameter settings $(\gamma_{\text{practical}}, b_{\text{practical}})$ significantly outperforms the previously suggested parameters, and is even comparable to grid search. Finally, we show in Section 2.8.5 that the settings $(\gamma_{\text{Hofmann}}, b = 20)$ can lead to very poor performance compared to our settings.

### 2.5.4 Optimality of our mini-batch size

In the last experiment, detailed in Section 2.8.6, we show that our estimation of the optimal mini-batch size $b_{\text{practical}}$ leads to a faster implementation of SAGA. We build a grid of mini-batch sizes[8] and compute the empirical total complexity required to achieve a relative error of $10^{-4}$, as in section 2.5.3. In Figure 2.5 we can see that the empirical complexity when using $b_{\text{practical}}$ is almost the same as one for the optimal mini-batch size calculated through grid search. Yet, our $b_{\text{practical}}$ being always larger than the mini-batch obtained by grid search, it leads to a faster algorithm for two reasons. Firstly, the corresponding step size is larger and secondly, and secondly, computing the stochastic gradients in parallel improves the

[8] Our grid is $\{2^i, i = 0, \ldots, 14\}$, with $2^{16}, 2^{18}$ and $n$ being added when needed.

running time. What is even more interesting, is that $b_{\text{practical}}$ always predicts a regime change, where using a larger mini-batch size results in a much larger empirical complexity.

## 2.6 Conclusions

We have explained the crucial role of the expected smoothness constant $\mathcal{L}$ in the convergence of a family of stochastic variance reduced descent algorithms. We have developed functional upper-bounds of this constant to build larger step sizes and closed-form optimal mini-batch values for the $b$-nice SAGA algorithm. Our experiments on artificial and real datasets showed the validity of our upper-bounds and the improvement in the total complexity using our step and optimal mini-batch sizes. Our results suggest a new parameter setting for mini-batch SAGA, that significantly outperforms previous suggested ones, and is even comparable with a grid search approach, without the computational burden of the later.

## 2.7 Technical tools and proofs

### 2.7.1 Linear algebra tools

This section is dedicated to the presentation of useful results to manipulate more easily the smoothness constants. We also provide the proof of the extension of the Bernstein bound for matrix sampled without replacement. These results are separated from the chapter since they are mainly technical tools.

#### 2.7.1.1 Spectral lemmas

Let us recall some useful spectral results on Hermitian and positive semi-definite matrices.

**Lemma 2.21.** *(Weyl's inequality) Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ symmetric matrices. Assume that the eigenvalues of $\mathbf{A}$ (resp. $\mathbf{B}$) are sorted* i.e., *$\lambda_1(\mathbf{A}) \geq \cdots \geq \lambda_n(\mathbf{A})$ (resp. $\lambda_1(\mathbf{B}) \geq \cdots \geq \lambda_n(\mathbf{B})$). Then, we have*

$$\lambda_{i+j-1}(\mathbf{A}+\mathbf{B}) \leq \lambda_i(\mathbf{A}) + \lambda_j(\mathbf{B}) \ . \tag{2.44}$$

*whenever $i, j \geq 1$ and $i + j - 1 \leq n$ .*

Moreover, as a direct consequence of the variational characterization of eigenvalues, namely

$$\lambda_{\max}(\mathbf{A}) = \max_{v\neq 0} \frac{v^\top \mathbf{A} v}{\|v\|_2^2} \ , \tag{2.45}$$

we have an inequality between the maximum diagonal term of a positive semi-definite matrices and its maximum eigenvalue.

**Lemma 2.22.** *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ positive semi-definite matrix and the vector containing its diagonal $d \overset{def}{=} \operatorname{diag}(\mathbf{A})$. Then, we have*

$$\max_{i=1,\ldots,n} d_i \leq \lambda_{\max}(\mathbf{A}) \ . \tag{2.46}$$

The following lemma is a direct consequence of Weyl's inequality for $i = j = 1$.

**Lemma 2.23.** *Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ symmetric matrices. Then, we have*

$$\lambda_{\max}(\mathbf{A}+\mathbf{B}) \leq \lambda_{\max}(\mathbf{A}) + \lambda_{\max}(\mathbf{B}) \ . \tag{2.47}$$

Lastly, we present a result arising from previous lemma.

**Lemma 2.24.** *Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$ symmetric matrices such that $\mathbf{B}$ is positive semi-definite. Then, we have*

$$\lambda_{\max}(\mathbf{A}-\mathbf{B}) \leq \lambda_{\max}(\mathbf{A}) \ . \tag{2.48}$$

*Proof.* Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ symmetric matrices such that $\mathbf{B}$ is positive semi-definite. We get directly

$$\begin{aligned} \lambda_{\max}(\mathbf{A} - \mathbf{B}) &\leq \lambda_{\max}(\mathbf{A}) + \lambda_{\max}(-\mathbf{B}) \\ &= \lambda_{\max}(\mathbf{A}) - \lambda_{\min}(\mathbf{B}) \\ &\leq \lambda_{\max}(\mathbf{A}) \ , \end{aligned}$$

where the first inequality stems from Lemma 2.23 and the second from $\mathbf{B} \succeq 0$. □

### 2.7.2 Matrix Bernstein inequality for sampling without replacement

In this section, we present the matrix Bernstein inequality for independent Hermitian matrices from Tropp et al. [149]. We also provide another version of this theorem for matrices sampled *without* replacement and prove it as explicitly as possible, taking our inspiration from Tropp [147]. The proof is based the possibility of transferring the results from sampling *with* to *without* through the inequality (2.53) due to Gross and Nesme [61]. The exact same work can be done for the tail bound, which is for instance used in Bach [8].

#### 2.7.2.1 Original Bernstein inequality for independent matrices

We first present Theorem 2.25 which gives a Bernstein inequality for a sum of random and independent Hermitian matrices whose eigenvalues are upper bounded. If the matrices $\mathbf{X}_k$ are sampled from a finite set $\mathcal{X}$, one can interpret this random sampling of independent matrices as a random sampling *with* replacement.

**Theorem 2.25** (Tropp et al. [149], Theorem 6.6.1: Matrix Bernstein Inequality)**.** *Consider a finite sequence $\{\mathbf{X}_k\}_{k=1,\dots,n}$ of $n$ independent, random, Hermitian matrices with dimension $d$. Assume that*

$$\mathbb{E}\,\mathbf{X}_k = 0 \quad \text{and} \quad \lambda_{\max}(\mathbf{X}_k) \leq L \quad \textit{for each index} k \ .$$

*Introduce the random matrix*

$$\mathbf{S}_{\mathbf{X}} \overset{def}{=} \sum_{k=1}^{n} \mathbf{X}_k \ .$$

*Let $v(\mathbf{S}_{\mathbf{X}})$ be the matrix variance statistic of the sum:*

$$v(\mathbf{S}_{\mathbf{X}}) \overset{def}{=} \left\| \mathbb{E}\,\mathbf{S}_{\mathbf{X}}^2 \right\| = \left\| \sum_{k=1}^{n} \mathbb{E}\,\mathbf{X}_k^2 \right\| = \lambda_{\max}\left( \sum_{k=1}^{n} \mathbb{E}\,\mathbf{X}_k^2 \right) \ . \tag{2.49}$$

*Then*

$$\mathbb{E}\,\lambda_{\max}(\mathbf{S}_{\mathbf{X}}) \leq \sqrt{2 v(\mathbf{S}_{\mathbf{X}}) \log d} + \frac{1}{3} L \log d \ . \tag{2.50}$$

This theorem is the one we extend in Theorem 2.37 to the case when the random matrices $\mathbf{X}_k$ are sampled *without* replacement from a finite set $\mathcal{X}$. We drew our inspiration from the proof of the matrix Chernoff inequality in Tropp [147] and the one of the matrix Bernstein tail bound in Bach [8], both in the case of sampling without replacement.

#### 2.7.2.2 Technical random matrices prerequisites

Before proving Theorem 2.37, which extends the matrix Bernstein inequality to sampling without replacement, we need to introduce the key tools of the matrix Laplace transform technique. This technique is precious to prove tail bounds for sums of random matrices such as Chernoff, Hoeffding or Bernstein bounds, as presented in [148].

Here, $\|\cdot\|$ denotes the spectral norm, which is defined for any Hermitian matrix $\mathbf{H}$ by

$$||\mathbf{H}|| = \max\left\{ \lambda_{\max}(\mathbf{H}), -\lambda_{\min}(\mathbf{H}) \right\} \ . \tag{2.51}$$

We also introduce the moment generating function (mgf) and the cumulant generating function (cgf) of a random matrix, which are essential in the Laplace transform method approach.

**Definition 2.26** (Matrix Mgf and Cgf)**.** *Let* $\mathbf{X}$ *be a random Hermitian matrix. For all* $\theta \in \mathbb{R}$*, the matrix generating function* $\mathbf{M}_X$ *and the matrix cumulant generating function* $\mathbf{\Xi}_X$ *are given by*

$$\mathbf{M}_X(\theta) \stackrel{def}{=} \mathbb{E}\, e^{\theta \mathbf{X}}$$

*and*

$$\mathbf{\Xi}_X(\theta) \stackrel{def}{=} \log \mathbb{E}\, e^{\theta \mathbf{X}} \ .$$

**Remark 2.27.** *These expectations may not exist for all values of* $\theta$*.*

**Proposition 2.28** (Tropp et al. [149], Proposition 3.2.2: Expectation Bound of the Maximum Eigenvalue)**.** *Let* $\mathbf{X}$ *be a random Hermitian matrix. Then*

$$\mathbb{E}\, \lambda_{\max}(\mathbf{X}) \leq \inf_{\theta > 0} \left\{ \frac{1}{\theta} \log \mathbb{E}\, \mathrm{tr}\, \mathrm{e}^{\theta \mathbf{X}} \right\} \ . \tag{2.52}$$

**Remark 2.29.** *This proposition is an adaptation of the Laplace transform method to obtain a bound of the expectation of the maximum eigenvalue of a random Hermitian matrix. Contrary to the tail bounds, there is no exact analog of the expectation bounds in the scalar setting.*

*Proof of Proposition 2.28.* Fix a positive number $\theta$. Because $\lambda_{\max}(\cdot)$ is a positive-homogeneous map, we have

$$\begin{aligned}
\mathbb{E}\, \lambda_{\max}(\mathbf{X}) &= \frac{1}{\theta} \mathbb{E}\, \lambda_{\max}(\theta \mathbf{X}) \\
&= \frac{1}{\theta} \mathbb{E}\, \log \mathrm{e}^{\lambda_{\max}(\theta \mathbf{X})} \\
&\leq \frac{1}{\theta} \log \mathbb{E}\, \mathrm{e}^{\lambda_{\max}(\theta \mathbf{X})} \\
&= \frac{1}{\theta} \log \mathbb{E}\, \lambda_{\max}\left(\mathrm{e}^{\theta \mathbf{X}}\right) \\
&\leq \frac{1}{\theta} \log \mathbb{E}\, \mathrm{tr}\, \mathrm{e}^{\theta \mathbf{X}} \ ,
\end{aligned}$$

where in the third line we used the Jensen's inequality, in the fourth one the spectral mapping theorem and in the last line the domination by the trace of a positive-definite matrix. □

**Theorem 2.30** (Tropp et al. [149], Theorem 8.1.1: Lieb)**.** *Let* $\mathbf{H}$ *be a fixed Hermitian matrix with dimension* $d$*. The function*

$$\mathbf{X} \to \mathrm{tr} \exp(\mathbf{H} + \mathbf{X})$$

*is a concave map on the the convex cone of* $d \times d$ *positive-definite matrices.*

*Proof of Theorem 2.30.* See Chapter 8 in Tropp et al. [149]. □

**Corollary 2.31.** *Let* $\mathbf{H}$ *be a fixed Hermitian matrix with dimension* $d$*. Let* $\mathbf{X}$ *be a random Hermitian matrix of same dimension. The following inequality holds*

$$\mathbb{E}\, \mathrm{tr} \exp(\mathbf{H} + \mathbf{X}) \leq \mathrm{tr} \exp\left(\mathbf{H} + \log \mathbb{E}\, e^{\mathbf{X}}\right)$$

*is a concave map on the the convex cone of* $d \times d$ *positive-definite matrices.*

*Proof of Corollary 2.31.* Introducing $\mathbf{Y} = e^{\mathbf{X}}$, we have directly

$$\begin{aligned}\mathbb{E}\,\mathrm{tr}\exp\left(\mathbf{H}+\mathbf{X}\right) &= \mathbb{E}\,\mathrm{tr}\exp\left(\mathbf{H}+\log e^{\mathbf{X}}\right)\\ &= \mathbb{E}\,\mathrm{tr}\exp\left(\mathbf{H}+\log\mathbf{Y}\right)\\ &\leq \mathrm{tr}\exp\left(\mathbf{H}+\log\mathbb{E}\,\mathbf{Y}\right)\\ &= \mathrm{tr}\exp\left(\mathbf{H}+\log\mathbb{E}\,e^{\mathbf{X}}\right)\quad .\end{aligned}$$

where the inequality comes from the application of Theorem 2.30 and Jensen's inequality. □

**Lemma 2.32** (Tropp et al. [149], Lemma 3.5.1 or Tropp [148], Lemma 3.4: Subadditivity of Matrix Cgfs)**.** *Consider a finite sequence $\{\mathbf{X}_k\}$ of independent, random, Hermitian matrices of the same dimension. Let $\theta \in \mathbb{R}$, then*

$$\begin{aligned}\mathrm{tr}\exp\left(\boldsymbol{\Xi}_{\sum_k \mathbf{X}_k}(\theta)\right) &= \mathbb{E}\,\mathrm{tr}\exp\left(\theta\sum\nolimits_k \mathbf{X}_k\right)\\ &\leq \mathrm{tr}\exp\left(\sum\nolimits_k \log\mathbb{E}\,\mathrm{e}^{\theta\mathbf{X}_k}\right)\\ &= \mathrm{tr}\exp\left(\sum\nolimits_k \boldsymbol{\Xi}_{\mathbf{X}_k}(\theta)\right)\quad .\end{aligned}$$

*Proof of Lemma 2.32.* Let us assume, without loss of generality, that $\theta = 1$. Let a finite sequence $\{\mathbf{X}_k\}_{k=1}^n$ of $n$ independent, random, Hermitian matrices of the same dimension. We write down $\mathbb{E}_k$ the expectation with respect only to the $k$-th random matrix $\mathbf{X}_k$.

$$\begin{aligned}\mathrm{tr}\exp\left(\boldsymbol{\Xi}_{\sum_{k=1}^n \mathbf{X}_k}(1)\right) &= \mathrm{tr}\exp\left(\log\mathbb{E}\,\exp\left(\sum_{k=1}^n \mathbf{X}_k\right)\right)\\ &= \mathbb{E}\,\mathrm{tr}\exp\left(\sum_{k=1}^n \mathbf{X}_k\right)\\ &= \mathbb{E}_1\ldots\mathbb{E}_{n-1}\mathbb{E}_n\,\mathrm{tr}\exp\left(\sum_{k=1}^{n-1}\mathbf{X}_k+\mathbf{X}_{n+1}\right)\\ &\leq \mathbb{E}_1\ldots\mathbb{E}_{n-1}\,\mathrm{tr}\exp\left(\sum_{k=1}^{n-1}\mathbf{X}_k+\log\mathbb{E}_n\,e^{\mathbf{X}_{n+1}}\right)\\ &= \mathbb{E}_1\ldots\mathbb{E}_{n-1}\,\mathrm{tr}\exp\left(\sum_{k=1}^{n-2}\mathbf{X}_k+\mathbf{X}_{n-1}+\log\mathbb{E}_n\,e^{\mathbf{X}_{n+1}}\right)\\ &\leq \mathbb{E}_1\ldots\mathbb{E}_{n-2}\,\mathrm{tr}\exp\left(\sum_{k=1}^{n-2}\mathbf{X}_k+\log\mathbb{E}_{n-1}\,e^{\mathbf{X}_{n-1}}+\log\mathbb{E}_n\,e^{\mathbf{X}_n}\right)\\ &\leq \cdots \leq \mathrm{tr}\exp\left(\sum\nolimits_k \log\mathbb{E}\,\mathrm{e}^{\theta\mathbf{X}_k}\right)\\ &= \mathrm{tr}\exp\left(\sum_k \boldsymbol{\Xi}_{\mathbf{X}_k}(\theta)\right)\quad .\end{aligned}$$

where first and second inequalities result from Corollary 2.31, the last one comes the fact that $\mathbb{E}_k\,e^{\mathbf{X}_k} = \mathbb{E}\,e^{\mathbf{X}_k}, \forall k \in [n]$ and the final equality directly comes from an identification of Definition 2.26. □

**Lemma 2.33** (Tropp et al. [149], Lemma 6.6.2: Matrix Bernstein Mgf and Cgf Bounds)**.** *Let $\mathbf{X}$ a random Hermitian matrix such that*

$$\mathbb{E}\,\mathbf{X} = 0 \quad \textit{and} \quad \lambda_{\max}\left(\mathbf{X}\right) \leq L\ .$$

*Then, for $0 < \theta < 3/L$,*

$$\mathbf{M}_X(\theta) := \mathbb{E}\,e^{\theta\mathbf{X}} \preceq \exp\left(\frac{\theta^2/2}{1-\theta L/3}\cdot\mathbb{E}\,\mathbf{X}^2\right)$$

*and*

$$\mathbf{\Xi}_X(\theta) := \log \mathbb{E}\, e^{\theta \mathbf{X}} \preceq \frac{\theta^2/2}{1 - \theta L/3} \cdot \mathbb{E}\, \mathbf{X}^2 \ .$$

*Proof of Lemma 2.33.* See Tropp et al. [149]. □

#### 2.7.2.3 Extended results for sampling without replacement

This section is dedicated to the main result, Lemma 2.34, needed for transferring results from sampling *with* to *without* replacement. This lemma is actually the matrix version of a classical result from Hoeffding [66]. We then combine it with previous results of Section 2.7.2.2 to produce a new master bound in Theorem 2.35, which is the key inequality of the proof of Theorem 2.37.

**Lemma 2.34** (Gross and Nesme [61], Domination of the Trace of the Mgf of a Sample Without Replacement)**.** *Consider two finite sequences, of same length $n$, $\{\mathbf{X}_k\}_{k=1,\dots,n}$ and $\{\mathbf{Y}_k\}_{k=1,\dots,n}$ of Hermitian random matrices sampled respectively* with *and* without *replacement from a finite set $\mathcal{X}$. Let $\theta \in \mathbb{R}$, $\mathbf{S_X} := \sum_{k=1}^n \mathbf{X}_k$ and $\mathbf{S_Y} := \sum_{k=1}^n \mathbf{Y}_k$, then*

$$\operatorname{tr} M_{\mathbf{S_Y}}(\theta) := \mathbb{E} \operatorname{tr} \exp(\theta \mathbf{S_Y}) \leq \mathbb{E} \operatorname{tr} \exp(\theta \mathbf{S_X}) \ . \tag{2.53}$$

*Proof of Lemma 2.34.* The left-hand side equality directly arises from Definition 2.26 and the fact that the trace commutes with the expectation because it is a linear operator. For the right-hand side inequality, see the proof in Gross and Nesme [61]. □

**Theorem 2.35** (Master Bound for a Sum of Random Matrices Sampled Without Replacement)**.** *Consider two finite sequences, of same length $n$, $\{\mathbf{X}_k\}_{k=1,\dots,n}$ and $\{\mathbf{Y}_k\}_{k=1,\dots,n}$ of Hermitian random matrices of same size sampled respectively* with *and* without *replacement from a finite set $\mathcal{X}$. Then*

$$\mathbb{E}\, \lambda_{\max}\left(\sum_{k=1}^n \mathbf{Y}_k\right) \leq \inf_{\theta>0}\left\{\frac{1}{\theta}\log \operatorname{tr} \exp\left(\sum_{k=1}^n \log \mathbb{E}\, \mathrm{e}^{\theta \mathbf{X}_k}\right)\right\} \ . \tag{2.54}$$

**Remark 2.36.** *This theorem is a modified version of Theorem 3.6.1 in Tropp et al. [149] for a sum of matrices sampled without replacement.*

*Proof of Theorem 2.35.* Consider two finite sequences, of same length, $\{\mathbf{X}_k\}$ and $\{\mathbf{Y}_k\}$ of Hermitian random matrices of same size sampled respectively *with* and *without* replacement from a finite set $\mathcal{X}$. Let $\theta$ a positive number.

$$\begin{aligned}\mathbb{E}\, \lambda_{\max}\left(\sum_{k=1}^n \mathbf{Y}_k\right) &\leq \inf_{\theta>0}\left\{\frac{1}{\theta}\log \mathbb{E} \operatorname{tr}\, \exp\left(\theta \sum_{k=1}^n \mathbf{Y}_k\right)\right\} \leq \inf_{\theta>0}\left\{\frac{1}{\theta}\log \mathbb{E} \operatorname{tr}\, \exp\left(\theta \sum_{k=1}^n \mathbf{X}_k\right)\right\} \\ &\leq \inf_{\theta>0}\left\{\frac{1}{\theta}\log \operatorname{tr} \exp\left(\sum_{k=1}^n \log \mathbb{E}\, \mathrm{e}^{\theta \mathbf{X}_k}\right)\right\} \ .\end{aligned}$$

where we used successively Proposition 2.28, Lemma 2.34 and Lemma 2.32. First, we use the expectation bound for the maximum eigenvalue. We then use the main result of Gross and Nesme [61] and invoked in Tropp [147] to extend the matrix Chernoff bound for matrices sampled *without* replacement. This lemma allows us to transfer our results to sampling *with* replacement. And finally, we then apply the subadditivity of matrix cgfs to get the desired result. □

#### 2.7.2.4 Bernstein inequality for sampling without replacement

The following theorem is almost the same than Theorem 2.25, but in the case of matrices sampled *without* replacement from a finite set. The proof stems from results established in previous Sections 2.7.2.2 and 2.7.2.3.

**Theorem 2.37** (Matrix Bernstein Inequality Without Replacement)**.** *Let $\mathcal{X}$ be a finite set of Hermitian matrices with dimension $d$ such that*

$$\lambda_{\max}(\mathbf{X}) \leq L, \quad \forall \mathbf{X} \in \mathcal{X} \ .$$

*Sample two finite sequences, of same length $n$, $\{\mathbf{X}_k\}_{k=1,\dots,n}$ and $\{\mathbf{Y}_k\}_{k=1,\dots,n}$ uniformly at random from $\mathcal{X}$ respectively* with *and* without *replacement such that*

$$\mathbb{E}\,\mathbf{X}_k = 0 \quad \forall k \ .$$

*Introduce the random matrices*

$$\mathbf{S}_{\mathbf{X}} \overset{def}{=} \sum_{k=1}^{n} \mathbf{X}_k \quad \textit{and} \quad \mathbf{S}_{\mathbf{Y}} \overset{def}{=} \sum_{k=1}^{n} \mathbf{Y}_k \ .$$

*Let $v(\mathbf{S}_{\mathbf{X}})$ be the matrix variance statistic of the second sum*

$$v(\mathbf{S}_{\mathbf{X}}) \overset{def}{=} \left\|\mathbb{E}\,\mathbf{S}_{\mathbf{X}}^2\right\| = \left\|\sum_{k=1}^{n} \mathbb{E}\,\mathbf{X}_k^2\right\| = \lambda_{\max}\left(\sum_{k=1}^{n} \mathbb{E}\,\mathbf{X}_k^2\right). \tag{2.55}$$

*Then*

$$\mathbb{E}\,\lambda_{\max}(\mathbf{S}_{\mathbf{Y}}) \leq \sqrt{2v(\mathbf{S}_{\mathbf{X}})\log d} + \frac{1}{3}L\log d \ . \tag{2.56}$$

*Proof of Theorem 2.37.* Consider $\mathcal{X}$ a finite set of Hermitian matrices of dimension $d$ such that

$$\lambda_{\max}(\mathbf{X}) \leq L \quad \forall \mathbf{X} \in \mathcal{X} \ .$$

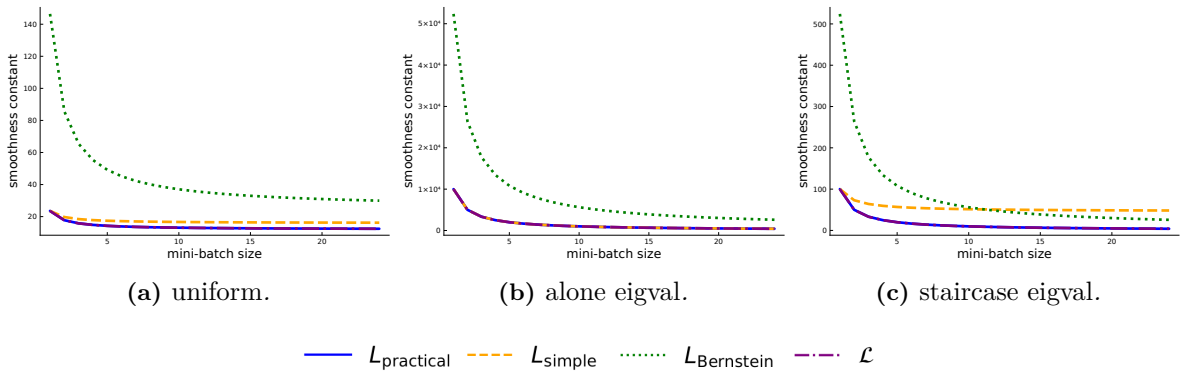
Sample two finite sequences, of same length, $\{\mathbf{X}_k\}$ and $\{\mathbf{Y}_k\}$ uniformly at random from $\mathcal{X}$ respectively *with* and *without* replacement such that

$$\mathbb{E}\,\mathbf{X}_k = 0 \quad \forall k \ .$$

The $\{\mathbf{X}_k\}$ matrices are thus independent. Introduce the sums $\mathbf{S}_{\mathbf{X}} = \sum_{k=1}^{n} \mathbf{X}_k$ and $\mathbf{S}_{\mathbf{Y}} = \sum_{k=1}^{n} \mathbf{Y}_k$. Let us bound the expectation of the largest eigenvalue of the latter

$$\begin{aligned}
\mathbb{E}\,\lambda_{\max}(\mathbf{S}_{\mathbf{Y}}) = \mathbb{E}\,\lambda_{\max}\left(\sum_{k=1}^{n} \mathbf{Y}_k\right) &\leq \inf_{\theta>0}\left\{\frac{1}{\theta}\log\operatorname{tr}\exp\left(\sum_{k=1}^{n}\log\mathbb{E}\,\mathrm{e}^{\theta\mathbf{X}_k}\right)\right\} \\
&\leq \inf_{0<\theta<3/L}\left\{\frac{1}{\theta}\log\operatorname{tr}\exp\left(\frac{\theta^2/2}{1-\theta L/3}\sum_{k=1}^{n}\mathbb{E}\,\mathbf{X}_k^2\right)\right\} \\
&\leq \inf_{0<\theta<3/L}\left\{\frac{1}{\theta}\log\left[d\,\lambda_{\max}\left(\exp\left(\frac{\theta^2/2}{1-\theta L/3}\mathbb{E}\,\mathbf{S}_{\mathbf{X}}^2\right)\right)\right]\right\} \\
&\leq \inf_{0<\theta<3/L}\left\{\frac{1}{\theta}\log\left[d\,\exp\left(\frac{\theta^2/2}{1-\theta L/3}\lambda_{\max}\left(\mathbb{E}\,\mathbf{S}_{\mathbf{X}}^2\right)\right)\right]\right\} \\
&\leq \inf_{0<\theta<3/L}\left\{\frac{1}{\theta}\log\left[d\,\exp\left(\frac{\theta^2/2}{1-\theta L/3}v(\mathbf{S}_{\mathbf{X}})\right)\right]\right\} \\
&= \inf_{0<\theta<3/L}\left\{\frac{\log d}{\theta} + \frac{\theta/2}{1-\theta L/3}v(\mathbf{S}_{\mathbf{X}})\right\} \ .
\end{aligned}$$

where the inequalities successively derive from Theorem 2.35, Lemma 2.33 combined with the monotony of $\operatorname{tr}\exp(\cdot)$, the fact that $\operatorname{tr}(\mathbf{M}) \leq d\,\lambda_{\max}(\mathbf{M})$, $\forall \mathbf{M} \in \mathbb{R}^{d\times d}$, the spectral mapping theorem and lastly (2.51) with $\mathbb{E}\,\mathbf{Y}^2 \succeq 0$. Finally, one can complete the infimum, for instance using a computer algebra




**(a)** uniform. **(b)** alone eigval. **(c)** staircase eigval.

**Figure 2.6:** *Expected smoothness constant $\mathcal{L}$ and its upper-bounds the mini-batch size $b$ varies (unscaled datasets, $\lambda = 10^{-1}$).*

system, to finish the proof as it was stated in the original proof by Tropp et al. [149] [9]. In conclusion,

$$\mathbb{E}\,\lambda_{\max}(\mathbf{S_Y}) \leq \sqrt{2v(\mathbf{S_X})\log d} + \frac{1}{3}L\log d \ .$$

□

## 2.8 Additional experiments

### 2.8.1 Estimates of the expected smoothness constant for artificial datasets

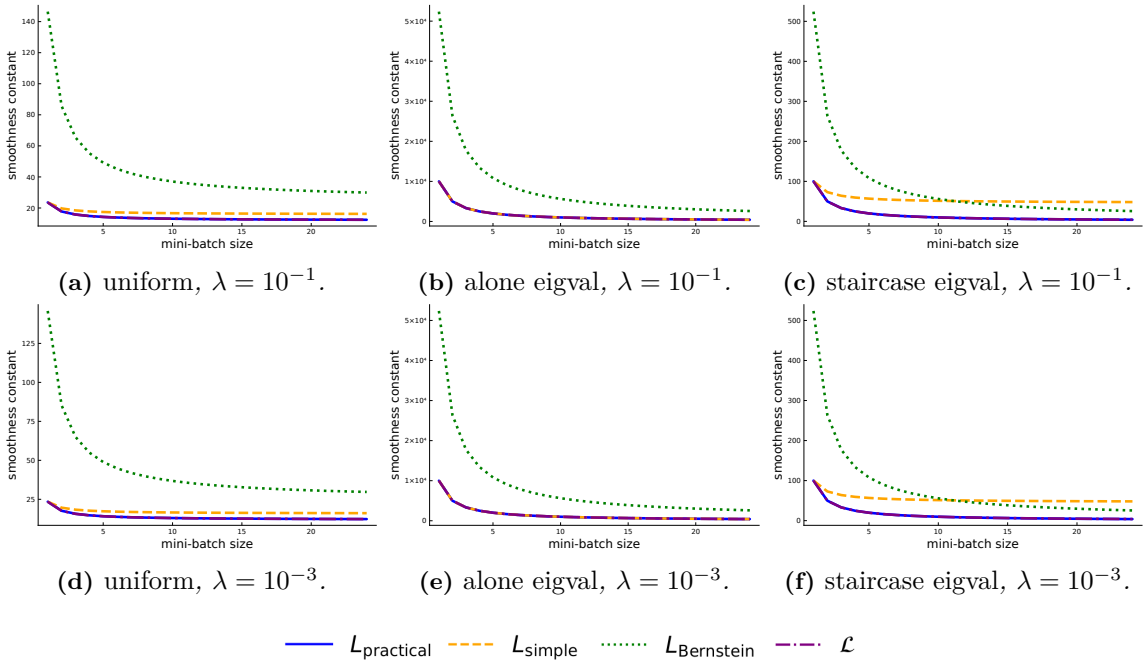
As described in Section 2.5, we compute our the *practical*, *simple* and *Bernstein bounds* and the true $\mathcal{L}$ for ridge regression applied to small artificial datasets: *uniform* ($n = 24, d = 50$), *staircase eigval* ($n = d = 24$) and *alone eigval* ($n = d = 24$). Figure 2.6 shows first that the *practical bound* is a very close approximation of $\mathcal{L}$. On the one hand, we observe in Figure 2.6a that the *Bernstein bound* performs poorly since the feature dimension is very small $d = 50$. On the other hand, Figure 2.6c shows a regime change for $b \approx 10$, which highlight the usefulness of combining our bounds to approximate the expected smoothness constant. Finally, we observe that for the *alone eigval* dataset Figure 2.6b, which has one very large eigenvalue far from the rest of the spectrum, the *simple bound* matches $\mathcal{L}$ because the gap between $\bar{L}$ and $L$ shrinks. Indeed, in this configuration $\bar{L} \approx L \approx L_{\max}/n$. When the spectrum is more concentrated, like for *staircase eigval*, we get a significant gap between $\bar{L}$ and $L$ as shown in Figure 2.6c, where the *simple bound* is far from $\mathcal{L}$ when $b = n$.

We also report the influence of changing the value of the regularization parameter $\lambda$. Figure 2.7 shows that this parameter has little impact on the general shape of the bounds and of $\mathcal{L}$.

Finally, we study the impact of scaling or standardizing (*i.e.,* removing the mean and dividing by the standard deviation for each feature) our artificial datasets. In order not to benefit from the diagonal shape of the *alone eigval* and *staircase eigval* datasets we also give examples of the bounds of $\mathcal{L}$ after a rotation of the data. The rotation aims at preserving the spectrum while erasing the diagonal structure of the covariance matrix $\mathbf{X}^\top\mathbf{X}$. This rotation procedure consists in transforming $\mathbf{X}$ into $\mathbf{Q}^\top\mathbf{X}\mathbf{Q}$, where $\mathbf{Q}$ is the orthogonal matrix given by the QR decomposition of a random squared matrix (with dimension the same as the one of $\mathbf{X}^\top$) with uniformly random coefficients $\mathbf{M}$, such that $\mathbf{M} = \mathbf{Q}\mathbf{R}$.

We observe in Figure 2.8 that rotations do not affect our estimates of $\mathcal{L}$, because they preserve the spectrum. Scaling non-diagonal datasets does not change the general shape neither. As predicted, scaling diagonal matrices leads to a particular case where the spectrum of the covariance matrix is flattened

[9] For instance : `Minimize[(log(d)/x) + ((x/2)/(1-(L/3)*x))*v, x >0, x < (3/L), x]` in Wolfram Alpha.

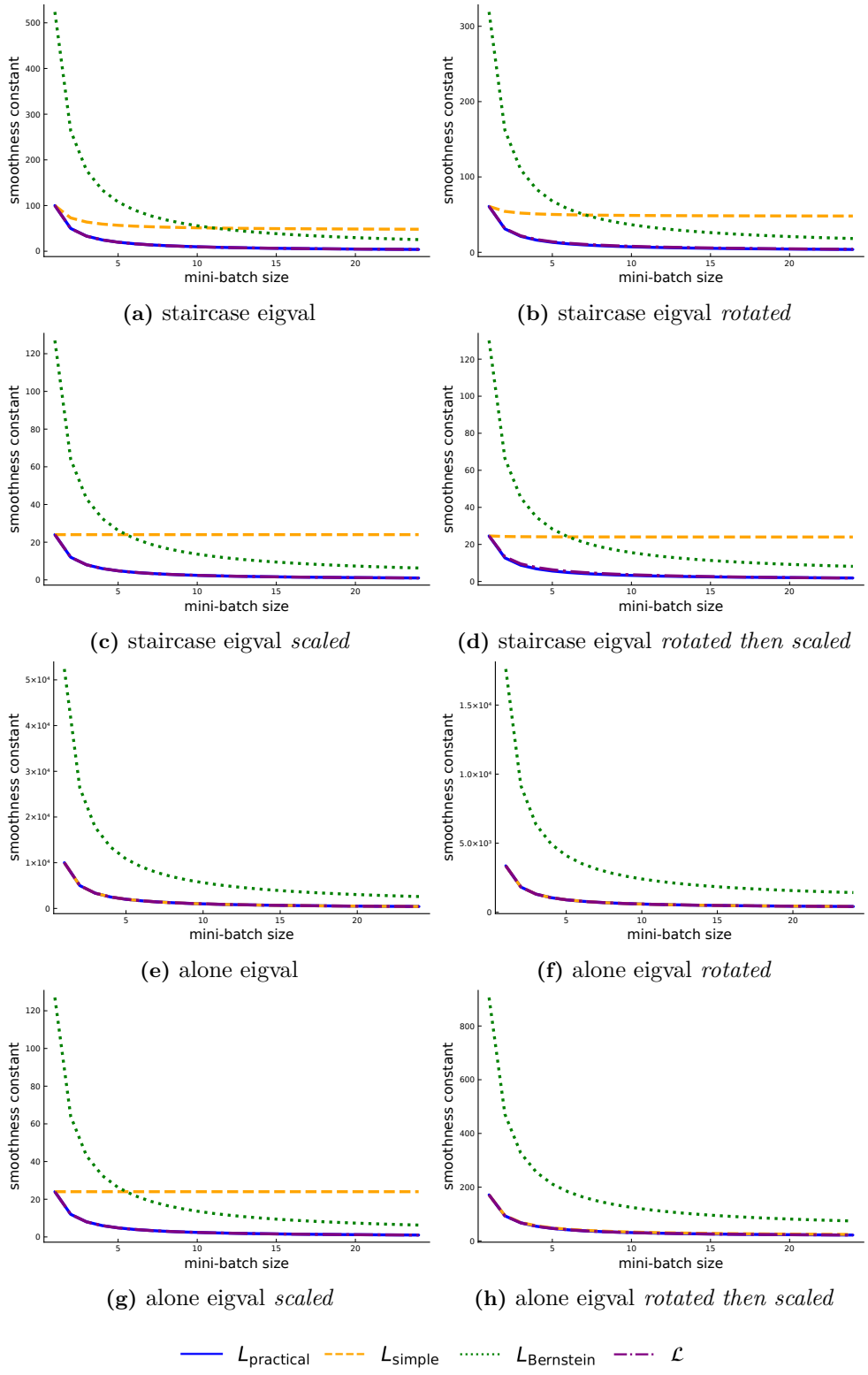



**(a)** uniform, $\lambda = 10^{-1}$. **(b)** alone eigval, $\lambda = 10^{-1}$. **(c)** staircase eigval, $\lambda = 10^{-1}$.

**(d)** uniform, $\lambda = 10^{-3}$. **(e)** alone eigval, $\lambda = 10^{-3}$. **(f)** staircase eigval, $\lambda = 10^{-3}$.

**Figure 2.7:** *Expected smoothness constant $\mathcal{L}$ and its upper-bounds as a function of the mini-batch size for unscaled datasets with $\lambda = 10^{-1}$ (top) and $\lambda = 10^{-3}$ (bottom).*

and for all $i \in [n]$, $L_i \approx L_{\max} \approx \bar{L}$. This is why we get a flat *simple bound* in Figures 2.8c and 2.8g. Even after those different types of preprocessing (rotation and scaling) and with different values of $\lambda$, we end up with the same strong observation that the *practical bound* is a very sharp approximation of the expected smoothness constant.




**(a)** staircase eigval **(b)** staircase eigval *rotated*

**(c)** staircase eigval *scaled* **(d)** staircase eigval *rotated then scaled*

**(e)** alone eigval **(f)** alone eigval *rotated*

**(g)** alone eigval *scaled* **(h)** alone eigval *rotated then scaled*

**Figure 2.8:** *Upper-bounds of the expected smoothness constant $\mathcal{L}$ for non-rotated (left) and rotated (right) datasets ($\lambda = 10^{-3}$).*

### 2.8.2 Estimates of the expected smoothness constant for real datasets

In what follows, we also used publicly available datasets from LIBSVM[10] provided by Chang and Lin [26] and from the UCI repository[11] provided by Dua and Graff [39]. We applied ridge regression to the following datasets: *YearPredictionMSD* ($n = 515,345, d = 90$) from LIBSVM and *slice* ($n = 53,500, d = 384$) from UCI. We also applied regularized logistic regression for binary classification on *ijcnn1* ($n = 141,691, d = 22$), *covtype.binary* ($n = 581,012, d = 54$), *real-sim* ($n = 72,309, d = 20,958$), *rcv1.binary* ($n = 697,641, d = 47,236$) and *news20.binary* ($n = 19,996, d = 1,355,191$) from LIBSVM. When a test set was available, we concatenated it with the train set to have more samples.

From Figure 2.9, we observe that after feature-scaling, the *Bernstein bound* is always a tighter upper bound of $\mathcal{L}$ than the *simple bound*. One can observe in Figure 2.10, that for unscaled datasets the *Bernstein bound* performs better than the *simple bound*, except for *YearPredictionMSD* ($n = 515,345, d = 90$) and *covtype.binary* ($n = 581,012, d = 54$).

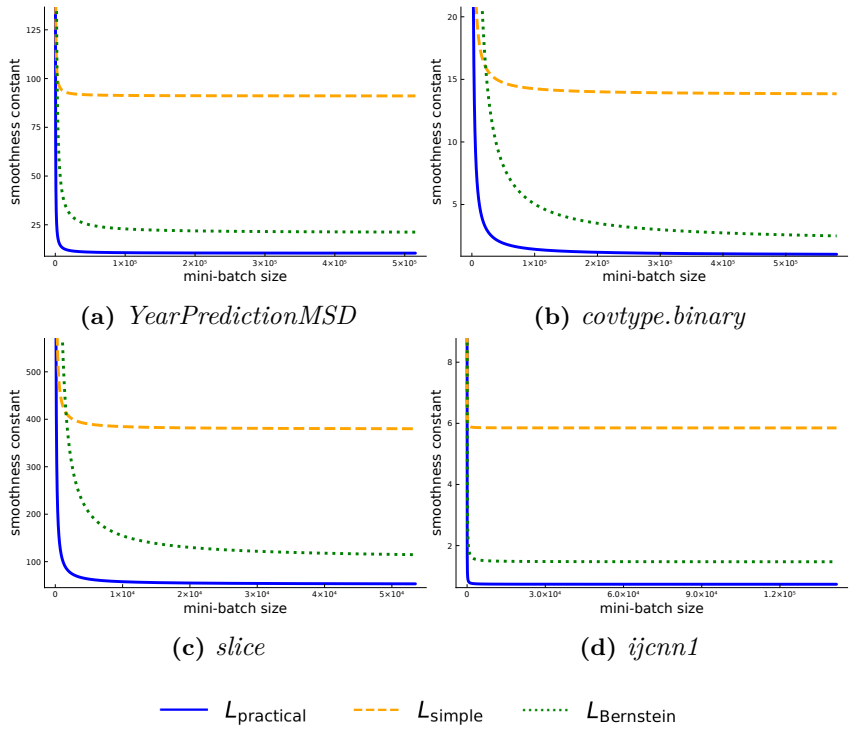



**Figure 2.9:** *Upper-bounds of the expected smoothness constant of $\mathcal{L}$ for real feature-scaled datasets ($\lambda = 10^{-1}$).*

[10] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`
[11] `https://archive.ics.uci.edu/ml/datasets/`

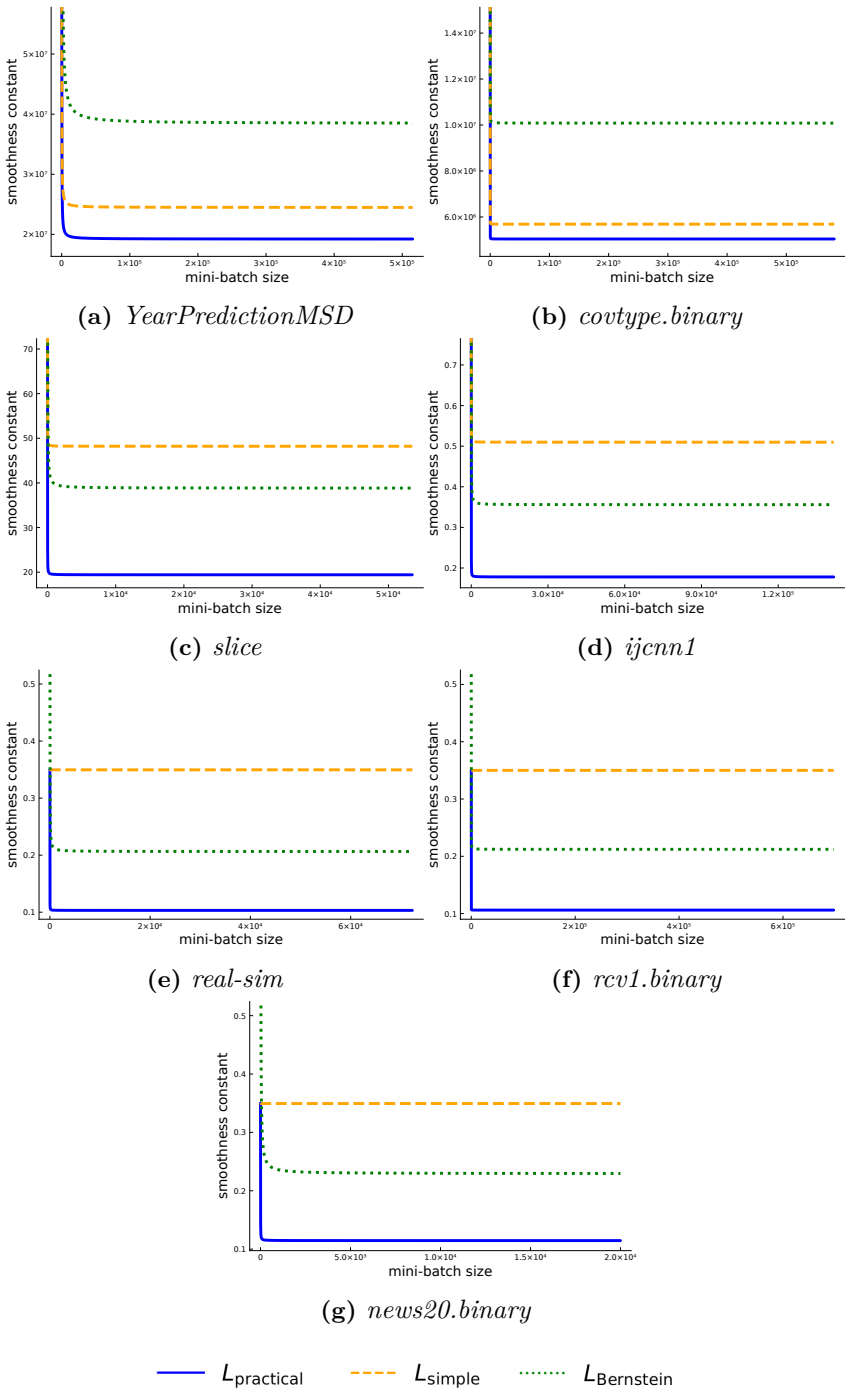
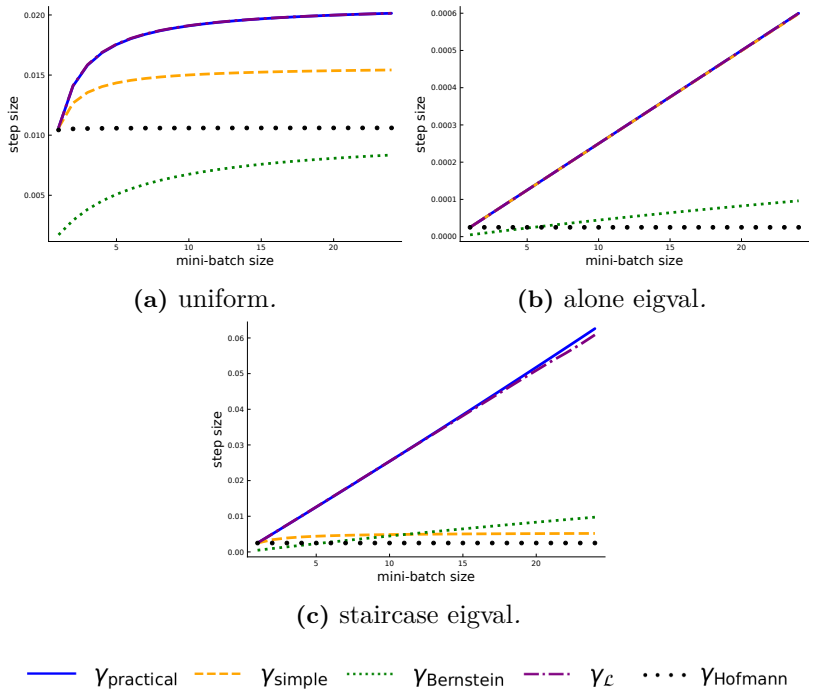



**Figure 2.10:** *Upper-bounds of the expected smoothness constant for real unscaled datasets ($\lambda = 10^{-1}$).*

### 2.8.3 Step size estimates for artificial datasets

In this section we give the step sizes estimate corresponding to the expected smoothness constant, the *practical*, *simple* and *Bernstein* upper-bounds for our small artificial datasets. In Figure 2.11, we show that the *practical* step size $\gamma_{\text{practical}}$ is larger than all others. Moreover, for except for small value sof $b$, our $\gamma_{\text{simple}}$ or $\gamma_{\text{Bernstein}}$ estimates are in most cases larger than the one proposed in [68].




**(a)** uniform. **(b)** alone eigval.

**(c)** staircase eigval.

**Figure 2.11:** *Step size estimates as a function the mini-batch size for unscaled artificial datasets ($\lambda = 10^{-1}$).*

### 2.8.4 Step size estimates for real datasets

Here we show the step sizes estimate corresponding to the *practical*, *simple* and *Bernstein* upper-bounds for real datasets detailed in Section 2.8.2. On these real data, unscaled in Figure 2.12 and scaled in Figure 2.13, we see that the gap between our step size estimates and $\gamma_{\text{Hofmann}}$ is even larger. We observe in Figure 2.12, accordingly to previous remarks in Section 2.8.2, that *Bernstein bound* leads to larger step sizes than the *simple* one, except for the unscaled *YearPredictionMSD* and *covtype.binary* datasets. Yet, as noticed before, Figure 2.13 seems to show that scaling the data leads to $\gamma_{\text{Bernstein}}$ larger than $\gamma_{\text{simple}}$.

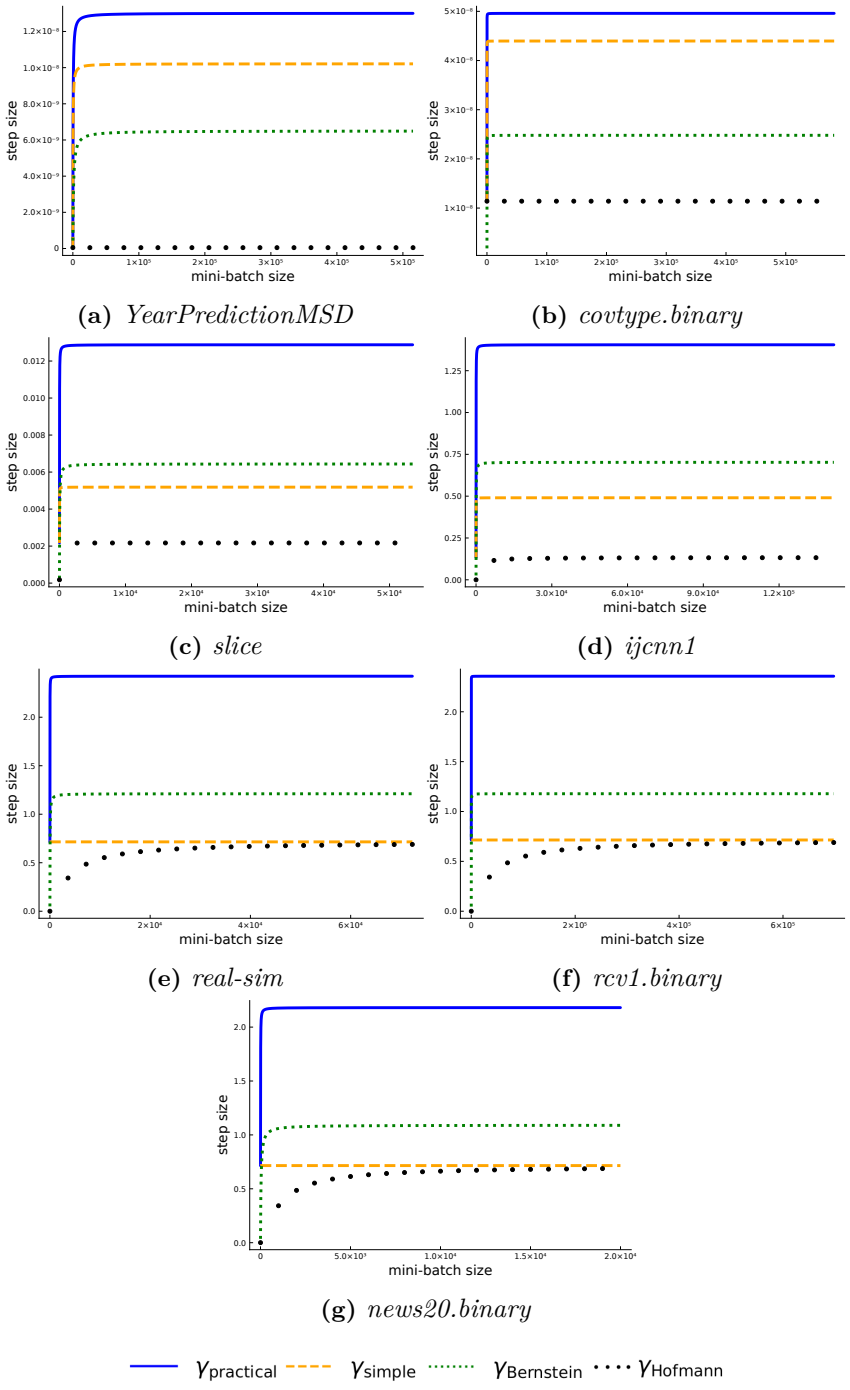



**Figure 2.12:** *Step size estimates as a function the mini-batch size for real unscaled datasets ($\lambda = 10^{-1}$).*

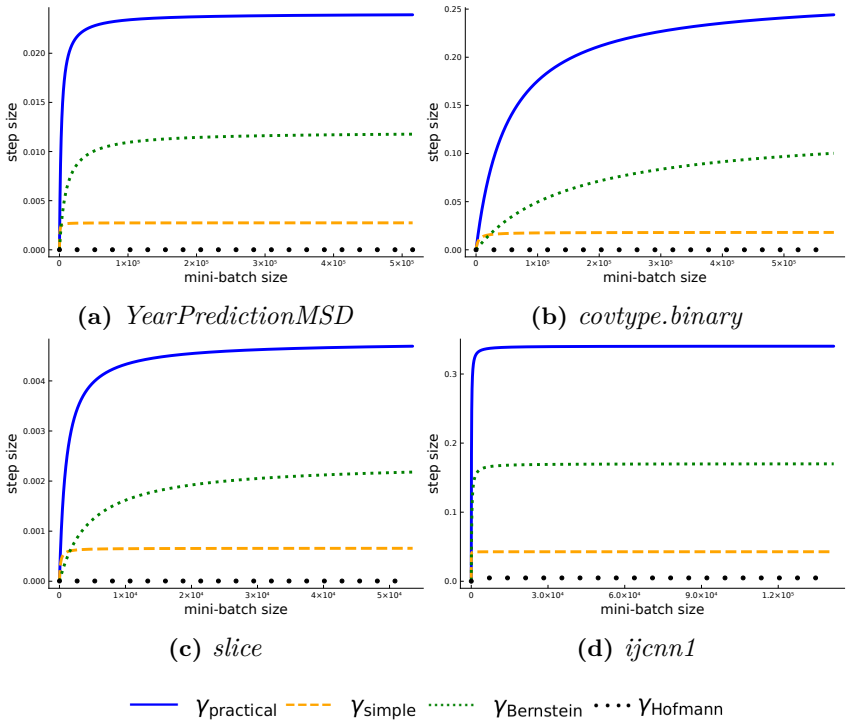



**Figure 2.13:** *Step size estimates as a function the mini-batch size for real feature-scaled datasets* ($\lambda = 10^{-1}$).

### 2.8.5 Comparison with previous SAGA settings

In this section we provide more example of the performance of our practical settings compared to previously known SAGA settings. In Figures 2.15 to 2.20 we run our experiments on real datasets introduced in detail in Section 2.8.1. SAGA implementations are run until the suboptimality reaches a relative error $(f(w^k) - f(w^*))/(f(w^0) - f(w^*))$ of $10^{-4}$, except in some cases where the Hofmann's runs exceeded our maximal number of epochs like in Figure 2.16. In Figure 2.18, the curves corresponding to Hofmann's settings are not displayed because they achieve a total complexity which is too large. Figure 2.14 shows an example of such a configuration.

These experiments show that our settings $(b_{\text{practical}}, \gamma_{\text{practical}})$ most of the time outperforms whether the classical $(b = 1, \gamma_{\text{Defazio}})$ or the $(b = 20, \gamma_{\text{Hofmann}})$ settings both in terms of epochs and running time.

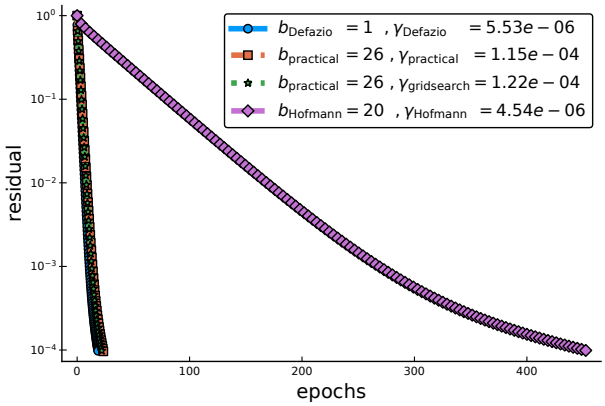
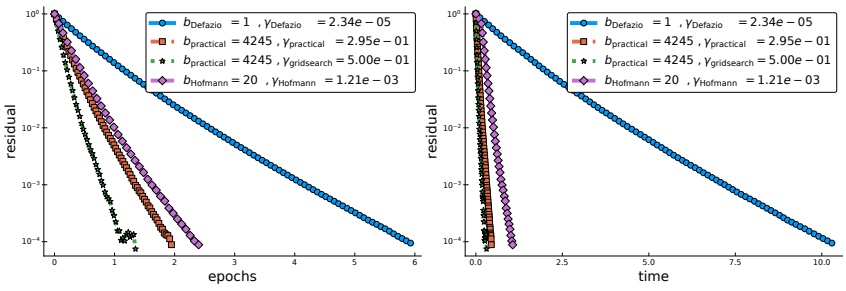
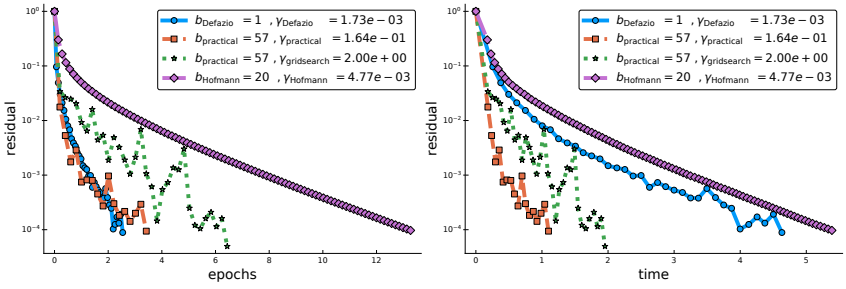



**Figure 2.14:** *Poor performance of Hofmann's settings for the feature-scaled dataset slice ($\lambda = 10^{-1}$).*

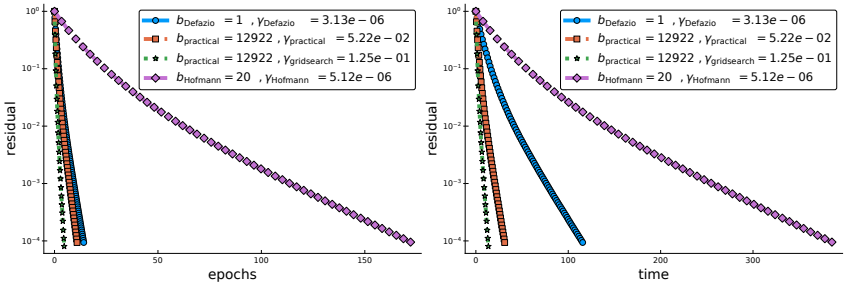



**(a)** $\lambda = 10^{-1}$

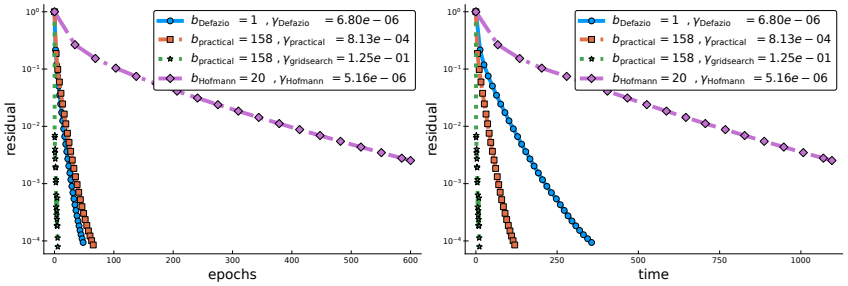



**(b)** $\lambda = 10^{-3}$

**Figure 2.15:** *Performance of SAGA implementations for the feature-scaled dataset ijcnn1.*

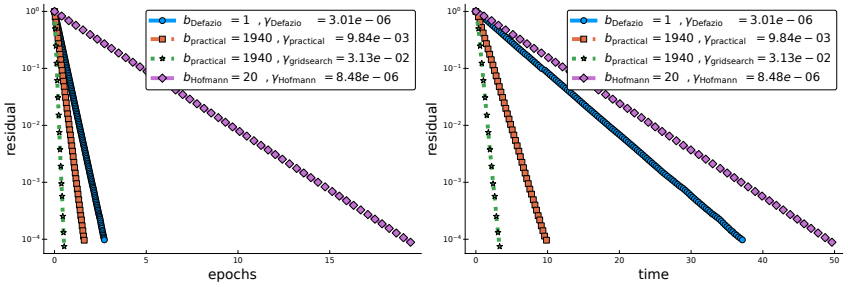



**(a)** $\lambda = 10^{-1}$

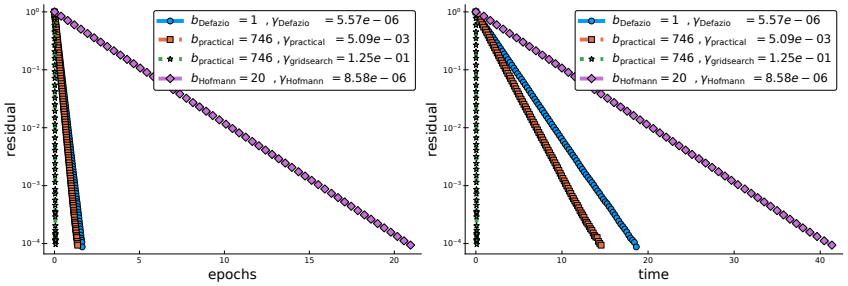



**(b)** $\lambda = 10^{-3}$

**Figure 2.16:** *Performance of SAGA implementations for the feature-scaled dataset covtype.binary.*

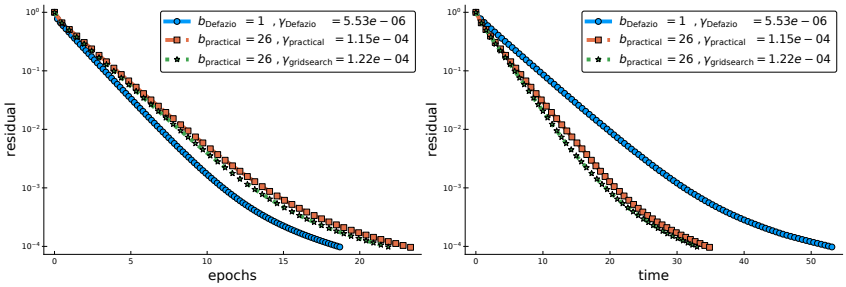



**(a)** $\lambda = 10^{-1}$

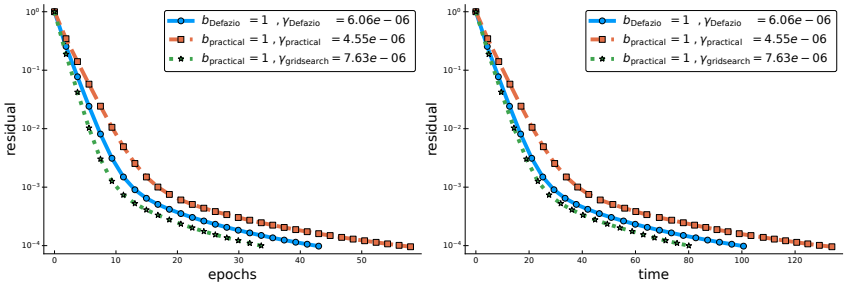



**(b)** $\lambda = 10^{-3}$

**Figure 2.17:** *Performance of SAGA implementations for the feature-scaled dataset YearPredictionMSD.*

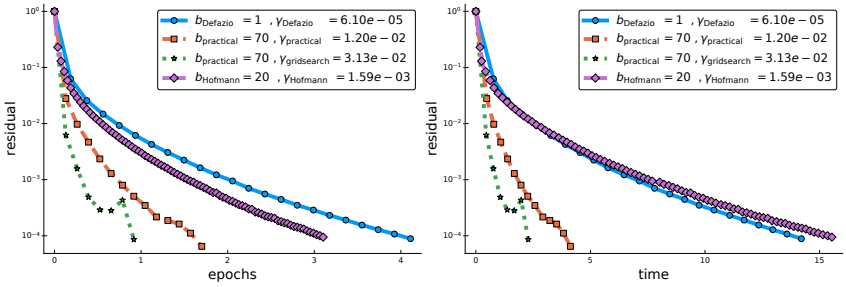



**(a)** $\lambda = 10^{-1}$

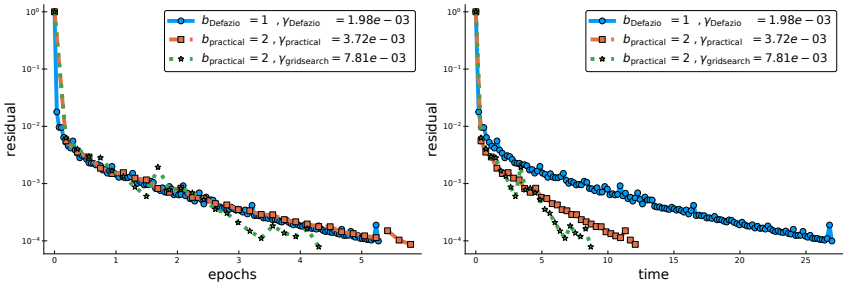



**(b)** $\lambda = 10^{-3}$

**Figure 2.18:** *Performance of SAGA implementations for the feature-scaled dataset slice.*

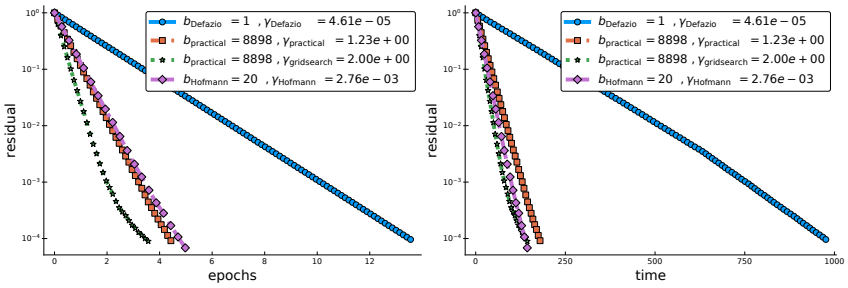



**(a)** $\lambda = 10^{-1}$

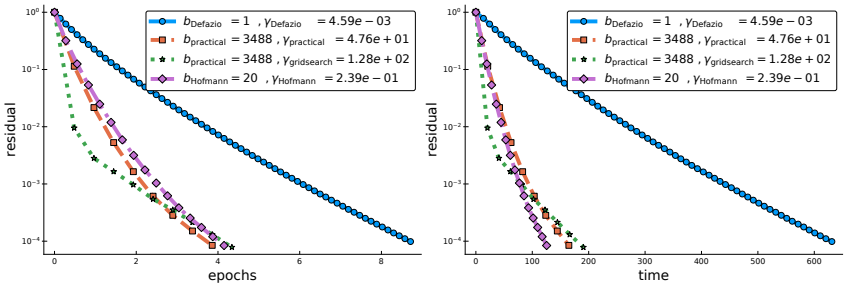



**(b)** $\lambda = 10^{-3}$

**Figure 2.19:** *Performance of SAGA implementations for the unscaled dataset slice.*




**(a)** $\lambda = 10^{-1}$



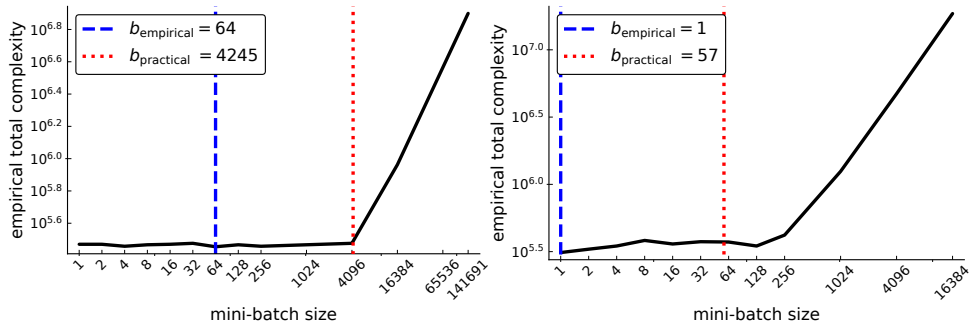

**(b)** $\lambda = 10^{-3}$

**Figure 2.20:** *Performance of SAGA implementations for the unscaled dataset real-sim.*

### 2.8.6 Optimality of the mini-batch size

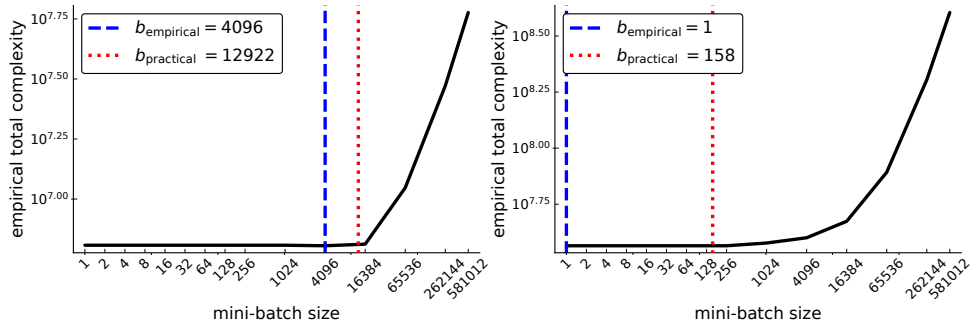
This experiment aims to estimate how close is our practical bound $b_{\text{practical}}$ to the empirical best mini-batch size one could get running a grid search. We recall that we use the following grid for the mini-batch sizes: $\{2^i, i = 0, \dots, 14\}$, with $2^{16}, 2^{18}$ and $n$ added in some cases. We show in the log-scaled Figures 2.21 to 2.26 the empirical total complexity $K_{\text{total}}$, *e.g.,* the number of computed stochastic gradients to reach a relative error of $10^{-4}$, as a function of the mini-batch $b$. For each mini-batch of the grid $b$, the step size used is the one corresponding to the *practical bound*, *i.e.,* when replacing $\mathcal{L}(b)$ by $\mathcal{L}_{\text{practical}}(b)$ in (2.17).

We always observe a change of regime in the empirical complexity. For small values of $b$, the complexity is of the same order of magnitude, then, for values greater than the empirical optimal mini-batch size, the complexity explodes. This experiment shows that our optimal mini-batch size $b_{\text{practical}}$ correctly designates the largest mini-batch achieving the best complexity as large as possible, without reaching the regime where the total complexity explodes.



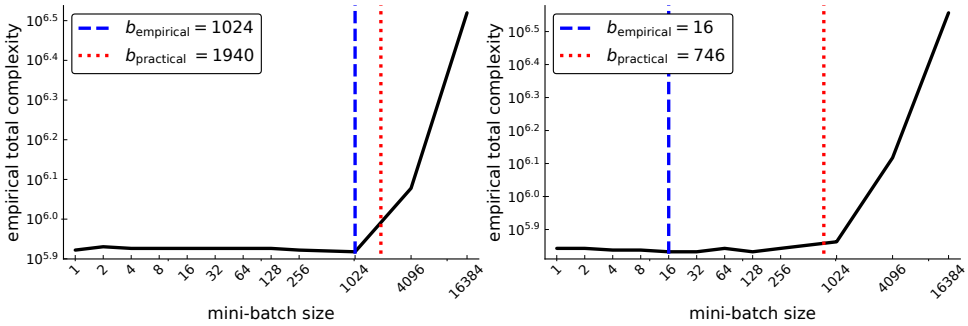

**(a)** $\lambda = 10^{-1}$ **(b)** $\lambda = 10^{-3}$

**Figure 2.21:** *Empirical total complexity versus mini-batch size for the feature-scaled* ijcnn1 *dataset.*



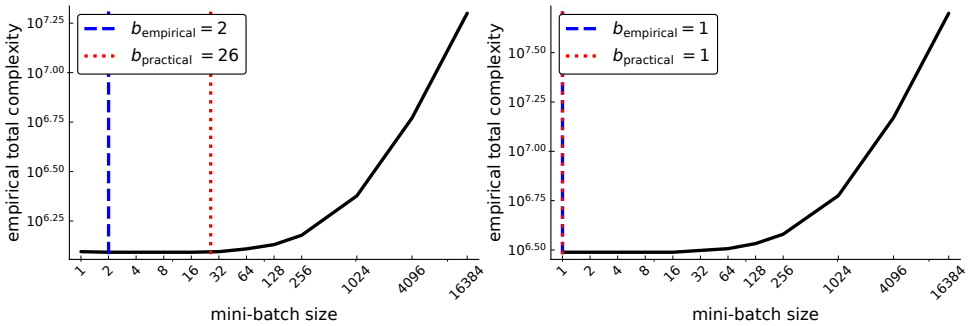

**(a)** $\lambda = 10^{-1}$ **(b)** $\lambda = 10^{-3}$

**Figure 2.22:** *Empirical total complexity versus mini-batch size for the feature-scaled* covtype.binary *dataset.*



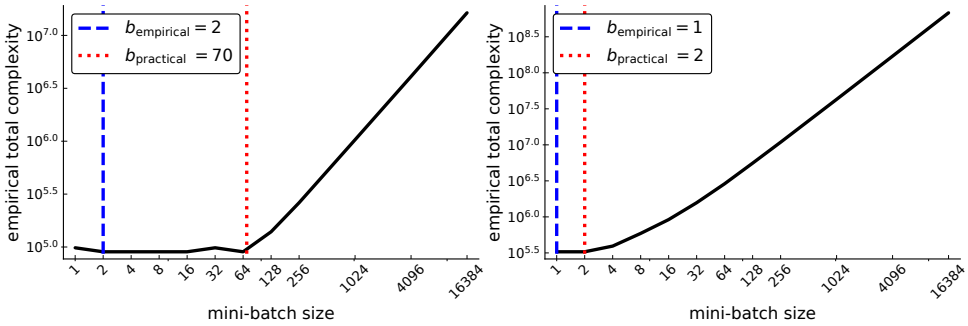

**(a)** $\lambda = 10^{-1}$ **(b)** $\lambda = 10^{-3}$

**Figure 2.23:** *Empirical total complexity versus mini-batch size for the feature-scaled* YearPredictionMSD *dataset.*



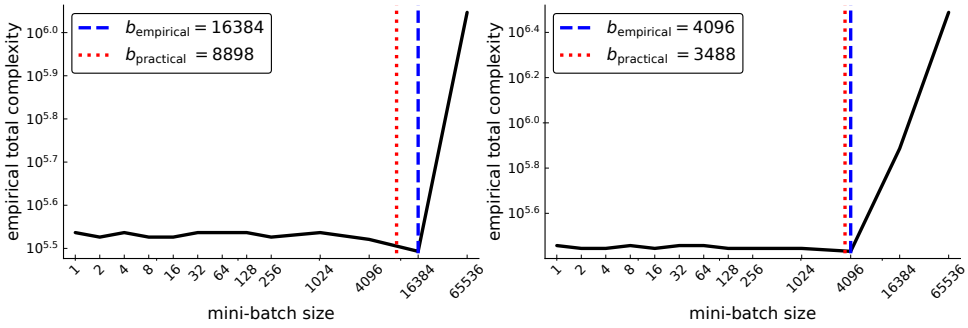

**(a)** $\lambda = 10^{-1}$ **(b)** $\lambda = 10^{-3}$

**Figure 2.24:** *Empirical total complexity versus mini-batch size for the feature-scaled* slice *dataset.*




**(a)** $\lambda = 10^{-1}$ **(b)** $\lambda = 10^{-3}$

**Figure 2.25:** *Empirical total complexity versus mini-batch size for the unscaled* slice *dataset.*




**(a)** $\lambda = 10^{-1}$ **(b)** $\lambda = 10^{-3}$

**Figure 2.26:** *Empirical total complexity versus mini-batch size for the unscaled* real-sim *dataset.*

CHAPTER 3

# Towards closing the gap between the theory and practice of SVRG

- Monsieur, répliqua Kovaliov d'un ton fort digne, je ne sais quel sens donner à vos paroles... L'affaire est pourtant bien claire... Enfin, monsieur, n'êtes-vous pas mon propre nez ?
Le Nez considéra le major avec un léger froncement de sourcils.
- Vous vous trompez, monsieur, je n'appartiens qu'à moi-même.

N. Gogol (1836). Le Nez

Among the very first variance reduced stochastic methods for solving the empirical risk minimization problem was the SVRG method [71]. SVRG is an inner-outer loop based method, where in the outer loop a reference full gradient is evaluated, after which $m \in \mathbb{N}$ steps of an inner loop are executed where the reference gradient is used to build a variance reduced estimate of the current gradient. The simplicity of the SVRG method and its analysis have led to multiple extensions and variants for even non-convex optimization. We provide a more general analysis of SVRG than had been previously done by using arbitrary sampling, which allows us to analyse virtually all forms of mini-batching through a single theorem. Furthermore, our analysis is focused on more practical variants of SVRG including a new variant of the loopless SVRG [68, 81, 83] and a variant of k-SVRG [121] where $m = n$ and where $n$ is the number of data points. Since our setup and analysis reflect what is done in practice, we are able to set the parameters such as the mini-batch size and step size using our theory in such a way that produces a more efficient algorithm in practice, as we show in extensive numerical experiments.

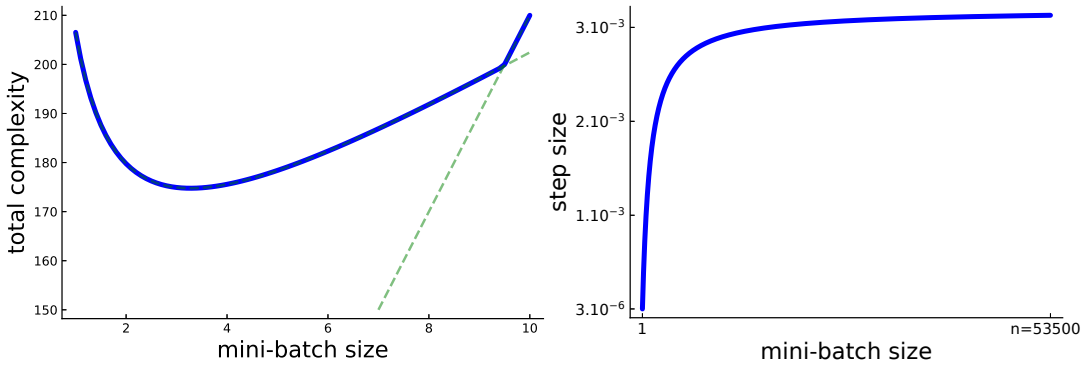
## 3.1 Introduction

Consider the problem of minimizing a $\mu$–strongly convex and $L$–smooth function $f$ where

$$w^* = \arg\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(w) =: f(w), \tag{3.1}$$

and each $f_i$ is convex and $L_i$–smooth. Several training problems in machine learning fit this format, e.g. least-squares, logistic regressions and conditional random fields. Typically each $f_i$ represents a regularized loss of an $i$th data point. When $n$ is large, algorithms that rely on full passes over the data, such as gradient descent, are no longer competitive. Instead, the stochastic version of gradient descent SGD [127] is often used since it requires only a mini-batch of data to make progress towards the solution. However, SGD suffers from high variance, which keeps the algorithm from converging unless a carefully often hand-tuned decreasing sequence of step sizes is chosen. This often results in a cumbersome parameter tuning and a slow convergence.

To address this issue, many variance reduced methods have been designed in recent years including




**(a)** *Total complexity* **(b)** *Step size*

**Figure 3.1:** *The total complexity* (3.3) *and the step size* (3.4) *as $b$ increases for random Gaussian data.*

SAG [129, 133], SAGA [33] and SDCA [139] that require only a constant step size to achieve linear convergence. In this chapter, we are interested in variance reduced methods with an inner-outer loop structure, such as S2GD [80], nguyen2017sarah [109], L-SVRG [81] and the orignal SVRG [71] algorithm. Here we present not only a more general analysis that allows for any mini-batching strategy, but also a more *practical* analysis, by analysing methods that are based on what works in practice, and thus providing an analysis that can inform practice.

## 3.2 Background and contributions

**Convergence under arbitrary samplings.** We give the first arbitrary sampling convergence results for SVRG type methods in the convex setting[1]. That is our analysis includes all forms of sampling including mini-batching and importance sampling as a special case. To better understand the significance of this result, we use mini-batching $b$ elements *without* replacement as a running example throughout the chapter. With this sampling the update step of SVRG, starting from $w^0 = z_0 \in \mathbb{R}^d$, takes the form of

$$w^{t+1} = w^t - \alpha \left( \frac{1}{b} \sum_{i \in B} \nabla f_i(w^t) - \frac{1}{b} \sum_{i \in B} \nabla f_i(z_{s-1}) + \nabla f(z_{s-1}) \right), \tag{3.2}$$

where $\alpha > 0$ is the step size, $B \subseteq [n] \overset{\text{def}}{=} \{1, \dots, n\}$ and $b = |B|$. Here $z_{s-1}$ is the *reference point* which is updated after $m \in \mathbb{N}$ steps, the $w^t$'s are the *inner iterates* and $m$ is the loop length. As a special case of our forthcoming analysis in Corollary 3.28, we show that the *total complexity* of the SVRG method based on (3.2) to reach an $\epsilon > 0$ accurate solution has a simple expression which depends on $n$, $m$, $b$, $\mu$, $L$ and $L_{\max} \overset{\text{def}}{=} \max_{i \in [n]} L_i$:

$$C_m(b) \overset{\text{def}}{=} 2\left(\frac{n}{m} + 2b\right) \max\left\{ \frac{3}{b} \frac{n-b}{n-1} \frac{L_{\max}}{\mu} + \frac{n}{b} \frac{b-1}{n-1} \frac{L}{\mu}, m \right\} \log\left(\frac{1}{\epsilon}\right), \tag{3.3}$$

so long as the step size is

$$\alpha = \frac{1}{2} \frac{b(n-1)}{3(n-b)L_{\max} + n(b-1)L}. \tag{3.4}$$

By total complexity we mean the total number of individual $\nabla f_i$ gradients evaluated. This shows that the total complexity is a simple function of the loop length $m$ and the mini-batch size $b$. See Figure 3.1 for an example for how total complexity evolves as we increase the mini-batch size.

**Optimal mini-batch and step sizes for SVRG.** The size of the mini-batch $b$ is often left as a parameter for the user to choose or set using a rule of thumb. The current analysis in the literature

[1] SVRG has very recently been analysed under arbitrary samplings in the non-convex setting [69].

| $n \leq \frac{L}{\mu}$ | | $\frac{L}{\mu} < n < \frac{3L_{\max}}{\mu}$ | | $n \geq \frac{3L_{\max}}{\mu}$ |
|---|---|---|---|---|
| $L_{\max} \geq \frac{nL}{3}$ | $L_{\max} < \frac{nL}{3}$ | $L_{\max} \geq \frac{nL}{3}$ | $L_{\max} < \frac{nL}{3}$ | |
| $n$ | $\lfloor \hat{b} \rfloor$ | $\lfloor \tilde{b} \rfloor$ | $\lfloor \min\{\hat{b}, \tilde{b}\} \rfloor$ | 1 |

**Table 3.1:** *Optimal mini-batch sizes for Algorithm 9 with $m = n$. The last line presents the optimal mini-batch sizes depending on all the possible problem settings, which are presented in the first two lines. Notations:* $\hat{b} = \sqrt{\frac{n}{2}\frac{3L_{\max}-L}{nL-3L_{\max}}}$, $\tilde{b} = \frac{(3L_{\max}-L)n}{n(n-1)\mu-nL+3L_{\max}}$.

for mini-batching shows that when increasing the mini-batch size $b$, while the iteration complexity can decrease[2], the total complexity increases or is invariant. See for instance results in the non-convex case [111, 122], and for the convex case [7], [79], [3] and finally [100] where one can find the iteration complexity of several variants of SVRG with mini-batching. However, in practice, mini-batching can often lead to faster algorithms. In contrast our total complexity (3.3) clearly highlights that when increasing the mini batch size, the total complexity can decrease and the step size increases, as can be seen in our plot of (3.3) and (3.4) in Figure 3.1. Furthermore $C_m(b)$ is a convex function in $b$ which allows us to determine the optimal mini-batch a priori. For $m = n$ – a widely used loop length in practice – the optimal mini-batch size, depending on the problem setting, is given in Table 3.1. Moreover, we can also determine the optimal loop length. The reason we were able to achieve these new tighter mini-batch complexity bounds was by using the recently introduced concept of expected smoothness [55] alongside a new constant we introduce in this chapter called the expected residual constant. The expected smoothness and residual constants, which we present later in Lemmas 3.9 and 3.10, show how mini-batching (and arbitrary sampling in general) combined with the smoothness of our data can determine how smooth in expectation our resulting mini-batched functions are. The expected smoothness constant has been instrumental in providing a tight mini-batch analysis for SGD [58], SAGA [45] and now SVRG.

**Practical variants.** We took special care so that our analysis allows for practical parameter settings. In particular, often the loop length is set to $m = n$ or $m = n/b$ in the case of mini-batching[3]. And yet, the classical SVRG analysis given in [71] requires $m \geq 20L_{\max}/\mu$ in order to ensure a resulting iteration complexity of $O((n + L_{\max}/\mu)\log(1/\epsilon))$. Furthermore, the standard SVRG analysis relies on averaging the $w^t$ inner iterates after every $m$ iterations of (3.2), yet this too is not what works well in practice[4]. To remedy this, we propose *Free-SVRG*, a variant of SVRG where the inner iterates are not averaged at any point. Furthermore, by developing a new Lyapunov style convergence for *Free-SVRG*, our analysis holds for any choice of $m$, and in particular, for $m = n$ we show that the resulting complexity is also given by $O((n + L_{\max}/\mu)\log(1/\epsilon))$.

We would like to clarify that, after submitting this work in 2019, it come to our attention that *Free-SVRG* is a special case of *k-SVRG* (2018) [121] when $k = 1$.

The only downside of *Free-SVRG* is that the reference point is set using a weighted averaging based on the strong convexity parameter. To fix this issue, [68], and later [81, 83], proposed a loopless version of SVRG. This loopless variant has no explicit inner-loop structure, it instead updates the reference point based on a coin toss and lastly requires no knowledge of the strong convexity parameter and no averaging

[2]Note that the total complexity is equal to the iteration complexity times the mini-batch size $b$.

[3]See for example the lightning package from `scikit-learn` [114]: http://contrib.scikit-learn.org/lightning/ and [109] for examples where $m = n$. See [4] for an example where $m = 5n/b$.

[4]Perhaps an exception to the above issues in the literature is the Katyusha method and its analysis [3], which is an accelerated variant of SVRG. In [3] the author shows that using a loop length $m = 2n$ and by not averaging the inner iterates, the Katyusha method achieves the accelerated complexity of $O((n + \sqrt{(nL_{\max})/\mu})\log(1/\epsilon))$. Though a remarkable advance in the theory of accelerated methods, the analysis in [3] does not hold for the unaccelerated case. This is important since, contrary to the name, the accelerated variants of stochastic methods are not always faster than their non-accelerated counterparts. Indeed, acceleration only helps in the stochastic setting when $L_{\max}/\mu \geq n$, in other words for problems that are sufficiently ill-conditioned.

whatsoever. We introduce *L-SVRG-D*, an improved variant of *Loopless-SVRG* that takes much larger step sizes after the reference point is reset, and gradually smaller step sizes thereafter. We provide a complexity analysis of *L-SVRG-D* that allows for arbitrary sampling and achieves the same complexity as *Free-SVRG*, albeit at the cost of introducing more variance into the procedure due to the coin toss.

## 3.3 Assumptions and sampling

We collect all of the assumptions we use in the following.

**Assumption 3.1.** *There exist $L \geq 0$ and $\mu \geq 0$ such that for all $x, y \in \mathbb{R}^d$,*

$$f(x) \quad \leq \quad f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2} \|x - y\|_2^2, \tag{3.5}$$

$$f(x) \quad \leq \quad f(y) + \langle \nabla f(x), x - y \rangle - \frac{\mu}{2} \|x - y\|_2^2. \tag{3.6}$$

*We say that $f$ is $L$–smooth (3.5) and $\mu$–strongly convex (3.6). Moreover, for all $i \in [n]$, $f_i$ is convex and there exists $L_i \geq 0$ such that $f_i$ is $L_i$–smooth.*

So that we can analyse all forms of mini-batching simultaneously through arbitrary sampling we make use of a *sampling vector*.

**Definition 3.2** (The sampling vector)**.** *We say that the random vector $v = [v_1, \ldots, v_n] \in \mathbb{R}^n$ with distribution $\mathcal{D}$ is a sampling vector if $\mathbb{E}_{\mathbf{S} \sim \mathcal{D}}[v_i] = 1$ for all $i \in [n]$.*

With a sampling vector we can compute an unbiased estimate of $f(w)$ and $\nabla f(w)$ via

$$f_v(w) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n v_i f_i(w) \quad \text{and} \quad \nabla f_v(w) \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n v_i \nabla f_i(w). \tag{3.7}$$

Indeed these are unbiased estimators since

$$\mathbb{E}_{\mathbf{S} \sim \mathcal{D}}[f_v(w)] \;=\; \frac{1}{n} \sum_{i=1}^n \mathbb{E}_{\mathbf{S} \sim \mathcal{D}}[v_i] f_i(w) \;=\; \frac{1}{n} \sum_{i=1}^n f_i(w) \;=\; f(w). \tag{3.8}$$

Likewise we can show that $\mathbb{E}_{\mathbf{S} \sim \mathcal{D}}[\nabla f_v(w)] = \nabla f(w)$. Computing $\nabla f_v$ is cheaper than computing the full gradient $\nabla f$ whenever $v$ is a sparse vector. In particular, this is the case when the support of $v$ is based on a mini-batch sampling.

**Definition 3.3** (Sampling)**.** *A sampling $S \subseteq [n]$ is any random set-valued map which is uniquely defined by the probabilities $\sum_{B \subseteq [n]} p_B = 1$ where $p_B \overset{def}{=} \mathbb{P}(S = B)$ for all $B \subseteq [n]$. A sampling $S$ is called proper if for every $i \in [n]$, we have that $p_i \overset{def}{=} \mathbb{P}[i \in S] = \sum_{C : i \in C} p_C > 0$.*

We can build a sampling vector using sampling as follows.

**Lemma 3.4** (Sampling vector)**.** *Let $S$ be a proper sampling. Let $p_i \overset{def}{=} \mathbb{P}[i \in S]$ and $\mathbf{P} \overset{def}{=} \text{Diag}(p_1, \ldots, p_n)$. Let $v = v(S)$ be a random vector defined by*

$$v(S) \;=\; \mathbf{P}^{-1} \sum_{i \in S} e_i \;\overset{def}{=}\; \mathbf{P}^{-1} e_S. \tag{3.9}$$

*It follows that $v$ is a sampling vector.*

*Proof.* The $i$-th coordinate of $v(S)$ is $v_i(S) = \mathbb{1}(i \in S)/p_i$ and thus

$$\mathbb{E}[v_i(S)] \;=\; \frac{\mathbb{E}[\mathbb{1}(i \in S)]}{p_i} \;=\; \frac{\mathbb{P}[i \in S]}{p_i} \;=\; 1. \qquad \square$$

Our forthcoming analysis holds for all samplings. However, we will pay particular attention to $b$-nice sampling, otherwise known as mini-batching without replacement, since it is often used in practice.

**Definition 3.5** ($b$-nice sampling)**.** *$S$ is a $b$-nice sampling if it is sampling such that*

$$\mathbb{P}[S = B] = \frac{1}{\binom{n}{b}}, \quad \forall B \subseteq [n] : |B| = b.$$

To construct such a sampling vector based on the $b$–nice sampling, note that $p_i = \frac{b}{n}$ for all $i \in [n]$ and thus we have that $v(S) = \frac{n}{b}\sum_{i\in S} e_i$ according to Lemma 3.4. The resulting subsampled function is then $f_v(w) = \frac{1}{|S|}\sum_{i\in S} f_i(w)$, which is simply the mini-batch average over $S$.

Using arbitrary sampling also allows us to consider non-uniform samplings, and for completeness, we present this sampling and several others hereafter.

**Definition 3.6** (Single-element sampling)**.** *Given a set of probabilities $(p_i)_{i\in[n]}$, $S$ is a single-element sampling if $\mathbb{P}(|S| = 1) = 1$ and*

$$\mathbb{P}(S = \{i\}) = p_i \quad \forall i \in [n].$$

**Definition 3.7** (Partition sampling)**.** *Given a partition $\mathcal{B}$ of $[n]$, $S$ is a partition sampling if*

$$p_B \overset{def}{=} \mathbb{P}(S = B) > 0 \quad \forall B \in \mathcal{B}, \text{ and } \sum_{B\in\mathcal{B}} p_B = 1.$$

**Definition 3.8** (Independent sampling)**.** *$S$ is an independent sampling if it includes every $i$ independently with probability $p_i > 0$.*

## 3.4 Free-SVRG: freeing up the inner loop size

Similarly to SVRG, *Free-SVRG* is an inner-outer loop variance reduced algorithm. It differs from the original SVRG [71] on two major points: how the reference point is reset and how the first iterate of the inner loop is defined, see Algorithm 9.

First, in SVRG, the reference point is the average of the iterates of the inner loop. Thus, old iterates and recent iterates have equal weights in the average. This is counterintuitive as one would expect that to reduce the variance of the gradient estimate used in (3.2), one needs a reference point which is closer to the more recent iterates. This is why, inspired by [107], we use the weighted averaging in *Free-SVRG* given in (3.10), which gives more importance to recent iterates compared to old ones.

Second, in SVRG, the first iterate of the inner loop is reset to the reference point. Thus, the inner iterates of the algorithm are not updated using a one step recurrence. In contrast, *Free-SVRG* defines the first iterate of the inner loop as the last iterate of the previous inner loop, as is also done in practice. These changes and a new Lyapunov function analysis are what allows us to freely choose the size of the inner loop[5]. To declutter the notation, we define for a given step size $\alpha > 0$:

$$S_m \overset{\text{def}}{=} \sum_{i=0}^{m-1}(1-\alpha\mu)^{m-1-i} \quad \text{and} \quad p_t \overset{\text{def}}{=} \frac{(1-\alpha\mu)^{m-1-t}}{S_m}, \quad \text{for } t = 0, \ldots, m-1. \tag{3.10}$$

### 3.4.1 Convergence analysis

Our analysis relies on two important constants called the *expected smoothness* constant $\mathcal{L}$ and the *expected residual* constant $\rho$. Their existence is a result of the smoothness of the function $f$ and that of the individual functions $f_i, i \in [n]$. First, we present the keys constant $\mathcal{L}$ and $\rho$ and establish their existence

[5]Hence the name of our method *Free*-SVRG.

**Algorithm 9** *Free-SVRG*

**Parameters** inner-loop length $m$, step size $\alpha$, sampling vector $v \sim \mathcal{D}$, $p_t$ defined in (3.10)
**Initialization** $z_0 = w_0^m \in \mathbb{R}^d$
**for** $s = 1, 2, \ldots, S$ **do**
  $w_s^0 = w_{s-1}^m$
  **for** $t = 0, 1, \ldots, m-1$ **do**
    Sample $v_t \sim \mathcal{D}$
    $g_s^t = \nabla f_{v_t}(w_s^t) - \nabla f_{v_t}(z_{s-1}) + \nabla f(z_{s-1})$
    $w_s^{t+1} = w_s^t - \alpha g_s^t$
  $z_s = \sum_{t=0}^{m-1} p_t w_s^t$
**return** $w_S^m$

and show general properties. Then, we determine their value for particular samplings allowed by arbitrary sampling. Finally, we dive into our main theorem ruling the convergence of Algorithm 9.

#### 3.4.1.1 Expected smoothness and expected residual

**Lemma 3.9** (Expected smoothness, Theorem 3.6 in [58])**.** *Let $v \sim \mathcal{D}$ be a sampling vector and assume that Assumption 3.1 holds. There exists $\mathcal{L} \geq 0$ such that for all $w \in \mathbb{R}^d$,*

$$\mathbb{E}_{\mathbf{S}\sim\mathcal{D}}\left[\|\nabla f_v(w) - \nabla f_v(w^*)\|_2^2\right] \leq 2\mathcal{L}\left(f(w) - f(w^*)\right). \tag{3.11}$$

*Proof.* Since the $f_i$'s are convex, each realization of $f_v$ is convex, and it follows from equation 2.1.7 in [106] that

$$\|\nabla f_v(x) - \nabla f_v(y)\|_2^2 \quad \leq \quad 2L_v\left(f_v(x) - f_v(y) - \langle \nabla f_v(y), x - y\rangle\right). \tag{3.12}$$

Taking expectation over the sampling gives

$$\begin{aligned}
\mathbb{E}\left[\|\nabla f_v(w) - \nabla f_v(w^*)\|_2^2\right] &\overset{(3.12)}{\leq} 2\mathbb{E}\left[L_v\left(f_v(w) - f_v(w^*) - \langle \nabla f_v(w^*), w - w^*\rangle\right)\right] \\
&\overset{(3.7)}{=} \frac{2}{n}\mathbb{E}\left[\sum_{i=1}^n L_v v_i\left(f_i(w) - f_i(w^*) - \langle \nabla f_i(w^*), w - w^*\rangle\right)\right] \\
&= \frac{2}{n}\sum_{i=1}^n \mathbb{E}\left[L_v v_i\right]\left(f_i(w) - f_i(y) - \langle \nabla f_i(w^*), w - w^*\rangle\right) \\
&\overset{(3.1)}{\leq} 2\max_{i=1,\ldots,n}\mathbb{E}\left[L_v v_i\right]\left(f(w) - f(w^*) - \langle \nabla f(w^*), w - w^*\rangle\right) \\
&= 2\max_{i=1,\ldots,n}\mathbb{E}\left[L_v v_i\right]\left(f(w) - f(w^*)\right).
\end{aligned}$$

By comparing the above with (3.11) we have that $\mathcal{L} = \max_{i=1,\ldots,n}\mathbb{E}\left[L_v v_i\right]$. □

**Lemma 3.10** (Expected residual)**.** *Let $v \sim \mathcal{D}$ be a sampling vector and assume that Assumption 3.1 holds. There exists $\rho \geq 0$ such that for all $w \in \mathbb{R}^d$,*

$$\mathbb{E}_{\mathbf{S}\sim\mathcal{D}}\left[\|\nabla f_v(w) - \nabla f_v(w^*) - \nabla f(w)\|_2^2\right] \leq 2\rho\left(f(w) - f(w^*)\right). \tag{3.13}$$

Before the proof, let us introduce the following lemma[6].

[6]Inspired from `https://www.cs.ubc.ca/~nickhar/W12/NotesMatrices.pdf`.

**Lemma 3.11** (Trace inequality)**.** *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be symmetric* $n \times n$ *such that* $\mathbf{A} \succcurlyeq 0$*. Then,*

$$\mathbf{Tr}\left(\mathbf{AB}\right) \leq \lambda_{\max}(\mathbf{B})\mathbf{Tr}\left(\mathbf{A}\right)$$

*Proof.* Let $\mathbf{A} = \sum_{i=1}^{n} \lambda_i(\mathbf{A}) u_i u_i^\top$, where $\lambda_1(\mathbf{A}) \geq \ldots \geq \lambda_n(\mathbf{A}) \geq 0$ denote the ordered eigenvalues of matrix $\mathbf{A}$. Setting $\tilde{u}_i \overset{\text{def}}{=} \sqrt{\lambda_i(\mathbf{A})} u_i$ for all $i \in [n]$, we can write $\mathbf{A} = \sum_{i=1}^{n} \tilde{u}_i \tilde{u}_i^\top$. Then,

$$\begin{aligned}
\mathbf{Tr}\left(AB\right) &= \mathbf{Tr}\left(\sum_{i=1}^{n} \tilde{u}_i \tilde{u}_i^\top \mathbf{B}\right) = \sum_{i=1}^{n} \mathbf{Tr}\left(\tilde{u}_i \tilde{u}_i^\top \mathbf{B}\right) = \sum_{i=1}^{n} \mathbf{Tr}\left(\tilde{u}_i^\top \mathbf{B} \tilde{u}_i\right) = \sum_{i=1}^{n} \tilde{u}_i^\top \mathbf{B} \tilde{u}_i \\
&\leq \lambda_{\max}(\mathbf{B}) \sum_{i=1}^{n} \tilde{u}_i^\top \tilde{u}_i = \lambda_{\max}(\mathbf{B})\mathbf{Tr}\left(\mathbf{A}\right),
\end{aligned}$$

where we use in the inequality that $\mathbf{B} \preccurlyeq \lambda_{\max}(\mathbf{B})\mathbf{I}_n$. □

We now turn to the proof of Lemma 3.10.

*Proof.* Let $v = [v_1, \ldots, v_n] \in \mathbb{R}^n$ be an unbiased sampling vector with $v_i \geq 0$ with probability one. We will show that there exists $\rho \in \mathbb{R}_+$ such that:

$$\mathbb{E}\left[\left\|\nabla f_v(z) - \nabla f_v(w^*) - (\nabla f(z) - \nabla f(w^*))\right\|_2^2\right] \leq 2\rho\left(f(z) - f(w^*)\right). \tag{3.14}$$

Let us expand the squared norm first. Define $\mathbf{DF}(z)$ as the Jacobian of $F(z) \overset{def}{=} [f_1(z), \ldots, f_n(z)]$ We denote $\mathbf{R} \overset{\text{def}}{=} (\mathbf{DF}(z) - \mathbf{DF}(w^*))$

$$\begin{aligned}
C &\overset{\text{def}}{=} \left\|\nabla f_v(z) - \nabla f_v(w^*) - (\nabla f(z) - \nabla f(w^*))\right\|_2^2 \\
&= \frac{1}{n^2}\left\|\left(\mathbf{DF}(z) - \mathbf{DF}(w^*)\right)(v - \mathbb{1})\right\|_2^2 \\
&= \frac{1}{n^2}\langle \mathbf{R}(v - \mathbb{1}), \mathbf{R}(v - \mathbb{1})\rangle_{\mathbb{R}^d} \\
&= \frac{1}{n^2}\mathbf{Tr}\left((v - \mathbb{1})^\top \mathbf{R}^\top \mathbf{R}(v - \mathbb{1})\right) \\
&= \frac{1}{n^2}\mathbf{Tr}\left(\mathbf{R}^\top \mathbf{R}(v - \mathbb{1})(v - \mathbb{1})^\top\right).
\end{aligned}$$

Taking expectation,

$$\begin{aligned}
\mathbb{E}\left[C\right] &= \frac{1}{n^2}\mathbf{Tr}\left(\mathbf{R}^\top \mathbf{R}\,\mathrm{Var}\, v\right) \\
&\leq \frac{1}{n^2}\mathbf{Tr}\left(\mathbf{R}^\top \mathbf{R}\right)\lambda_{\max}(\mathrm{Var}\, v).
\end{aligned} \tag{3.15}$$

Moreover, since the $f_i$'s are convex and $L_i$-smooth, it follows from equation 2.1.7 in [106] that

$$\begin{aligned}
\mathbf{Tr}\left(\mathbf{R}^\top \mathbf{R}\right) &= \sum_{i=1}^{n} \left\|\nabla f_i(z) - \nabla f_i(w^*)\right\|_2^2 \\
&\leq 2\sum_{i=1}^{n} L_i(f_i(z) - f_i(w^*) - \langle \nabla f_i(w^*), z - w^*\rangle) \\
&\leq 2nL_{\max}(f(z) - f(w^*)).
\end{aligned} \tag{3.16}$$

Therefore,

$$\mathbb{E}[C] \overset{(3.15)+(3.16)}{\leq} 2\frac{\lambda_{\max}(\operatorname{Var} v)}{n} L_{\max}(f(z) - f(w^*)). \tag{3.17}$$

Which means

$$\rho = \frac{\lambda_{\max}(\operatorname{Var} v)}{n} L_{\max} \tag{3.18}$$

□

Hence depending on the sampling $S$, we need to study the eigenvalues of the matrix $\operatorname{Var} v$, whose general term is given by

$$(\operatorname{Var} v)_{ij} = \begin{cases} \frac{1}{p_i} - 1 & \text{if } i = j \\ \frac{P_{ij}}{p_i p_j} - 1 & \text{otherwise,} \end{cases} \tag{3.19}$$

with

$$p_i \overset{\text{def}}{=} \mathbb{P}(i \in S) \text{ and } P_{ij} \overset{\text{def}}{=} \mathbb{P}(i \in S, j \in S) \text{ for } i, j \in [n] \tag{3.20}$$

To specialize our results to particular samplings, we introduce some notations:

- $\mathcal{B}$ designates all the possible sets for the sampling $S$,
- $b = |B|$, where $B \in \mathcal{B}$, when the sizes of all the elements of $\mathcal{B}$ are equal.

#### 3.4.1.2 Properties of the expected smoothness and residual

Before going deeper into the convergence analysis of our algorithms, we present some properties of $\mathcal{L}$ and $\rho$. The following lemma shows that the expected smoothness constant is always greater than the strong convexity constant.

**Lemma 3.12.** *If the expected smoothness inequality* (3.11) *holds with constant $\mathcal{L}$ and $f$ is $\mu$–strongly convex, then $\mathcal{L} \geq \mu$.*

For the proof of the later lemma, we need a well known result on the Polyak-Lojasiewicz (PL) inequality.

**Lemma 3.13** (PL inequality)**.** *If $f$ is $\mu$–strongly convex, then for all $w \in \mathbb{R}^d$*

$$\frac{1}{2\mu} \|\nabla f(w)\|_2^2 \geq f(w) - f(w^*), \quad \forall w \in \mathbb{R}^d \ . \tag{3.21}$$

*Proof.* Since $f$ is $\mu$–strongly convex, we have from, rearranging (3.6), that for all $x, y \in \mathbb{R}^d$

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \|x - y\|_2^2 \ .$$

Minimizing both sides of this inequality in $y$ proves (3.21). □

*Proof of Lemma 3.12.* We have, since $\mathbb{E}[\nabla f_v(w) - \nabla f_v(w^*)] = \nabla f(w)$

$$\begin{aligned} \mathbb{E}\left[\|\nabla f_v(w) - \nabla f_v(w^*) - \nabla f(w)\|_2^2\right] &\overset{(3.23)}{=} \mathbb{E}\left[\|\nabla f_v(w) - \nabla f_v(w^*)\|_2^2\right] - \|\nabla f(w)\|_2^2 \\ &\overset{(3.11)+(3.21)}{\leq} 2(\mathcal{L} - \mu)(f(w) - f(w^*)). \end{aligned} \tag{3.22}$$

Hence $2(\mathcal{L} - \mu)(f(w) - f(w^*)) \geq 0$, which means $\mathcal{L} \geq \mu$. where in the first equality we used the following property

$$\mathbb{E}\left[\|X - \mathbb{E}[X]\|_2^2\right] = \mathbb{E}\left[\|X\|_2^2\right] - \|\mathbb{E}[X]\|_2^2 \leq \mathbb{E}\left[\|X\|_2^2\right] \ , \tag{3.23}$$

for any random vector $X \in \mathbb{R}^d$. □

**Remark 3.14.** *Consider the expected residual constant $\rho$ defined in Lemma 3.10. This constant verifies for all $w \in \mathbb{R}^d$,*

$$\mathbb{E}\left[\|\nabla f_v(w) - \nabla f_v(w^*) - \nabla f(w)\|\right] \leq 2\rho(f(w) - f(w^*)).$$

*From Equation* (3.22)*, we can see that we can use $\rho = \mathcal{L}$ as the expected residual constant.*

#### 3.4.1.3 Expected smoothness and expected residual for particular samplings

Next, we present these constants for $b$–nice sampling, single-element sampling, partition sampling and independent sampling. The results on the expected smoothness constants related to the samplings we present here are all derived in [58] and thus are given without proof.

**$b$–nice sampling** Though Lemma 3.9 establishes the existence of the expected smoothness $\mathcal{L}$, it was only very recently that a tight estimate of $\mathcal{L}$ was conjectured in [45] and proven in [58]. In particular, for our working example of $b$–nice sampling, we have that the constants $\mathcal{L}$ and $\rho$ have simple closed formulae that depend on $b$.

**Lemma 3.15** ($\mathcal{L}$ and $\rho$ for $b$-nice sampling)**.** *Let $v$ be a sampling vector based on the $b$–nice sampling from Definition 3.5. It follows that*

$$\mathcal{L} = \mathcal{L}(b) \overset{def}{=} \frac{1}{b}\frac{n-b}{n-1}L_{\max} + \frac{n}{b}\frac{b-1}{n-1}L \ , \tag{3.24}$$

$$\rho = \rho(b) \overset{def}{=} \frac{1}{b}\frac{n-b}{n-1}L_{\max} \ . \tag{3.25}$$

*Proof.* For uniform $b$-nice sampling, we have using notations from (3.20)

$$\begin{aligned} \forall i \in [n],\ p_i &= \frac{c_1}{|\mathcal{B}|}, \\ \forall i,j \in [n],\ P_{ij} &= \frac{c_2}{|\mathcal{B}|}, \end{aligned}$$

with $c_1 = \binom{n-1}{b-1}$, $c_2 = \binom{n-2}{b-2}$ and $|\mathcal{B}| = \binom{n}{b}$. Hence,

$$\operatorname{Var} v \overset{(3.19)}{=} \begin{bmatrix} \frac{|\mathcal{B}|}{c_1} - 1 & \frac{|\mathcal{B}|c_2}{c_1^2} - 1 & \dots & \frac{|\mathcal{B}|c_2}{c_1^2} - 1 & \frac{|\mathcal{B}|c_2}{c_1^2} - 1 \\ \frac{|\mathcal{B}|c_2}{c_1^2} - 1 & \frac{|\mathcal{B}|}{c_1} - 1 & \dots & \frac{|\mathcal{B}|c_2}{c_1^2} - 1 & \frac{|\mathcal{B}|c_2}{c_1^2} - 1 \\ \vdots & & \ddots & & \vdots \\ \frac{|\mathcal{B}|c_2}{c_1^2} - 1 & \dots & \dots & \frac{|\mathcal{B}|}{c_1} - 1 & \frac{|\mathcal{B}|c_2}{c_1^2} - 1 \\ \frac{|\mathcal{B}|c_2}{c_1^2} - 1 & \dots & \dots & \frac{|\mathcal{B}|c_2}{c_1^2} - 1 & \frac{|\mathcal{B}|}{c_1} - 1 \end{bmatrix}.$$

As noted in Appendix C of [55], $\operatorname{Var} v$ is then a circulant matrix with associated vector

$$\left(\frac{|\mathcal{B}|}{c_1} - 1, \frac{|\mathcal{B}|c_2}{c_1^2} - 1, \dots, \frac{|\mathcal{B}|c_2}{c_1^2} - 1\right),$$

and, as such, it has two eigenvalues

$$\begin{aligned} \lambda_1 &\overset{\text{def}}{=} \frac{|\mathcal{B}|}{c_1}\left(1 + (n-1)\frac{c_2}{c_1}\right) - n = 0, \\ \lambda_2 &\overset{\text{def}}{=} \frac{|\mathcal{B}|}{c_1}\left(1 - \frac{c_2}{c_1}\right) = \frac{n(n-b)}{b(n-1)}. \end{aligned} \tag{3.26}$$

Hence, the expected residual can be computed explicitly as

$$\rho \overset{(3.18)}{=} \frac{n-b}{(n-1)b} L_{\max}. \tag{3.27}$$

□

The reason that the expected smoothness and expected residual constants are so useful in obtaining a tight mini-batch analysis is because, as the mini-batch size $b$ goes from $n$ to 1, $\mathcal{L}(b)$ (resp. $\rho(b)$) gracefully interpolates between the smoothness of the full function $\mathcal{L}(n) = L$ (resp. $\rho(n) = 0$), and the smoothness of the individual $f_i$ functions $\mathcal{L}(1) = L_{\max}$ (resp $\rho(1) = L_{\max}$).

### Single-element sampling.

**Lemma 3.16** ($\mathcal{L}$ for single-element sampling. Proposition 3.7 in [58])**.** *Consider $S$ a single-element sampling from Definition 3.6. If for all $i \in [n]$, $f_i$ is $L_i$–smooth, then*

$$\mathcal{L} \quad = \quad \frac{1}{n} \max_{i \in [n]} \frac{L_i}{p_i}$$

*where $p_i = \mathbb{P}(S = \{i\})$.*

**Remark 3.17.** *Consider $S$ a single-element sampling from Definition 3.6. Then, the probabilities that maximize $\mathcal{L}$ are*

$$p_i = \frac{L_i}{\sum_{j \in [n]} L_j}.$$

*Consequently,*

$$\mathcal{L} = \bar{L} \overset{def}{=} \frac{1}{n} \sum_{i=1}^{n} L_i.$$

In contrast, for uniform single-element sampling, *i.e.,* when $p_i = \frac{1}{n}$ for all $i$, we have $\mathcal{L} = L_{\max}$, which can be significantly larger than $\bar{L}$. Since the step sizes of all our algorithms are a decreasing function of $\mathcal{L}$, importance sampling can lead to much faster algorithms.

From Remark 3.14, we can take $\mathcal{L}$ as the expected residual constant. Thus, we simply use the expected smoothness constant from Lemma 3.16.

**Lemma 3.18** ($\rho$ for single-element sampling)**.** *Consider single-element sampling from Definition 3.6. If for all $i \in [n]$, $f_i$ is $L_i$-smooth, then*

$$\rho = \frac{1}{n} \max_{i \in [n]} \frac{L_i}{p_i}.$$

### Partition sampling.

**Lemma 3.19** ($\mathcal{L}$ for partition sampling. Proposition 3.7 in [58])**.** *Given a partition $\mathcal{B}$ of $[n]$, consider $S$ a partition sampling from Definition 3.8. For all $B \in \mathcal{B}$, suppose that $f_B(w) \overset{def}{=} \frac{1}{b} \sum_{i \in \mathcal{B}} f_i(w)$ is $L_B$–smooth. Then, with $p_B = \mathbb{P}(S = B)$*

$$\mathcal{L} = \frac{1}{n} \max_{B \in \mathcal{B}} \frac{L_B}{p_B} \quad .$$

**Lemma 3.20** ($\rho$ for uniform partition sampling)**.** *Suppose that $b$ divises $n$ and consider partition sampling from Definition 3.7. Given a partition $\mathcal{B}$ of $[n]$ of size $\frac{b}{n}$, if each $f_i$ is $L_{\max}$-smooth, then,*

$$\rho = \left(1 - \frac{b}{n}\right) L_{\max} \quad .$$

*Proof.* Recall that for partition sampling, we choose *a priori* a partition $\mathcal{B} = B_1 \sqcup \cdots \sqcup B_{\frac{n}{b}}$ of $[n]$. Then, for $k \in [\frac{n}{b}]$,

$$\forall i \in [n],\ p_i = \begin{cases} p_{B_k} = \frac{b}{n} & \text{if } i \in B_k \\ 0 & \text{otherwise,} \end{cases} \tag{3.28}$$

$$\forall i,j \in [n],\ P_{ij} = \begin{cases} p_{B_k} = \frac{b}{n} & \text{if } i,j \in B_k \\ 0 & \text{otherwise.} \end{cases} \tag{3.29}$$

Let $k \in [\frac{n}{b}]$. If $i,j \in B_k$, then $\frac{1}{p_i} - 1 = \frac{P_{ij}}{p_i p_j} - 1 = \frac{n}{b} - 1$.

As a result, up to a reordering of the observations, $\operatorname{Var} v$ is a block diagonal matrix, whose diagonal matrices, which are all equal, are given by, for $k \in [\frac{n}{b}]$,

$$\mathbf{V}_k = \left(\frac{n}{b} - 1\right) \mathbb{1}_b \mathbb{1}_b^\top = \begin{bmatrix} \frac{n}{b}-1 & \frac{n}{b}-1 & \dots & \frac{n}{b}-1 & \frac{n}{b}-1 \\ \frac{n}{b}-1 & \frac{n}{b}-1 & \dots & \frac{n}{b}-1 & \frac{n}{b}-1 \\ \vdots & & \ddots & & \vdots \\ \frac{n}{b}-1 & \dots & \dots & \frac{n}{b}-1 & \frac{n}{b}-1 \\ \frac{n}{b}-1 & \dots & \dots & \frac{n}{b}-1 & \frac{n}{b}-1 \end{bmatrix} \in \mathbb{R}^{b \times b}.$$

Since all the matrices on the diagonal are equal, the eigenvalues of $\operatorname{Var} v$ are simply those of one of these matrices. Any matrix $\mathbf{V}_k = \left(\frac{n}{b} - 1\right) \mathbb{1}_b \mathbb{1}_b^\top$ we consider has two eigenvalues: 0 and $n - b$. Then,

$$\rho \overset{(3.18)}{=} \left(1 - \frac{b}{n}\right) L_{\max}. \tag{3.30}$$

□

If $b = n$, SVRG with uniform partition sampling boils down to gradient descent as we recover $\rho = 0$. For $b = 1$, we have $\rho = \left(1 - \frac{1}{n}\right) L_{\max}$.

### Independent sampling.

**Lemma 3.21** ($\mathcal{L}$ for independent sampling. Proposition 3.8 in [58])**.** *Consider $S$ a independent sampling from Definition 3.8. Note $p_i = \mathbb{P}(i \in S)$. If for all $i \in [n]$, $f_i$ is $L_i$–smooth and $f$ is $L$–smooth, then*

$$\mathcal{L} = L + \max_{i \in [n]} \frac{1 - p_i}{p_i} \frac{L_i}{n}$$

*where $p_i = \mathbb{P}(S = \{i\})$.*

**Lemma 3.22** ($\rho$ for independent sampling)**.** *Consider independent sampling from Definition 3.8. Let $p_i = \mathbb{P}(i \in S)$. If each $f_i$ is $L_{\max}$-smooth, then*

$$\rho = \left(\frac{1}{\min_{i \in [n]} p_i} - 1\right) \frac{L_{\max}}{n}. \tag{3.31}$$

*Proof.* Using the notations from (3.20), we have

$$\begin{aligned} \forall i \in [n],\ p_i &= p_i, \\ \forall i,j \in [n],\ P_{ij} &= p_i p_j \quad \text{when } i \neq j. \end{aligned}$$

Thus, according to (3.19):

$$\operatorname{Var} v = \operatorname{Diag}\left(\frac{1}{p_1} - 1, \frac{1}{p_2} - 1, \dots, \frac{1}{p_n} - 1\right).$$

whose largest eigenvalue is

$$\lambda_{\max}(\operatorname{Var} v) = \max_{i\in[n]} \frac{1}{p_i} - 1 = \frac{1}{\min_{i\in[n]} p_i} - 1.$$

Consequently,

$$\rho \overset{(3.18)}{=} \left(\frac{1}{\min_{i\in[n]} p_i} - 1\right) \frac{L_{\max}}{n}. \tag{3.32}$$

□

If $p_i = \frac{1}{n}$ for all $i \in [n]$, which corresponds in expectation to uniform single-element sampling SVRG since $\mathbb{E}[|S|] = 1$, we have $\rho = \frac{n-1}{n} L_{\max}$. While if $p_i = 1$ for all $i \in [n]$, this leads to gradient descent and we recover $\rho = 0$.

The following remark gives a condition to construct an independent sampling with $E|S| = b$.

**Remark 3.23.** *One can add the following condition on the probabilities: $\sum_{i=1}^n p_i = b$, such that $\mathbb{E}[|S|] = b$. Such a sampling is called b-independent sampling. This condition is obviously met if $p_i = \frac{b}{n}$ for all $i \in [n]$.*

**Lemma 3.24.** *Let $S$ be a independent sampling from $[n]$ and let $p_i = \mathbb{P}[i \in S]$ for all $i \in [n]$. If $\sum_{i=1}^n p_i = b$, then $\mathbb{E}[|S|] = b$.*

*Proof.* Let us model our sampling by a tossing of $n$ independent rigged coins. Let $X_1, \ldots, X_n$ be $n$ Bernoulli random variables representing these tossed coin, *i.e.,* $X_i \sim \mathcal{B}(p_i)$, with $p_i \in [0,1]$ for $i \in [n]$. If $X_i = 1$, then the point $i$ is selected in the sampling $S$. Thus the number of selected points in the mini-batch $|S|$ can be denoted as the following random variable $\sum_{i=1}^n X_i$, and its expectation equals

$$\mathbb{E}[|S|] = \mathbb{E}\left[\sum_{i=1}^n X_i\right] = \sum_{i=1}^n \mathbb{E}[X_i] = \sum_{i=1}^n p_i = b \ .$$

□

**Remark 3.25.** *Note that one does not need the independence of the $(X_i)_{i=1,\ldots,n}$.*

#### 3.4.1.4 Main convergence theorem

We start by showing that we can bound the second moment of a variance reduced gradient estimate using $\mathcal{L}$ and $\rho$ as follows.

**Lemma 3.26.** *Let Assumption 3.1 hold. Let $w, z \in \mathbb{R}^d$ and $v \sim \mathcal{D}$ be sampling vector. Consider*

$$g(w,z) \overset{def}{=} \nabla f_v(w) - \nabla f_v(z) + \nabla f(z) \ . \tag{3.33}$$

*As a consequence of* (3.11) *and* (3.13) *we have that*

$$\mathbb{E}_{\mathbf{S}\sim\mathcal{D}}\left[\|g(w,z)\|_2^2\right] \leq 4\mathcal{L}(f(w) - f(w^*)) + 4\rho(f(z) - f(w^*)). \tag{3.34}$$

*Proof.*

$$\begin{aligned}
\mathbb{E}_{\mathbf{S}\sim\mathcal{D}}\left[\|g(w,z)\|_2^2\right] &\overset{(3.33)}{=} \mathbb{E}_{\mathbf{S}\sim\mathcal{D}}\left[\|\nabla f_v(w)-\nabla f_v(w^*)+\nabla f_v(w^*)-\nabla f_v(z)+\nabla f(z)\|_2^2\right]\\
&\overset{(3.35)}{\leq} 2\mathbb{E}_{\mathbf{S}\sim\mathcal{D}}\left[\|\nabla f_v(w)-\nabla f_v(w^*)\|_2^2\right]\\
&\quad +2\mathbb{E}_{\mathbf{S}\sim\mathcal{D}}\left[\|\nabla f_v(z)-\nabla f_v(w^*)-\nabla f(z)\|_2^2\right]\\
&\overset{(3.11)+(3.13)}{\leq} 4\mathcal{L}(f(w)-f(w^*))+4\rho(f(z)-f(w^*)).
\end{aligned}$$

where in the second inequality we used the following inequality

$$\|a+b\|_2^2 \leq 2\|a\|_2^2+2\|b\|_2^2 \ . \tag{3.35}$$

for all $a,b\in\mathbb{R}^d$. □

Next we present a new Lyapunov style convergence analysis through which we will establish the convergence of the iterates and the function values simultaneously.

**Theorem 3.27.** *Consider the setting of Algorithm 9 and the following Lyapunov function*

$$\phi_s \overset{def}{=} \|w_s^m-w^*\|_2^2+\mathcal{P}_s \quad \textit{where} \quad \mathcal{P}_s \overset{def}{=} 8\alpha^2\rho S_m(f(z_s)-f(w^*)). \tag{3.36}$$

*If Assumption 3.1 holds and if* $\alpha\leq\frac{1}{2(\mathcal{L}+2\rho)}$*, then*

$$\mathbb{E}\left[\phi_s\right] \ \leq \ \beta^s\phi_0, \quad \textit{where} \quad \beta=\max\left\{(1-\alpha\mu)^m,\frac{1}{2}\right\}. \tag{3.37}$$

*Proof of Theorem 3.27.* To clarify the notations, we recall that $g_s^t \overset{\text{def}}{=} g(w_s^t,z_{s-1})$. Then, we get

$$\begin{aligned}
\mathbb{E}_t\left[\|w_s^{t+1}-w^*\|_2^2\right] &= \mathbb{E}_t\left[\|w_s^t-w^*-\alpha g_s^t\|_2^2\right]\\
&= \|w_s^t-w^*\|_2^2-2\alpha\mathbb{E}_t\left[g_s^t\right]^\top(w_s^t-w^*)+\alpha^2\mathbb{E}_t\left[\|g_s^t\|_2^2\right]\\
&\overset{(3.8)+(3.34)}{\leq} \|w_s^t-w^*\|_2^2-2\alpha\nabla f(w_s^t)^\top(w_s^t-w^*)\\
&\quad +2\alpha^2\left[2\mathcal{L}(f(w_s^t)-f(w^*))+2\rho(f(z_{s-1})-f(w^*))\right]\\
&\overset{(3.6)}{\leq} (1-\alpha\mu)\|w_s^t-w^*\|_2^2-2\alpha(1-2\alpha\mathcal{L})\left(f(w_s^t)-f(w^*)\right)\\
&\quad +4\alpha^2\rho(f(z_{s-1})-f(w^*)).
\end{aligned} \tag{3.38}$$

Note that since $\alpha\leq\frac{1}{2(\mathcal{L}+2\rho)}$ and $\rho\geq 0$, we have that

$$\alpha \overset{\text{Lemma 3.12}}{\leq} \frac{1}{2\mu},$$

and consequently $(1-\alpha\mu)>0$. Thus by iterating (3.38) over $t=0,\ldots,m-1$ and taking the expectation,

since $w_s^0 = w_{s-1}^m$, we obtain

$$\begin{aligned}
\mathbb{E}\left[\|w_s^m - w^*\|_2^2\right] &\leq (1-\alpha\mu)^m \mathbb{E}\left[\|w_{s-1}^m - w^*\|_2^2\right] \\
&\quad -2\alpha(1-2\alpha\mathcal{L})\sum_{t=0}^{m-1}(1-\alpha\mu)^{m-1-t}\mathbb{E}\left[f(w_s^t) - f(w^*)\right] \\
&\quad +4\alpha^2\rho\mathbb{E}\left[f(z_{s-1}) - f(w^*)\right]\sum_{t=0}^{m-1}(1-\alpha\mu)^{m-1-t} \\
&\overset{(3.10)}{=} (1-\alpha\mu)^m \mathbb{E}\left[\|w_{s-1}^m - w^*\|_2^2\right] - 2\alpha(1-2\alpha\mathcal{L})S_m\sum_{t=0}^{m-1}p_t\mathbb{E}\left[f(w_s^t) - f(w^*)\right] \\
&\quad +4\alpha^2\rho S_m\mathbb{E}\left[f(z_{s-1}) - f(w^*)\right] \\
&\overset{(3.36)}{=} (1-\alpha\mu)^m \mathbb{E}\left[\|w_{s-1}^m - w^*\|_2^2\right] - 2\alpha(1-2\alpha\mathcal{L})S_m\sum_{t=0}^{m-1}p_t\mathbb{E}\left[f(w_s^t) - f(w^*)\right] \\
&\quad +\frac{1}{2}\mathbb{E}\left[\mathcal{P}_{s-1}\right]. \qquad (3.39)
\end{aligned}$$

Weights $p_t$ are defined in (3.10). We note that $(1-\alpha\mu) > 0$ implies that $p_t > 0$ for all $t = 0, \ldots, m-1$, and by construction we get $\sum_{t=0}^{m-1} p_t = 1$. Since $f$ is convex, we have by Jensen's inequality that

$$\begin{aligned}
f(z_s) - f(w^*) &= f\left(\sum_{t=0}^{m-1} p_t w_s^t\right) - f(w^*) \\
&\leq \sum_{t=0}^{m-1} p_t(f(w_s^t) - f(w^*)). \qquad (3.40)
\end{aligned}$$

Consequently,

$$\mathbb{E}\left[\mathcal{P}_s\right] \overset{(3.36)+(3.40)}{\leq} 8\alpha^2\rho S_m\sum_{t=0}^{m-1}p_t\mathbb{E}\left[\left(f(w_s^t) - f(w^*)\right)\right]. \qquad (3.41)$$

As a result,

$$\begin{aligned}
\mathbb{E}\left[\phi_s\right] &= \mathbb{E}\left[\|w_s^m - w^*\|_2^2\right] + \mathbb{E}\left[\mathcal{P}_s\right] \\
&\overset{(3.39)+(3.41)}{\leq} (1-\alpha\mu)^m \mathbb{E}\left[\|w_{s-1}^m - w^*\|_2^2\right] + \frac{1}{2}\mathbb{E}\left[\mathcal{P}_{s-1}\right] \\
&\quad -2\alpha(1-2\alpha(\mathcal{L}+2\rho))S_m\sum_{t=0}^{m-1}p_t\mathbb{E}\left[\left(f(w_s^t) - f(w^*)\right)\right].
\end{aligned}$$

Since $\alpha \leq \frac{1}{2(\mathcal{L}+2\rho)}$, the above implies

$$\begin{aligned}
\mathbb{E}\left[\phi_s\right] &\leq (1-\alpha\mu)^m \mathbb{E}\left[\|w_{s-1}^m - w^*\|_2^2\right] + \frac{1}{2}\mathbb{E}\left[\mathcal{P}_{s-1}\right] \\
&\leq \beta\mathbb{E}\left[\phi_{s-1}\right],
\end{aligned}$$

where $\beta = \max\{(1-\alpha\mu)^m, \frac{1}{2}\}$.

Moreover, if we set $z_s = w_s^t$ with probability $p_t$, for $t = 0, \ldots, m-1$, the result would still hold. Indeed (3.40) would hold with equality and the rest of the proof would follow verbatim. □

### 3.4.2 Total complexity for $b$–nice sampling

To gain better insight into the convergence rate stated in Theorem 3.27, we present the total complexity of Algorithm 9 when $v$ is defined via the $b$–nice sampling introduced in Definition 3.5.

**Corollary 3.28.** *Consider the setting of Algorithm 9 and suppose that we use $b$–nice sampling. Let $\alpha = \frac{1}{2(\mathcal{L}(b)+2\rho(b))}$, where $\mathcal{L}(b)$ and $\rho(b)$ are given in* (3.24) *and* (3.25)*. We have that the total complexity of finding an $\epsilon > 0$ approximate solution that satisfies $\mathbb{E}\left[\|w_s^m - w^*\|_2^2\right] \leq \epsilon\, \phi_0$ is*

$$C_m(b) \quad \stackrel{def}{=} \quad 2\left(\frac{n}{m} + 2b\right) \max\left\{\frac{\mathcal{L}(b) + 2\rho(b)}{\mu}, m\right\} \log\left(\frac{1}{\epsilon}\right). \tag{3.42}$$

*Proof.* Noting $\beta = \max\left\{\left(1 - \frac{\mu}{2(\mathcal{L}(b)+2\rho(b))}\right)^m, \frac{1}{2}\right\}$, we need to chose $s$ so that $\beta^s \leq \epsilon$, that is $s \geq \frac{\log(1/\epsilon)}{\log(1/\beta)}$. Since in each inner iteration we evaluate $2b$ gradients of the $f_i$ functions, and in each outer iteration we evaluate all $n$ gradients, this means that the total complexity will be given by

$$\begin{aligned} C \quad &\stackrel{\text{def}}{=} \quad (n + 2bm)\frac{\log(1/\epsilon)}{\log(1/\beta)} \\ &= \quad (n + 2bm)\max\left\{-\frac{1}{m\log(1 - \frac{\mu}{2(\mathcal{L}(b)+2\rho(b))})}, \frac{1}{\log 2}\right\}\log\left(\frac{1}{\epsilon}\right) \\ &\stackrel{(3.43)}{\leq} \quad (n + 2bm)\max\left\{\frac{1}{m}\frac{2(\mathcal{L}(b) + 2\rho(b))}{\mu}, 2\right\}\log\left(\frac{1}{\epsilon}\right). \end{aligned}$$

where in the last line we used the following logarithm inequality For all $x > 0$,

$$\log(x) \leq x - 1 \ . \tag{3.43}$$

□

Now (3.3) results from plugging (3.24) and (3.25) into (3.42). As an immediate sanity check, we check the two extremes $b = n$ and $b = 1$. When $b = n$, we would expect to recover the iteration complexity of gradient descent, as we do in the next corollary[7].

**Corollary 3.29.** *Consider the setting of Corollary 3.28 with $b = n$ and $m = 1$, thus $\alpha = \frac{1}{2(\mathcal{L}(n)+2\rho(n))} = \frac{1}{2L}$. Hence, the resulting total complexity* (3.42) *is given by*

$$C_1(n) \ = \ 6n\frac{L}{\mu}\log\left(\frac{1}{\epsilon}\right) \ .$$

In practice, the most common setting is choosing $b = 1$ and the size of the inner loop $m = n$. Here we recover a complexity that is common to other non-accelerated algorithms [129, 133], [33], [80], and for a range of values of $m$ including $m = n$.

**Corollary 3.30.** *Consider the setting of Corollary 3.28 with $b = 1$ and thus $\alpha = \frac{1}{2(\mathcal{L}(1)+2\rho(1))} = \frac{1}{6L_{\max}}$. Hence the resulting total complexity* (3.42) *is given by*

$$C_m(1) \ = \ 18\left(n + \frac{L_{\max}}{\mu}\right)\log\left(\frac{1}{\epsilon}\right) \ ,$$

*so long as $m \in \left[\min(n, \frac{L_{\max}}{\mu}), \max(n, \frac{L_{\max}}{\mu})\right]$.*

[7] Though the resulting complexity is 6 times the tightest gradient descent complexity, it is of the same order.

*Proof.* Recall that from (3.42), using the fact that $\mathcal{L}(1) = \rho(1) = L_{\max}$, we have

$$C_m(1) = 2\left(\frac{n}{m} + 2\right) \max\left\{\frac{3L_{\max}}{\mu}, m\right\} \log\left(\frac{1}{\epsilon}\right).$$

When $n \geq \frac{L_{\max}}{\mu}$, then, $m \in \left[\frac{L_{\max}}{\mu}, n\right]$. We can rewrite $C_m(1)$ as

$$C_m(1) = 2(n + 2m) \max\left\{\frac{1}{m}\frac{3L_{\max}}{\mu}, 1\right\} \log\left(\frac{1}{\epsilon}\right).$$

We have $\frac{1}{m}\frac{3L_{\max}}{\mu} \leq 3$ and $n + 2m \leq 3n$. Hence,

$$C_m(1) \leq 18n \log\left(\frac{1}{\epsilon}\right) = O\left(\left(n + \frac{L_{\max}}{\mu}\right) \log\left(\frac{1}{\epsilon}\right)\right).$$

When $n \leq \frac{L_{\max}}{\mu}$, then, $m \in \left[n, \frac{L_{\max}}{\mu}\right]$. We have $\frac{n}{m} \leq 1$ and $m \leq \frac{3L_{\max}}{\mu}$. Hence,

$$C_m(1) \leq \frac{18L_{\max}}{\mu} \log\left(\frac{1}{\epsilon}\right) = O\left(\left(n + \frac{L_{\max}}{\mu}\right) \log\left(\frac{1}{\epsilon}\right)\right).$$

□

Thus total complexity is essentially invariant for $m = n$, $m = L_{\max}/\mu$ and everything in between.

## 3.5 L-SVRG-D: a decreasing step size approach

Although *Free-SVRG* solves multiple issues regarding the construction and analysis of SVRG, it still suffers from an important issue: it requires the knowledge of the strong convexity constant, as is the case for the original SVRG algorithm [71]. One can of course use an explicit small regularization parameter as a proxy, but this can result in a slower algorithm.

A loopless variant of SVRG was proposed and analysed in [68, 81, 83]. At each iteration, their method makes a coin toss. With (a low) probability $p$, typically $1/n$, the reference point is reset to the previous iterate, and with probability $1 - p$, the reference point remains the same. This method does not require knowledge of the strong convexity constant.

Our method, *L-SVRG-D*, uses the same loopless structure as in [68, 81, 83] but introduces different step sizes at each iteration, see Algorithm 10. We initialize the step size to a fixed value $\alpha > 0$. At each iteration we toss a coin, and if it lands heads (with probability $1 - p$) the step size decreases by a factor $\sqrt{1-p}$. If it lands tails (with probability $p$) the reference point is reset to the most recent iterate and the step size is reset to its initial value $\alpha$.

This allows us to take larger steps than *L-SVRG* when we update the reference point, *i.e.,* when the variance of the unbiased estimate of the gradient is low, and smaller steps when this variance increases.

**Theorem 3.31.** *Consider the iterates of Algorithm 10 and the following Lyapunov function*

$$\phi^k \overset{def}{=} \left\|w^k - w^*\right\|_2^2 + \mathcal{P}^k \quad \textit{where} \quad \mathcal{P}^k \overset{def}{=} \frac{8\alpha_k^2\mathcal{L}}{p(3-2p)}\left(f(z^k) - f(w^*)\right), \quad \forall k \in \mathbb{N}. \tag{3.44}$$

*If Assumption 3.1 holds and*

$$\alpha \leq \frac{1}{2\zeta_p\mathcal{L}}, \quad \textit{where} \quad \zeta_p \overset{def}{=} \frac{(7-4p)(1-(1-p)^{\frac{3}{2}})}{p(2-p)(3-2p)}, \tag{3.45}$$

**Algorithm 10** L-SVRG-D

**Parameters** step size $\alpha$, $p \in (0,1]$, and a sampling vector $v \sim \mathcal{D}$
**Initialization** $z^0 = w^0 \in \mathbb{R}^d$, $\alpha_0 = \alpha$
**for** $k = 1, 2, \ldots, K-1$ **do**

Sample $v_k \sim \mathcal{D}$
$g^k = \nabla f_{v_k}(w^k) - \nabla f_{v_k}(z^k) + \nabla f(z^k)$
$w^{k+1} = w^k - \alpha_k g^k$
$$(z^{k+1}, \alpha_{k+1}) = \begin{cases} (w^k, \alpha) & \text{with probability } p \\ (z^k, \sqrt{1-p}\,\alpha_k) & \text{with probability } 1-p \end{cases}$$

**return** $w^K$

*then*

$$\mathbb{E}\left[\phi^k\right] \leq \beta^k \phi^0, \quad \textit{where} \quad \beta \;=\; \max\left\{1 - \frac{2}{3}\alpha\mu, 1 - \frac{p}{2}\right\}. \tag{3.46}$$

**Remark 3.32.** *To get a sense of the formula of the step size given in* (3.45)*, it is easy to show that $\zeta_p$ is an increasing function of $p$ such that $7/4 \;\leq\; \zeta_p \;\leq\; 3$. Since typically $p \approx 0$, we often take a step which is approximately $\alpha \leq 2/(7\mathcal{L})$.*

Before analysing Algorithm 10, we present a lemma that allows to compute the expectations $\mathbb{E}\left[\alpha_k\right]$ and $\mathbb{E}\left[\alpha_k^2\right]$, that will be used in the analysis.

**Lemma 3.33.** *Consider the step sizes defined by Algorithm 10. We have*

$$\mathbb{E}\left[\alpha_k\right] \;=\; \frac{(1-p)^{\frac{3k+2}{2}}(1-\sqrt{1-p})+p}{1-(1-p)^{\frac{3}{2}}}\alpha. \tag{3.47}$$

$$\mathbb{E}\left[\alpha_k^2\right] \;=\; \frac{1+(1-p)^{2k+1}}{2-p}\alpha^2. \tag{3.48}$$

*Proof.* Taking expectation with respect to the filtration induced by the sequence of step sizes $\{\alpha_1, \ldots, \alpha_k\}$

$$\mathbb{E}_p\left[\alpha_{k+1}\right] = (1-p)\sqrt{1-p}\;\alpha_k + p\alpha. \tag{3.49}$$

Then taking total expectation

$$\mathbb{E}\left[\alpha_{k+1}\right] = (1-p)\sqrt{1-p}\mathbb{E}\left[\alpha_k\right] + p\alpha. \tag{3.50}$$

Hence the sequence $(\mathbb{E}\left[\alpha_k\right])_{k\geq 1}$ is uniquely defined by

$$\mathbb{E}\left[\alpha_k\right] \;=\; \frac{(1-p)^{\frac{3k+2}{2}}(1-\sqrt{1-p})+p}{1-(1-p)^{\frac{3}{2}}}\alpha. \tag{3.51}$$

Indeed, applying (3.50) recursively gives

$$\mathbb{E}\left[\alpha_k\right] = (1-p)^{\frac{3k}{2}}\alpha + p\alpha\sum_{i=0}^{k-1}(1-p)^{\frac{3i}{2}}.$$

Adding up the geometric series gives

$$\begin{aligned}\mathbb{E}\left[\alpha_k\right] &= \alpha(1-p)^{\frac{3k}{2}} + p\alpha\frac{1-(1-p)^{\frac{3k}{2}}}{1-(1-p)^{\frac{3}{2}}} \\ &= \frac{(1-p)^{\frac{3k}{2}}(1-(1-p)^{\frac{3}{2}}) - (1-p)^{\frac{3k}{2}}p + p}{1-(1-p)^{\frac{3}{2}}}\alpha \ .\end{aligned}$$

Which leads to (3.51) by factorizing. The same arguments are used to compute $\mathbb{E}\left[\alpha_k^2\right]$. □

We now present a proof of Theorem 3.31.

*Proof.* We recall that $g^k \overset{\text{def}}{=} \nabla f(w^k)$. First, we get

$$\begin{aligned}\mathbb{E}_k\left[\|w^{k+1}-w^*\|_2^2\right] &= \mathbb{E}_k\left[\|w^k - w^* - \alpha_k g^k\|_2^2\right] \\ &= \|w^k-w^*\|_2^2 - 2\alpha_k\mathbb{E}_k\left[g^k\right]^\top(w^k-w^*) + \alpha_k^2\mathbb{E}_k\left[\|g^k\|_2^2\right] \\ &\overset{(3.8)+(3.34)+\text{Rem. 3.14}}{\leq} \|w^k-w^*\|_2^2 - 2\alpha_k\nabla f(w^k)^\top(w^k-w^*) \\ &\quad +2\alpha_k^2\left[2\mathcal{L}(f(w^k)-f(w^*)) + 2\mathcal{L}(f(z^k)-f(w^*))\right] \\ &\overset{(3.6)}{\leq} (1-\alpha_k\mu)\|w^k-w^*\|_2^2 - 2\alpha_k(1-2\alpha_k\mathcal{L})\left(f(w^k)-f(w^*)\right) \\ &\quad +4\alpha_k^2\mathcal{L}(f(z^k)-f(w^*)) \\ &\overset{(3.44)}{=} (1-\alpha_k\mu)\|w^k-w^*\|_2^2 - 2\alpha_k(1-2\alpha_k\mathcal{L})\left(f(w^k)-f(w^*)\right) \\ &\quad +p\left(\frac{3}{2}-p\right)\mathcal{P}^k.\end{aligned}$$

Hence we have, taking total expectation and noticing that the variables $\alpha_k$ and $w^k$ are independent,

$$\begin{aligned}\mathbb{E}\left[\left\|w^{k+1}-w^*\right\|_2^2\right] &\leq (1-\mathbb{E}\left[\alpha_k\right]\mu)\,\mathbb{E}\left[\left\|w^k-w^*\right\|_2^2\right] - 2\mathbb{E}\left[\alpha_k(1-2\alpha_k\mathcal{L})\right]\mathbb{E}\left[f(w^k)-f(w^*)\right] \\ &\quad +p\left(\frac{3}{2}-p\right)\mathbb{E}\left[\mathcal{P}^k\right]. \end{aligned} \tag{3.52}$$

We have also have

$$\begin{aligned}\mathbb{E}_k\left[\mathcal{P}^{k+1}\right] &= (1-p)\frac{8(1-p)\alpha_k^2\mathcal{L}}{p(3-2p)}\left(f(z^k)-f(w^*)\right) + p\frac{8\alpha^2\mathcal{L}}{p(3-2p)}\left(f(w^k)-f(w^*)\right) \\ &= (1-p)^2\mathcal{P}^k + \frac{8\alpha^2\mathcal{L}}{3-2p}\left(f(w^k)-f(w^*)\right).\end{aligned}$$

Hence, taking total expectation gives

$$\mathbb{E}\left[\mathcal{P}^{k+1}\right] = (1-p)^2\mathbb{E}\left[\mathcal{P}^k\right] + \frac{8\alpha^2\mathcal{L}}{3-2p}\mathbb{E}\left[f(w^k)-f(w^*)\right] \tag{3.53}$$

Consequently,

$$\begin{aligned}\mathbb{E}\left[\phi^{k+1}\right] &\overset{(3.44)+(3.52)+(3.53)}{\leq} & (1-\mathbb{E}\left[\alpha_k\right]\mu)\,\mathbb{E}\left[\left\|w^k-w^*\right\|_2^2\right] \\ & & -2\left(\mathbb{E}\left[\alpha_k(1-2\alpha_k\mathcal{L})\right]-4\frac{\alpha^2\mathcal{L}}{3-2p}\right)\mathbb{E}\left[f(w^k)-f(w^*)\right] \\ & & +\left(1-\frac{p}{2}\right)\mathbb{E}\left[\mathcal{P}^k\right] \\ &= & (1-\mathbb{E}\left[\alpha_k\right]\mu)\,\mathbb{E}\left[\left\|w^k-w^*\right\|_2^2\right] \\ & & -2\left(\mathbb{E}\left[\alpha_k\right]-2\left(\mathbb{E}\left[\alpha_k^2\right]+\frac{2}{3-2p}\alpha^2\right)\mathcal{L}\right)\mathbb{E}\left[f(w^k)-f(w^*)\right] \\ & & +\left(1-\frac{p}{2}\right)\mathbb{E}\left[\mathcal{P}^k\right]. \end{aligned} \tag{3.54}$$

From Lemma 3.33, we have $\mathbb{E}\left[\alpha_k\right]=\frac{(1-p)^{\frac{3k+2}{2}}(1-\sqrt{1-p})+p}{1-(1-p)^{\frac{3}{2}}}\alpha$, and we can show that for all $k$

$$\mathbb{E}\left[\alpha_k\right]\geq\frac{2}{3}\alpha, \tag{3.55}$$

Letting $q=1-p$ we have that

$$\begin{aligned}\frac{(1-p)^{\frac{3k+2}{2}}(1-\sqrt{1-p})+p}{1-(1-p)^{\frac{3}{2}}} &= \frac{q^{\frac{3k+2}{2}}(1-\sqrt{q})+1-q}{1-q^{\frac{3}{2}}} \\ &= q^{\frac{3k+2}{2}}\frac{1-\sqrt{q}}{1-q^{3/2}}+\frac{1-q}{1-q^{3/2}} \\ &\geq \frac{1-q}{1-q^{3/2}} \\ &\geq \frac{2}{3},\quad \forall q\in[0,1]\ . \end{aligned}$$

Consequently,

$$\begin{aligned}\mathbb{E}\left[\phi^{k+1}\right] &\overset{(3.52)+(3.53)+(3.55)}{\leq} & \left(1-\frac{2}{3}\alpha\mu\right)\mathbb{E}\left[\left\|w^k-w^*\right\|_2^2\right] \\ & & -2\left(\mathbb{E}\left[\alpha_k\right]-2\left(\mathbb{E}\left[\alpha_k^2\right]+\frac{2}{3-2p}\alpha^2\right)\mathcal{L}\right)\mathbb{E}\left[f(w^k)-f(w^*)\right] \\ & & +\left(1-\frac{p}{2}\right)\mathbb{E}\left[\mathcal{P}^k\right]. \end{aligned} \tag{3.56}$$

To declutter the notations, let us define

$$a_k \overset{\text{def}}{=} \frac{(1-p)^{\frac{3k+2}{2}}(1-\sqrt{1-p})+p}{1-(1-p)^{\frac{3}{2}}} \tag{3.57}$$

$$b_k \overset{\text{def}}{=} \frac{1+(1-p)^{2k+1}}{2-p} \tag{3.58}$$

so that $\mathbb{E}\left[\alpha_k\right]=a_k\alpha$ and $\mathbb{E}\left[\alpha_k^2\right]=b_k\alpha^2$. Then (3.56) becomes

$$\begin{aligned}\mathbb{E}\left[\phi^{k+1}\right] &\leq & \left(1-\frac{2}{3}\alpha\mu\right)\mathbb{E}\left[\left\|w^k-w^*\right\|_2^2\right] \\ & & -2\alpha\left(a_k-2\alpha\left(b_k+\frac{2}{3-2p}\right)\mathcal{L}\right)\mathbb{E}\left[f(w^k)-f(w^*)\right] \\ & & +\left(1-\frac{p}{2}\right)\mathbb{E}\left[\mathcal{P}^k\right]. \end{aligned} \tag{3.59}$$

Next we would like to drop the second term in (3.59). For this we need to guarantee that

$$a_k - 2\alpha\mathcal{L}\left(b_k + \frac{2}{3-2p}\right) \geq 0 \tag{3.60}$$

Let $q \overset{\text{def}}{=} 1-p$ so that the above becomes

$$\frac{q^{\frac{3k+2}{2}}(1-\sqrt{q})+1-q}{1-q^{\frac{3}{2}}} - 2\alpha\mathcal{L}\left(\frac{1+q^{2k+1}}{1+q} + \frac{2}{1+2q}\right) \geq 0.$$

In other words, after dividing through by $\left(\frac{1+q^{2k+1}}{1+q} + \frac{2}{1+2q}\right)$ and re-arranging, we require that

$$\begin{aligned} 2\alpha\mathcal{L} &\leq \frac{\frac{q^{\frac{3k+2}{2}}(1-\sqrt{q})+1-q}{1-q^{\frac{3}{2}}}}{\frac{1+q^{2k+1}}{1+q} + \frac{2}{1+2q}} \\ &= \frac{\frac{q^{\frac{3k+2}{2}}(1-\sqrt{q})+1-q}{1-q^{\frac{3}{2}}}}{\frac{(1+q^{2k+1})(1+2q)+2(1+q)}{(1+q)(1+2q)}} \\ &= \frac{q^{\frac{3k+2}{2}}(1-\sqrt{q})+1-q}{q^{2k+1}(1+2q)+3+4q}\,\frac{(1+q)(1+2q)}{1-q^{\frac{3}{2}}}\ . \end{aligned} \tag{3.61}$$

We are now going to show that

$$\frac{q^{\frac{3k+2}{2}}(1-\sqrt{q})+1-q}{q^{2k+1}(1+2q)+3+4q} \geq \frac{1-q}{3+4q}. \tag{3.62}$$

Indeed, multiplying out the denominators of the above gives

$$\begin{aligned} F(q) &\overset{\text{def}}{=} (3+4q)\left(q^{\frac{3k+2}{2}}(1-\sqrt{q})+1-q\right) - (1-q)\left(q^{2k+1}(1+2q)+3+4q\right) \\ &= q^{\frac{3k+2}{2}}(1-\sqrt{q})(3+4q) - q^{2k+1}(1+2q)(1-q) \\ &= q^{\frac{3k+2}{2}}(1-\sqrt{q})\left(3+4q-q^{\frac{k}{2}}(1+2q)(1+\sqrt{q})\right). \end{aligned}$$

And since $q^{\frac{k}{2}} \leq 1$, we have

$$\begin{aligned} F(q) &\geq q^{\frac{3k+2}{2}}(1-\sqrt{q})\left(3+4q-(1+2q)(1+\sqrt{q})\right) \\ &= 2q^{\frac{3k+2}{2}}(1-\sqrt{q})(1-q\sqrt{q}) \\ &\geq 0. \end{aligned}$$

As a result (3.62) holds. And thus if

$$2\alpha\mathcal{L} \leq \frac{1-q}{3+4q}\frac{(1+q)(1+2q)}{1-q^{\frac{3}{2}}}$$

holds, then (3.61) is verified for all $k$. This is why we impose the upper bound on the step size given in (3.45), which ensures that (3.60) is satisfied. Finally, this condition being verified, we get that

$$\begin{aligned} \mathbb{E}\left[\phi^{k+1}\right] &\overset{(3.59)+(3.60)}{\leq} \left(1-\frac{2}{3}\alpha\mu\right)\mathbb{E}\left[\left\|w^k-w^*\right\|_2^2\right] + \left(1-\frac{p}{2}\right)\mathbb{E}\left[\mathcal{P}^k\right] \\ &\leq \beta\mathbb{E}\left[\phi^k\right], \end{aligned} \tag{3.63}$$

where $\beta = \max\left\{1-\frac{2}{3}\alpha\mu, 1-\frac{p}{2}\right\}$. □

**Corollary 3.34.** *Consider the setting of Algorithm 10 and suppose that we use b–nice sampling. Let* $\alpha = \frac{1}{2\zeta_p \mathcal{L}(b)}$. *We have that the total complexity of finding an* $\epsilon > 0$ *approximate solution that satisfies* $\mathbb{E}\left[\left\|w^k - w^*\right\|_2^2\right] \leq \epsilon\, \phi^0$ *is*

$$C_p(b) \overset{def}{=} 2(2b + pn) \max\left\{\frac{3\zeta_p}{2}\frac{\mathcal{L}(b)}{\mu}, \frac{1}{p}\right\} \log\left(\frac{1}{\epsilon}\right). \tag{3.64}$$

*Proof.* We have that

$$\mathbb{E}\left[\phi^k\right] \leq \beta^k \phi^0,$$

where $\beta = \max\left\{1 - \frac{1}{3\zeta_p}\frac{\mu}{\mathcal{L}(b)}, 1 - \frac{p}{2}\right\}$. Hence using Lemma 3.35, we have that the iteration complexity for an $\epsilon > 0$ approximate solution that verifies $\mathbb{E}\left[\phi^k\right] \leq \epsilon\phi^0$ is

$$2 \max\left\{\frac{3\zeta_p}{2}\frac{\mathcal{L}(b)}{\mu}, \frac{1}{p}\right\} \log\left(\frac{1}{\epsilon}\right).$$

For the total complexity, one can notice that in expectation, we compute $2b + pn$ stochastic gradients at each iteration.

**Lemma 3.35** (Complexity bounds)**.** *Consider the sequence* $(\alpha_k)_k \in \mathbb{R}_+$ *of positive scalars that converges to* 0 *according to*

$$\alpha_k \leq \rho^k \alpha_0,$$

*where* $\rho \in [0, 1)$. *For a given* $\epsilon \in (0, 1)$, *we have that*

$$k \geq \frac{1}{1-\rho} \log\left(\frac{1}{\epsilon}\right) \implies \alpha_k \leq \epsilon\alpha_0. \tag{3.65}$$

□

## 3.6 Optimal parameter settings: loop, mini-batch and step sizes

In this section, we restrict our analysis to $b$–nice sampling. First, we determine the optimal loop size for Algorithm 9. Then, we examine the optimal mini-batch and step sizes for particular choices of the inner loop size $m$ for Algorithm 9 and of the probability $p$ of updating the reference point in Algorithm 10, that play analogous roles. Note that the steps used in our algorithms depend on $b$ through the expected smoothness constant $\mathcal{L}(b)$ and the expected residual constant $\rho(b)$. Hence, optimizing the total complexity in the mini-batch size also determines the optimal step size.

### 3.6.1 Optimal loop size for Free-SVRG

Here we determine the optimal value of $m$ for a fixed batch size $b$, denoted by $m^*(b)$, which minimizes the total complexity (3.42).

**Proposition 3.36.** *The loop size that minimizes* (3.42) *and the resulting total complexity is given by*

$$m^*(b) = \frac{\mathcal{L}(b) + 2\rho(b)}{\mu} \quad \text{and} \quad C_{m^*}(b) = 2\left(n + 2b\frac{\mathcal{L}(b) + 2\rho(b)}{\mu}\right)\log\left(\frac{1}{\epsilon}\right). \tag{3.66}$$

**Remark 3.37.** *Examining the total complexities of Algorithms 9 and 10, given in* (3.42) *and* (3.64), *we can see that, when setting* $p = 1/m$ *in Algorithm 10, these complexities only differ by constants.*

*Proof.* Dropping the $\log(1/\epsilon)$ for brevity, we distinguish two cases, $m \geq \frac{2(\mathcal{L}(b)+2\rho(b))}{\mu}$ and $m \leq \frac{2(\mathcal{L}(b)+2\rho(b))}{\mu}$.

1. $m \geq \frac{2(\mathcal{L}(b)+2\rho(b))}{\mu}$: Then $C_m(b) = 2(n+2bm)$, and hence we should use the smallest $m$ possible, that is, $m = \frac{2(\mathcal{L}(b)+2\rho(b))}{\mu}$.
2. $m \leq \frac{2(\mathcal{L}(b)+2\rho(b))}{\mu}$: Then $C_m(b) = \frac{2(n+2bm)}{m}\frac{2(\mathcal{L}(b)+2\rho(b))}{\mu} = 2\left(\frac{n}{m}+2b\right)\frac{2(\mathcal{L}(b)+2\rho(b))}{\mu}$. Hence $C_m(b)$ is decreasing in $m$ and we should then use the highest possible value for $m$, that is $m = \frac{2(\mathcal{L}(b)+2\rho(b))}{\mu}$.

The result now follows by substituting $m = \frac{2(\mathcal{L}(b)+2\rho(b))}{\mu}$ into (3.42). □

For example when $b=1$, we have that $m^*(1) = 3L_{\max}/\mu$ and $C_{m^*}(1) = O((n + L_{\max}/\mu)\log(1/\epsilon))$, which is the same complexity as achieved by the range of $m$ values given in Corollary 3.30. Thus, as we also observed in Corollary 3.30, the total complexity is not very sensitive to the choice of $m$, and $m = n$ is a perfectly safe choice as it achieves the same complexity as $m^*$. We also confirm this numerically with a series of experiments in Section 3.8.2.2.

### 3.6.2 Optimal mini-batch size for L-SVRG-D

By using a similar proof as for Proposition 3.40, we derive the following result.

**Proposition 3.38.** *Note* $b^* \overset{def}{=} \arg\min_{b\in[n]} C_p(b)$, *where* $C_p(b)$ *is defined in* (3.64). *For the widely used choice* $p = \frac{1}{n}$, *we have that*

$$b^* = \begin{cases} 1 & \text{if } n \geq \frac{3\zeta_{1/n}}{2}\frac{L_{\max}}{\mu} \\ \lfloor \min(\tilde{b}, \hat{b}) \rfloor & \text{if } \frac{3\zeta_{1/n}}{2}\frac{L}{\mu} < n < \frac{3\zeta_{1/n}}{2}\frac{L_{\max}}{\mu} \\ \lfloor \hat{b} \rfloor & \text{otherwise, if } n \leq \frac{3\zeta_{1/n}}{2}\frac{L}{\mu} \end{cases}, \tag{3.67}$$

*where* $\zeta_p$ *is defined in* (3.45) *for* $p \in (0,1]$ *and:*

$$\hat{b} = \sqrt{\frac{n}{2}\frac{L_{\max}-L}{nL-L_{\max}}},\ \tilde{b} = \frac{\frac{3\zeta_p}{2}n(L_{\max}-L)}{\mu n(n-1) - \frac{3\zeta_p}{2}(nL-L_{\max})}.$$

Because $\zeta_p$ depends on $p$, optimizing the total complexity with respect to $b$ for the case $p = \frac{b}{n}$ is extremely cumbersome. Thus, we restrain our study for the optimal mini-batch sizes for Algorithm 10 to the case where $p = \frac{1}{n}$.

*Proof.* For brevity, we temporarily drop the term $\log\left(\frac{1}{\epsilon}\right)$ in $C_p(b)$ defined in Equation (3.64). Hence, we want to find, for different values of $m$:

$$b^* = \arg\min_{b\in[n]} C_{1/n}(b) := 2\,(2b+1)\max\{\pi(b), m\}\ , \tag{3.68}$$

where $\pi(b) \overset{\text{def}}{=} \frac{3\zeta_p}{2}\frac{\mathcal{L}(b)}{\mu}$. We have

$$\pi(b) = \frac{3\zeta_p}{2}\frac{1}{\mu(n-1)}\left(\frac{n(L_{\max}-L)}{b} + nL - L_{\max}\right). \tag{3.69}$$

Since $L_{\max} \geq L$, $\pi(b)$ is a decreasing function on $[1,n]$. We distinguish three cases:

- if $n > \pi(1) = \frac{3\zeta_p}{2}\frac{L_{\max}}{\mu}$, then for all $b \in [1,n], n > \pi(b)$. Hence,

  $$C_n(b) = 2(2b+1)n.$$

  $C_{1/n}(b)$ is an increasing function of $b$. Hence

  $$b^* = 1.$$

- if $n < \pi(n) = \frac{3\zeta_p}{2}\frac{L}{\mu}$, then for all $b \in [1,n], n < \pi(b)$. Hence,

$$C_{1/n}(b) = 2(2b+1)\pi(b).$$

Now, consider the function

$$\begin{aligned} G(b) &\overset{\text{def}}{=} (2b+1)\pi(b) \\ &= \frac{3\zeta_p}{2}\frac{1}{\mu(n-1)}\left(2(nL - L_{\max})b + \frac{n(L_{\max}-L)}{b}\right) + \Omega, \end{aligned}$$

where $\Omega$ replaces constants which don't depend on $b$. The first derivative of $G(b)$ is

$$G'(b) = \frac{3\zeta_p}{2}\frac{1}{\mu(n-1)}\left(-\frac{n(L_{\max}-L)}{b^2} + 2(nL - L_{\max})\right),$$

and its second derivative is

$$G''(b) = \frac{3\zeta_p n(L_{\max}-L)}{\mu(n-1)b^3} \geq 0.$$

$G(b)$ is a convex function, and we can find its minimizer by setting its first derivative to zero. This minimizer is

$$\hat{b} \overset{\text{def}}{=} \sqrt{\frac{n}{2}\frac{L_{\max}-L}{nL - L_{\max}}}.$$

Indeed, from the following lemma, we have $nL \geq L_{\max}$.

**Lemma 3.39.** *Consider convex and $L_i$–smooth functions $f_i$, where $L_i \geq 0$ for all $i \in [n]$, and define $L_{\max} = \max_{i\in[n]} L_i$. Let*

$$f(w) = \frac{1}{n}\sum_{i=1}^n f_i(w)$$

*for any $w \in \mathbb{R}^d$. Suppose that $f$ is $L$–smooth, where $L \geq 0$. Then,*

$$nL \geq L_{\max}. \tag{3.70}$$

*Proof.* Let $w, y \in \mathbb{R}^d$. Since $f$ is $L$–smooth, we have

$$f(w) \leq f(y) + \nabla f(y)^\top (w-y) + \frac{L}{2}\|w-y\|_2^2.$$

Hence, multiplying by $n$ on both sides,

$$\sum_{i=1}^n f_i(w) \leq \sum_{i=1}^n f_i(y) + \sum_{i=1}^n \nabla f_i(y)^\top (w-y) + \frac{nL}{2}\|w-y\|_2^2.$$

Rearranging this inequality,

$$\sum_{i=1}^n \left(f_i(w) - f_i(y) - \nabla f_i(y)^\top (w-y)\right) \leq \frac{nL}{2}\|w-y\|_2^2. \tag{3.71}$$

Since the functions $f_i$ are convex, we have for all $i \in [n]$,

$$f_i(w) - f_i(y) - \nabla f_i(y)^\top (w-y) \geq 0.$$

Then, as a consequence of (3.71), we have that for all $i \in [n]$,

$$f_i(w) - f_i(y) - \nabla f_i(y)^\top (w-y) \leq \frac{nL}{2} \|w-y\|_2^2 .$$

Rearranging this inequality,

$$f_i(w) \leq f_i(y) + \nabla f_i(y)^\top (w-y) + \frac{nL}{2} \|w-y\|_2^2 .$$

But since for all $i \in [n]$, $L_i$ is the smallest positive constant that verifies

$$f_i(w) \leq f_i(y) + \nabla f_i(y)^\top (w-y) + \frac{L_i}{2} \|w-y\|_2^2 ,$$

we have for all $i \in [n], L_i \leq nL$. Hence $L_{\max} \leq nL$. □

Thus, in this case, $C_{1/n}(b)$ is a convex function and its minimizer is

$$b^* = \lfloor \hat{b} \rfloor .$$

- if $\frac{3\zeta_p}{2}\frac{L}{\mu} = \pi(n) \leq n \leq \pi(1) = \frac{3\zeta_p}{2}\frac{L_{\max}}{\mu}$. Then there exists $b \in [1, n]$ such that $\pi(b) = n$ and its expression is given by

$$\tilde{b} = \frac{\frac{3\zeta_p}{2} n(L_{\max} - L)}{\mu n(n-1) - \frac{3\zeta_p}{2}(nL - L_{\max})}.$$

Consequently, the function $C_n(b)$ is decreasing on $\left[1, \min\left\{\tilde{b}, \hat{b}\right\}\right]$ and increasing on $\left[\min\left\{\tilde{b}, \hat{b}\right\}, n\right]$. Hence,

$$b^* = \lfloor \min\left\{\tilde{b}, \hat{b}\right\} \rfloor .$$

□

### 3.6.3 Optimal mini-batch and step sizes

In the following proposition, we determine the optimal mini-batch and step sizes for two practical choices of the size of the loop $m$.

**Proposition 3.40.** *Let $b^* \overset{def}{=} \arg\min_{b \in [n]} C_m(b)$, where $C_m(b)$ is defined in (3.42). For the widely used choice $m = n$, we have that $b^*$ is given by Table 3.1. For another widely used choice $m = n/b$, which allows to make a full pass over the data set during each inner loop, we have*

$$b^* = \begin{cases} \lfloor \bar{b} \rfloor & \text{if } n > \frac{3L_{\max}}{\mu} \\ 1 & \text{if } \frac{3L_{\max}}{L} < n \leq \frac{3L_{\max}}{\mu} \\ n & \text{otherwise, if } n \leq \frac{3L_{\max}}{L} \end{cases} , \quad \text{where } \bar{b} \overset{def}{=} \frac{n(n-1)\mu - 3n(L_{\max} - L)}{3(nL - L_{\max})} . \tag{3.72}$$

*Proof.* Recall that have from Lemma 3.15:

$$\mathcal{L}(b) = \frac{1}{b}\frac{n-b}{n-1} L_{\max} + \frac{n}{b}\frac{b-1}{n-1} L, \tag{3.73}$$

$$\rho(b) = \frac{1}{b}\frac{n-b}{n-1} L_{\max}. \tag{3.74}$$

For brevity, we temporarily drop the term $\log\left(\frac{1}{\epsilon}\right)$ in $C_m(b)$ defined in Equation (3.42). Hence, we want

to find, for different values of $m$:

$$b^* = \arg\min_{b\in[n]} C_m(b) := 2\left(\frac{n}{m} + 2b\right)\max\{\kappa(b), m\}, \tag{3.75}$$

where $\kappa(b) \overset{\text{def}}{=} \frac{\mathcal{L}(b)+2\rho(b)}{\mu}$.

**When $m = n$.** In this case we have

$$C_n(b) \overset{(3.42)}{=} 2(2b+1)\max\{\kappa(b), n\}, \tag{3.76}$$

Writing $\kappa(b)$ explicitly:

$$\kappa(b) = \frac{1}{\mu(n-1)}\left((3L_{\max} - L)\frac{n}{b} + nL - 3L_{\max}\right).$$

Since $3L_{\max} > L$, $\kappa(b)$ is a decreasing function of $b$. In the light of this observation, we will determine the optimal mini-batch size. The upcoming analysis is summarized in Table 3.1.

We distinguish three cases:

- If $n \le \frac{L}{\mu}$: then $\kappa(n) = \frac{L}{\mu} \ge n$. Since $\kappa(b)$ is decreasing, this means that for all $b \in [n], \kappa(b) \ge n$. Consequently, $C_n(b) = 2(2b+1)\kappa(b)$. Differentiating twice:

  $$C_n^{''}(b) = \frac{4}{\mu(n-1)}\frac{(3L_{\max} - L)n}{b^3} > 0.$$

  Hence $C_n(b)$ is a convex function. Now examining its first derivative:

  $$C_n^{'}(b) = \frac{2}{\mu(n-1)}\left(-\frac{(3L_{\max} - L)n}{b^2} + 2(nL - 3L_{\max})\right),$$

  we can see that:

  - If $n \le \frac{3L_{\max}}{L}$, $C_n(b)$ is a decreasing function, hence

    $$b^* = n.$$

  - If $n > \frac{3L_{\max}}{L}$, $C_n(b)$ admits a minimizer, which we can find by setting its first derivative to zero. The solution is

    $$\hat{b} \overset{\text{def}}{=} \sqrt{\frac{n}{2}\frac{3L_{\max} - L}{nL - 3L_{\max}}}.$$

    Hence,

    $$b^* = \lfloor \hat{b} \rfloor$$

- If $n \ge \frac{3L_{\max}}{\mu}$, then $\kappa(1) = 3\frac{L_{\max}}{\mu}$. Since $\kappa(b)$ is decreasing, this means that for all $b \in [n], \kappa(b) \le n$. Hence, $C_n(b) = 2(2b+1)n$. $C_n(b)$ is an increasing function of $b$. Therefore,

  $$b^* = 1.$$

- If $\frac{L}{\mu} < n < \frac{3L_{\max}}{\mu}$, we have $\kappa(1) > n$ and $\kappa(n) < n$. Hence there exists $\tilde{b} \in [1,\, n]$ such that $\kappa(b) = n$, and it is given by

  $$\tilde{b} \overset{\text{def}}{=} \frac{(3L_{\max} - L)n}{n(n-1)\mu - nL + 3L_{\max}}. \tag{3.77}$$

Define $G(b) \overset{\text{def}}{=} (2b+1)\kappa(b)$. Then,

$$\arg\min_{b\in[1,\,n]} G(b) = \begin{cases} n & \text{if } n \leq \frac{3L_{\max}}{L}, \\ \hat{b} & \text{if } n > \frac{3L_{\max}}{L}. \end{cases} \tag{3.78}$$

As a result, we have that

- if $n \leq \frac{3L_{\max}}{L}$, $G(b)$ is decreasing on $[1,n]$, hence $C_n(b)$ is decreasing on $[1,\tilde{b}]$ and increasing on $[\tilde{b},n]$. Then,
$$b^* = \lfloor \tilde{b} \rfloor.$$
- if $n > \frac{3L_{\max}}{L}$, $G(b)$ is decreasing on $[1,\hat{b}]$ and increasing on $[\hat{b},n]$. Hence $C_n(b)$ is decreasing on $[1,\min\{\hat{b},\tilde{b}\}]$ and increasing on $[\min\{\hat{b},\tilde{b}\},1]$. Then,
$$b^* = \lfloor \min\{\hat{b},\tilde{b}\} \rfloor.$$

To summarize, we have for $m = n$,

$$b^* \quad = \quad \begin{cases} 1 & \text{if } n \geq \frac{3L_{\max}}{\mu} \\ \lfloor \min(\tilde{b},\hat{b}) \rfloor & \text{if } \max\{\frac{L}{\mu}, \frac{3L_{\max}}{L}\} < n < \frac{3L_{\max}}{\mu} \\ \lfloor \hat{b} \rfloor & \text{if } \frac{3L_{\max}}{L} < n < \frac{L}{\mu} \\ \lfloor \tilde{b} \rfloor & \text{if } \frac{L}{\mu} < n \leq \frac{3L_{\max}}{L} \\ n & \text{otherwise, if } n \leq \min\{\frac{L}{\mu}, \frac{3L_{\max}}{L}\} \end{cases} \tag{3.79}$$

**When $m = n/b$.** In this case we have

$$C_m(b) \overset{(3.42)}{=} 6\max\{b\kappa(b), n\},$$

with

$$b\kappa(b) = \frac{1}{\mu(n-1)}\left((3L_{\max} - L)n + (nL - 3L_{\max})b\right),$$

and thus $\kappa(1) = \frac{3L_{\max}}{\mu}$ and $n\kappa(n) = \frac{nL}{\mu} \geq n$. We distinguish two cases:

- if $n \leq \frac{3L_{\max}}{L}$, then $b\kappa(b)$ is decreasing in $b$. Since $n\kappa(n) \geq n$, $C_m(b) = 6b\kappa(b)$, thus $C_m(b)$ is decreasing in $b$, hence
$$b^* = n$$
- if $n > \frac{3L_{\max}}{L}$, $b\kappa(b)$ is increasing in $b$. Thus,
  - if $n \leq \frac{3L_{\max}}{\mu} = \kappa(1)$, then $C_m(b) = 6b\kappa(b)$. Hence $b^* = 1$.
  - if $n > \frac{3L_{\max}}{\mu}$, using the definition of $\tilde{b}$ in Equation (3.77), we have that
$$C_m(b) = \begin{cases} 6n & \text{for } b \in [1,\bar{b}] \\ 6b\kappa(b) & \text{for } b \in [\bar{b},n] \end{cases},$$
where
$$\bar{b} = \frac{n(n-1)\mu - (3L_{\max} - L)n}{nL - 3L_{\max}}$$
is the batch size $b \in [n]$ which verifies $b\kappa(b) = n$. Hence $b^*$ can be any point in $\{1,\ldots,\lfloor\bar{b}\rfloor\}$. In light of shared memory parallelism, $b^* = \lfloor\bar{b}\rfloor$ would be the most practical choice.

□

Previously, theory showed that the total complexity would increase as the mini-batch size increases, and thus established that single-element sampling was optimal. However, notice that for $m = n$ and $m = n/b$,

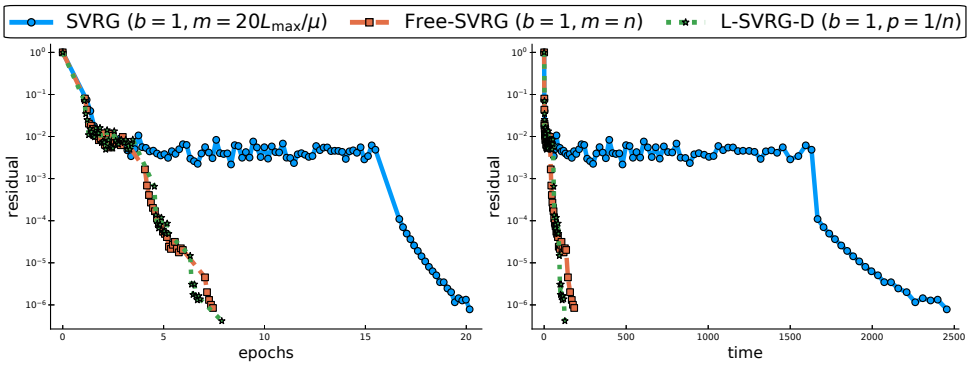



**Figure 3.2:** *Comparison of theoretical variants of SVRG without mini-batching ($b = 1$) on the ijcnn1 data set.*

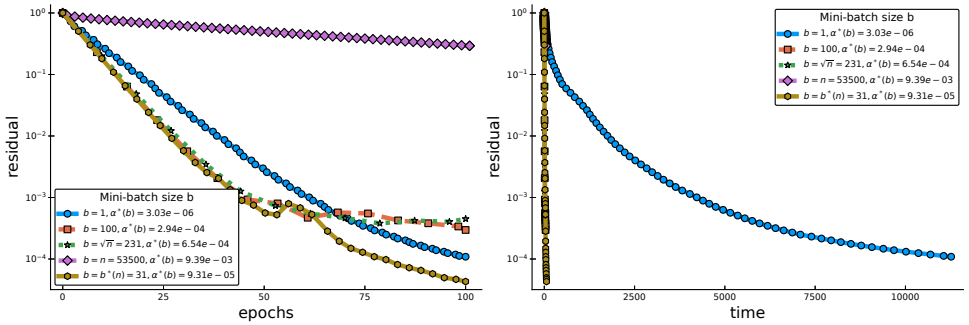



**Figure 3.3:** *Optimality of our mini-batch size $b^*$ given in Table 3.1 for Free-SVRG on the slice data set.*

the usual choices for $m$ in practice, the optimal mini-batch size is different than 1 for a range of problem settings. Since our algorithms are closer to the SVRG variants used in practice, we argue that our results explain why practitioners experiment that mini-batching works, as we verify in the next section.

## 3.7 Experiments

We performed a series of experiments on data sets from LIBSVM [26] and the UCI repository [39], to validate our theoretical findings. We tested $l_2$–regularized logistic regression on *ijcnn1* and *real-sim*, and ridge regression on *slice* and *YearPredictionMSD*. We used two choices for the regularizer: $\lambda = 10^{-1}$ and $\lambda = 10^{-3}$. All of our code is implemented in `Julia 1.0`. Due to lack of space, most figures have been relegated to Section 3.8 in the supplementary material.

**Practical theory.** Our first round of experiments aimed at verifying if our theory does result in efficient algorithms. Indeed, we found that *Free-SVRG* and *L-SVRG-D* with the parameter setting given by our theory are often faster than SVRG with settings suggested by the theory in [71], that is $m = 20L_{\max}/\mu$ and $\alpha = 1/10L_{\max}$. See Figure 3.2, and Section 3.8.1 for more experiments comparing different theoretical parameter settings.

**Optimal mini-batch size.** We also confirmed numerically that when using *Free-SVRG* with $m = n$, the optimal mini-batch size $b^*$ derived in Table 3.1 was highly competitive as compared to the range of mini-batch sizes $b \in \{1, 100, \sqrt{n}, n\}$. See Figure 3.3 and several more such experiments in Section 3.8.2.1. We also explore the optimality of our $m^*$ in more experiments in Section 3.8.2.2.

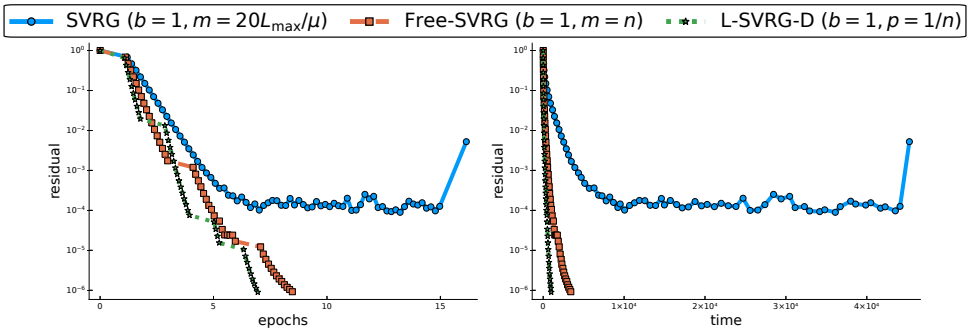
## 3.8 Additional experiments

### 3.8.1 Comparison of theoretical variants of SVRG

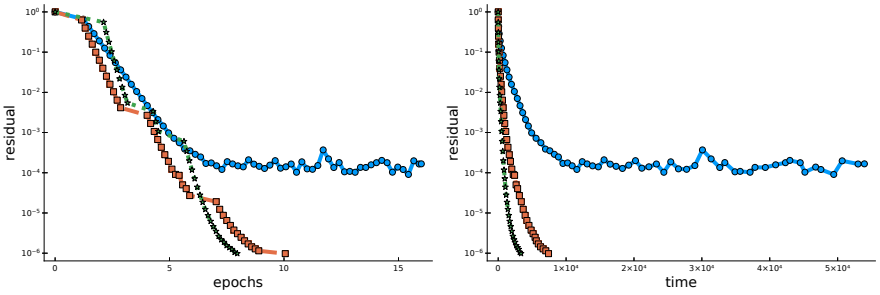
In this series of experiments, we compare the performance of the SVRG algorithm with the settings of [71] against *Free-SVRG* and *L-SVRG-D* with the settings given by our theory.

#### 3.8.1.1 Experiment 1.a: comparison without mini-batching ($b = 1$)

A widely used choice for the size of the inner loop is $m = n$. Since our algorithms allow for a free choice of the size of the inner loop, we set $m = n$ for *Free-SVRG* and $p = 1/n$ for *L-SVRG-D*, and use a mini-batch size $b = 1$. For vanilla *SVRG*, we set $m$ to its theoretical value $20L_{\max}/\mu$ as in [24]. See Figures 3.4, 3.5, 3.6 and 3.7. We can see that *Free-SVRG* and *L-SVRG-D* often outperform the SVRG algorithm [71]. It is worth noting that, in Figure 3.4a, 3.6a and 3.7 the classic version of SVRG can lead to increase of the suboptimality when entering the outer loop. This is due to the fact that the reference point is set to a weighted average of the iterates of the inner loop, instead of the last iterate.

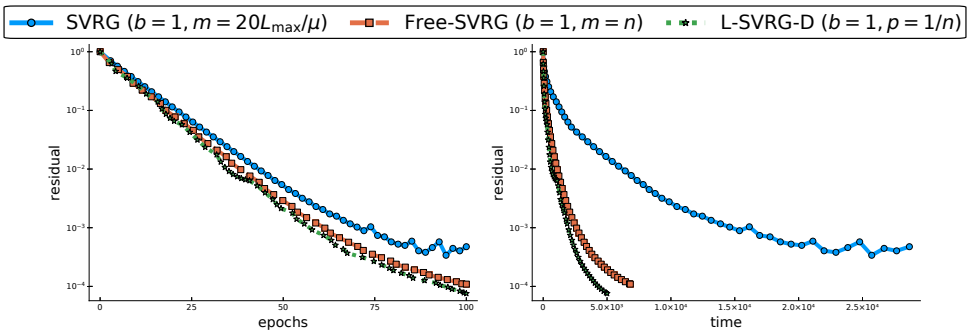



**(a)** $\lambda = 10^{-1}$

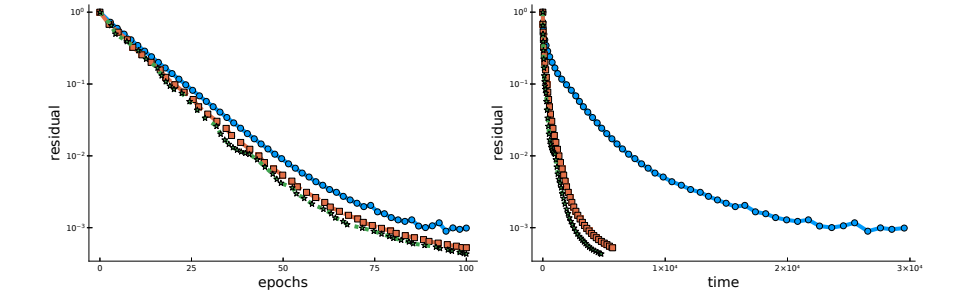



**(b)** $\lambda = 10^{-3}$

**Figure 3.4:** *Comparison of theoretical variants of SVRG without mini-batching ($b = 1$) on the YearPredictionMSD data set.*

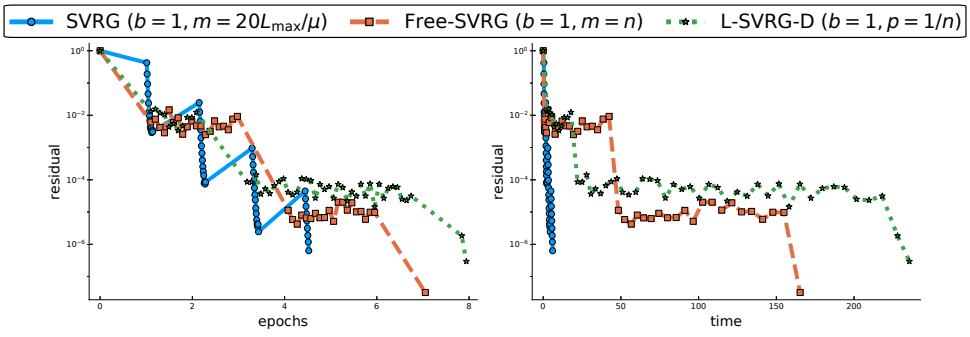



**(a)** $\lambda = 10^{-1}$

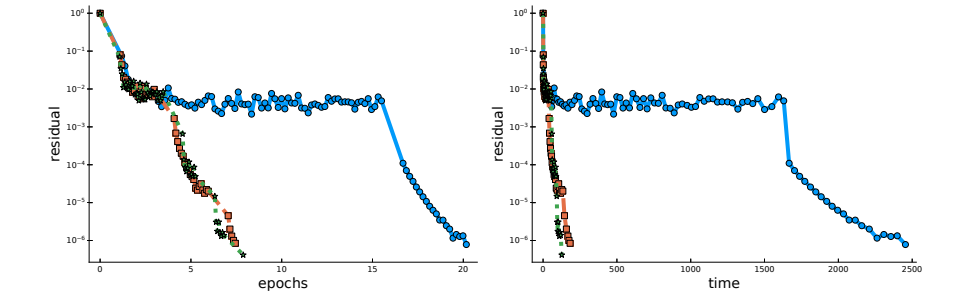



**(b)** $\lambda = 10^{-3}$

**Figure 3.5:** *Comparison of theoretical variants of SVRG without mini-batching ($b = 1$) on the slice data set.*

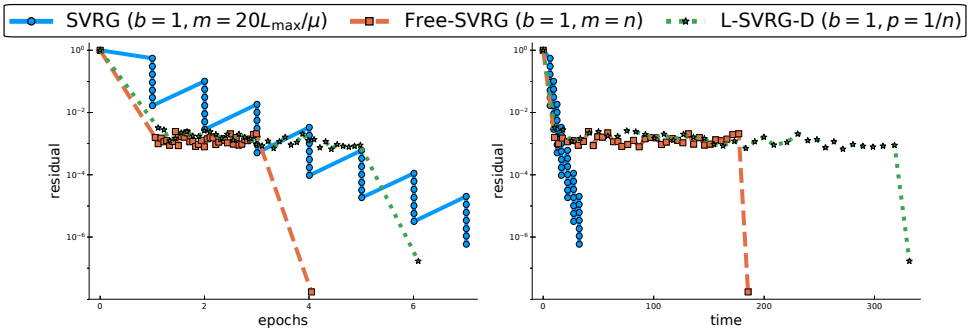



**(a)** $\lambda = 10^{-1}$

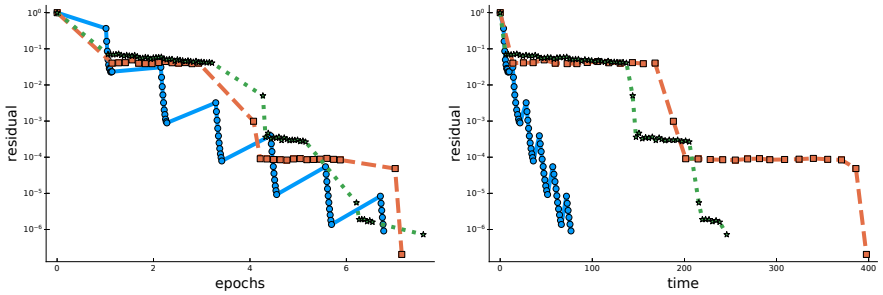



**(b)** $\lambda = 10^{-3}$

**Figure 3.6:** *Comparison of theoretical variants of SVRG without mini-batching ($b = 1$) on the ijcnn1 data set.*




**(a)** $\lambda = 10^{-1}$

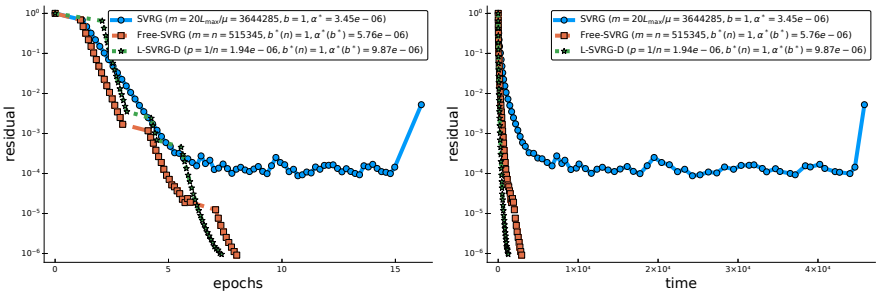



**(b)** $\lambda = 10^{-3}$

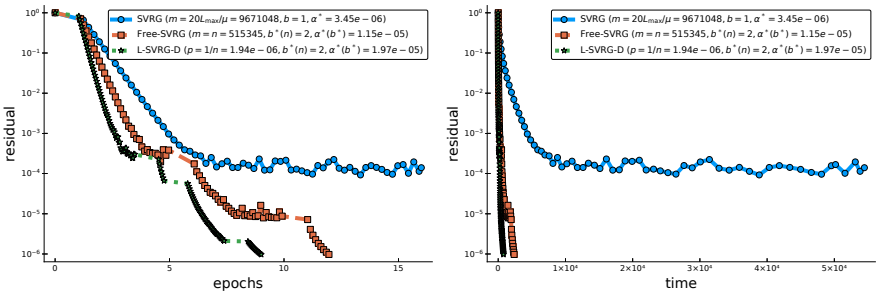
**Figure 3.7:** *Comparison of theoretical variants of SVRG without mini-batching ($b = 1$) on the real-sim data set.*

### 3.8.1.2 Experiment 1.b: optimal mini-batching

Here we use the optimal mini-batch sizes we derived for *Free-SVRG* in Table 3.1 and *L-SVRG-D* in (3.67). Since the original SVRG theory has no analysis for mini-batching, and the current existing theory shows that its total complexity increases with $b$, we use $b = 1$ for SVRG. Like in Section 3.8.1.1, the inner loop length is set to $m = n$. We confirm in these experiments that setting the mini-batch size to our predicted optimal value $b^*$ doesn't hurt our algorithms' performance. See Figures 3.8, 3.9, 3.10 and 3.11. Note that in Section 3.8.2.2, we further confirm that $b^*$ outperforms multiple other choices of the mini-batch size. In most cases, *Free-SVRG* and *L-SVRG-D* outperform the vanilla SVRG algorithm both on the epoch and time plots, except for the regularized logistic regression on the *real-sim* data set (see Figure 3.11), which is a very easy problem since it is well conditioned. Comparing Figures 3.5 and 3.9 clearly underlines the speed improvement due to optimal mini-batching, both in epoch and time plots.

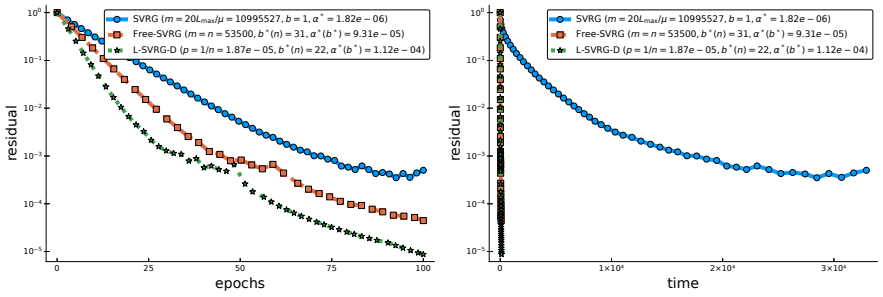



**(a)** $\lambda = 10^{-1}$

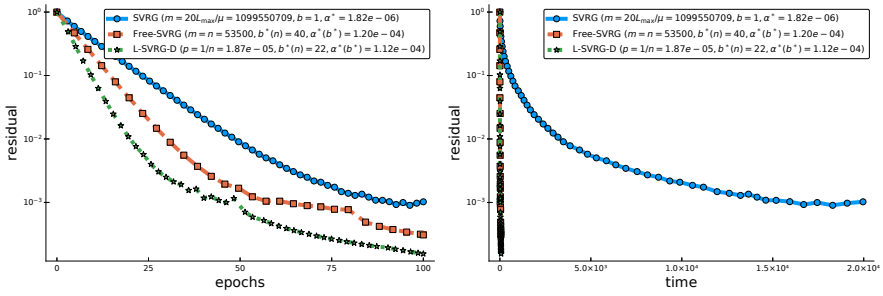



**(b)** $\lambda = 10^{-3}$

**Figure 3.8:** *Comparison of theoretical variants of SVRG with optimal mini-batch size $b^*$ when theoretically available on the YearPredictionMSD data set.*

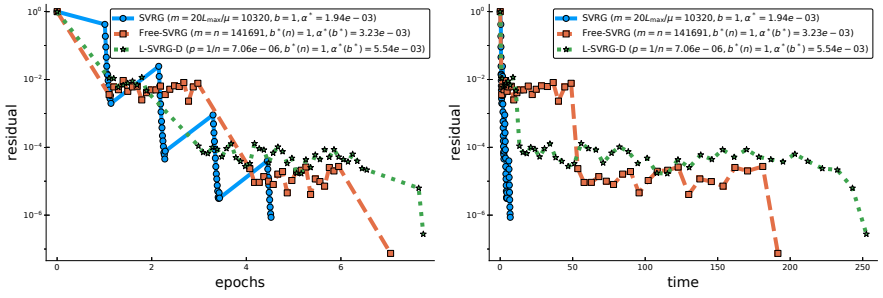



**(a)** $\lambda = 10^{-1}$

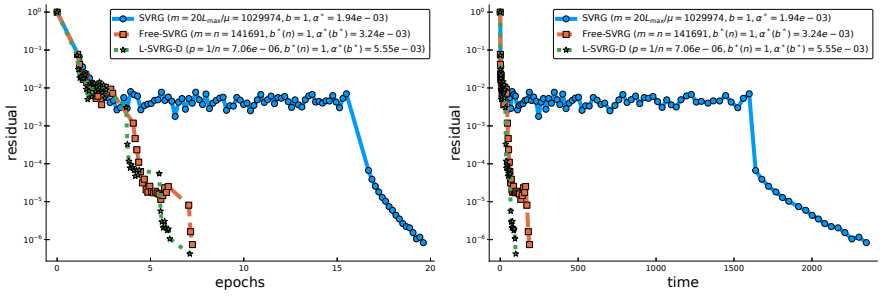



**(b)** $\lambda = 10^{-3}$

**Figure 3.9:** *Comparison of theoretical variants of SVRG with optimal mini-batch size $b^*$ when theoretically available on the slice data set.*

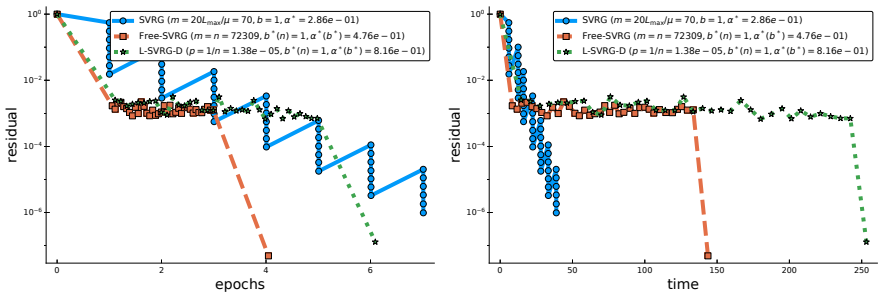



**(a)** $\lambda = 10^{-1}$

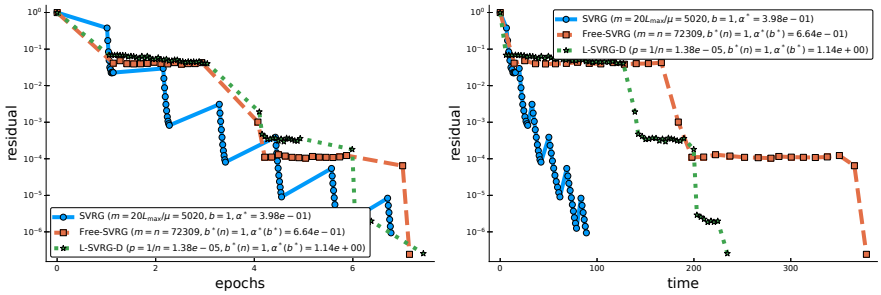



**(b)** $\lambda = 10^{-3}$

**Figure 3.10:** *Comparison of theoretical variants of SVRG with optimal mini-batch size $b^*$ when theoretically available on the ijcnn1 data set.*




**(a)** $\lambda = 10^{-1}$

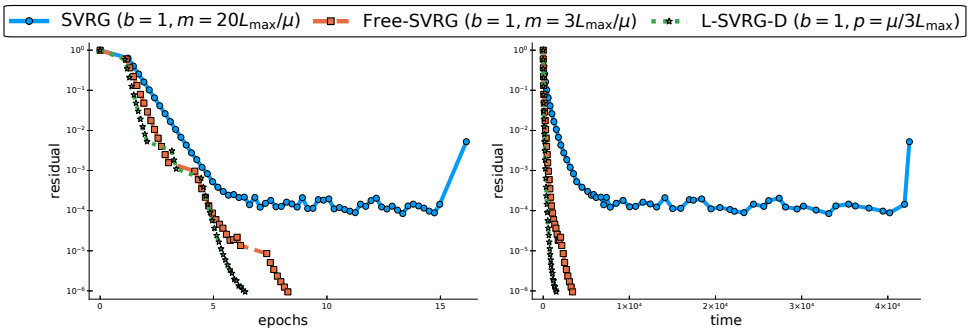



**(b)** $\lambda = 10^{-3}$

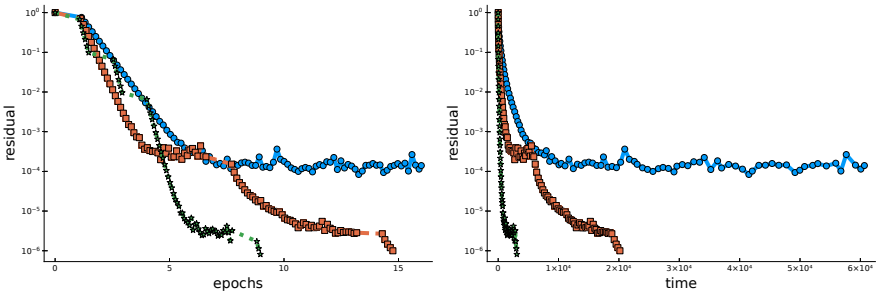
**Figure 3.11:** *Comparison of theoretical variants of SVRG with optimal mini-batch size $b^*$ when theoretically available on the real-sim data set.*

#### 3.8.1.3 Experiment 1.c: theoretical inner loop size or update probability without mini-batching

Here, using $b = 1$, we set the inner loop size for *Free-SVRG* to its optimal value $m^* = 3L_{\max}/\mu$ that we derived in Proposition 3.36. We set $p = 1/m^*$ for *L-SVRG-D*. The inner loop length is set like in Section 3.8.1.1. See Figures 3.12, 3.13, 3.14 and 3.15. By setting the size of the inner loop to its optimal value $m^*$, the results are similar to the one in experiments 1.a and 1.b. Yet, when comparing Figure 3.5 and Figure 3.13, we observe that it leads to a clear speed up of *Free-SVRG* and *L-SVRG-D*.

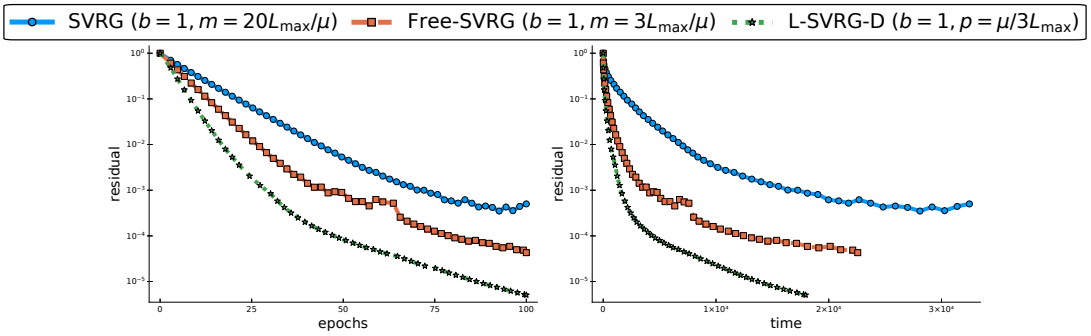



**(a)** $\lambda = 10^{-1}$

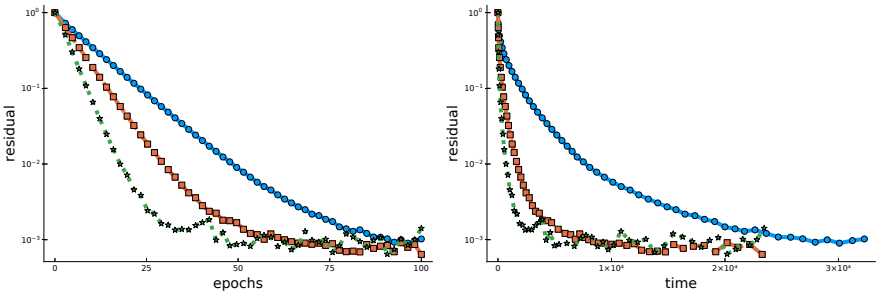



**(b)** $\lambda = 10^{-3}$

**Figure 3.12:** *Comparison of theoretical variants of SVRG with optimal inner loop size $m^*$ when theoretically available ($b = 1$) on the YearPredictionMSD data set.*

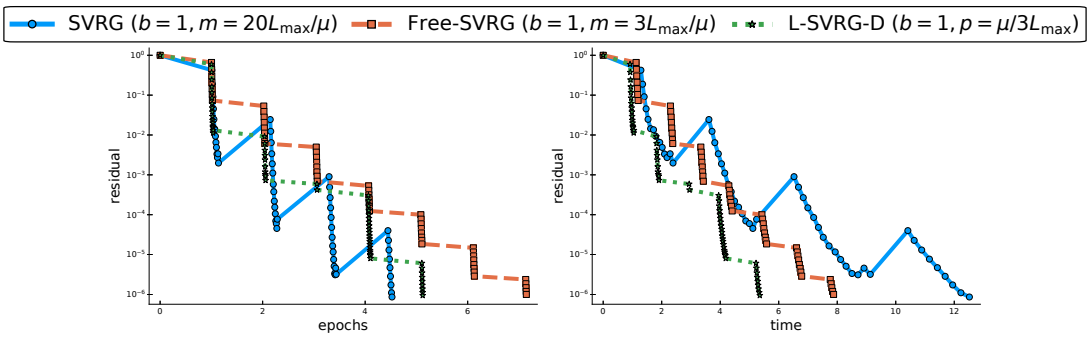



**(a)** $\lambda = 10^{-1}$

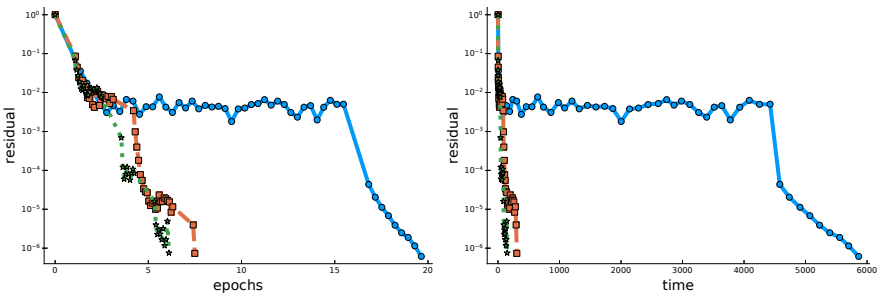



**(b)** $\lambda = 10^{-3}$

**Figure 3.13:** *Comparison of theoretical variants of SVRG with optimal inner loop size $m^*$ when theoretically available ($b = 1$) on the slice data set.*

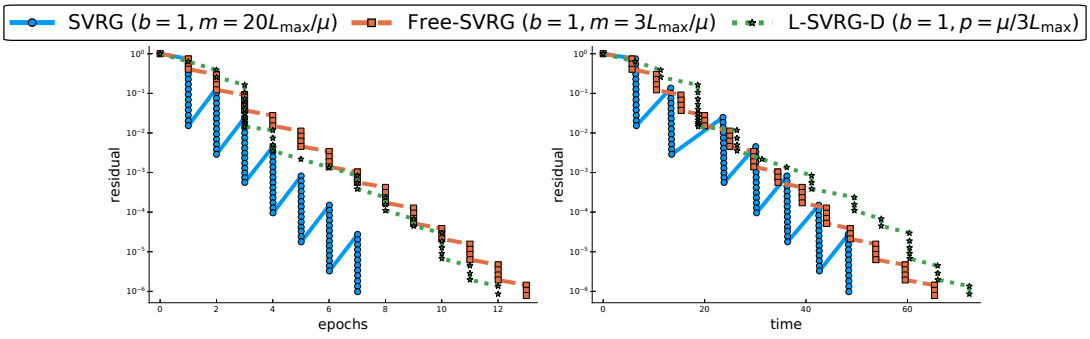



**(a)** $\lambda = 10^{-1}$

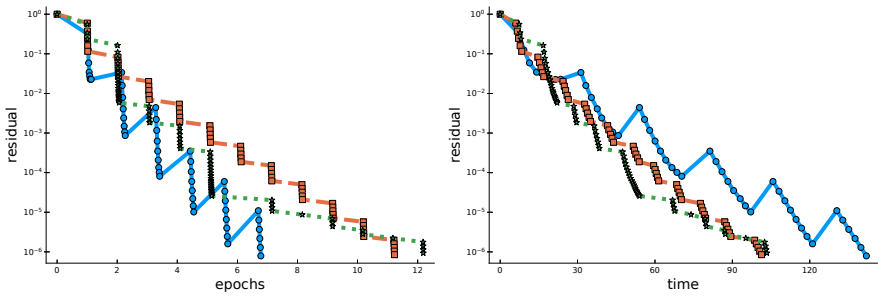



**(b)** $\lambda = 10^{-3}$

**Figure 3.14:** *Comparison of theoretical variants of SVRG with optimal inner loop size $m^*$ when theoretically available ($b = 1$) on the ijcnn1 data set.*




**(a)** $\lambda = 10^{-1}$




**(b)** $\lambda = 10^{-3}$

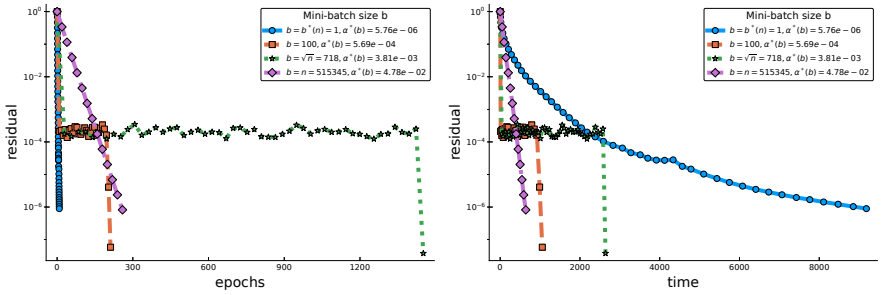
**Figure 3.15:** *Comparison of theoretical variants of SVRG with optimal inner loop size $m^*$ when theoretically available ($b = 1$) on the real-sim data set.*

### 3.8.2 Optimality of our theoretical parameters

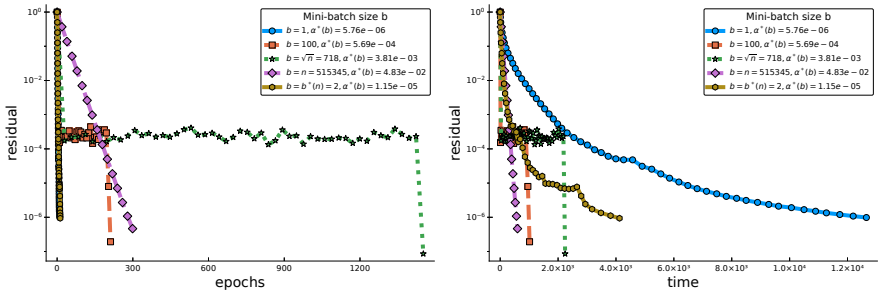
In this series of experiments, we only consider *Free-SVRG* for which we evaluate the efficiency of our theoretical optimal parameters, namely the mini-batch size $b^*$ and the inner loop length $m^*$.

#### 3.8.2.1 Experiment 2.a: comparing different choices for the mini-batch size

Here we consider *Free-SVRG* and compare its performance for different batch sizes: the optimal one $b^*$, 1, 100, $\sqrt{n}$ and $n$. In Figure 3.16, 3.17, 3.18 and 3.19, we show that the optimal mini-batch size we predict using Table 3.1 always leads to the fastest convergence in epoch plot (or at least near the fastest in Figure 3.16b).

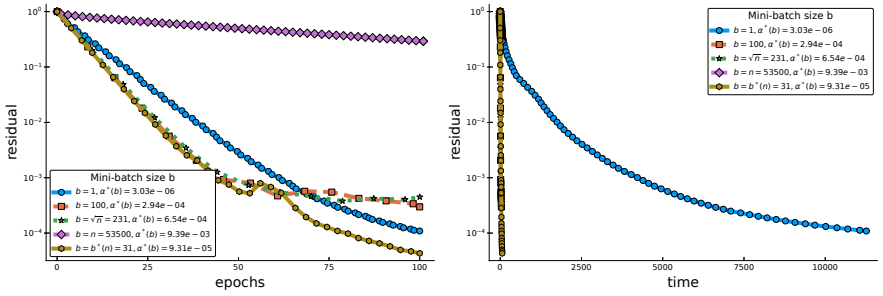



**(a)** $\lambda = 10^{-1}$

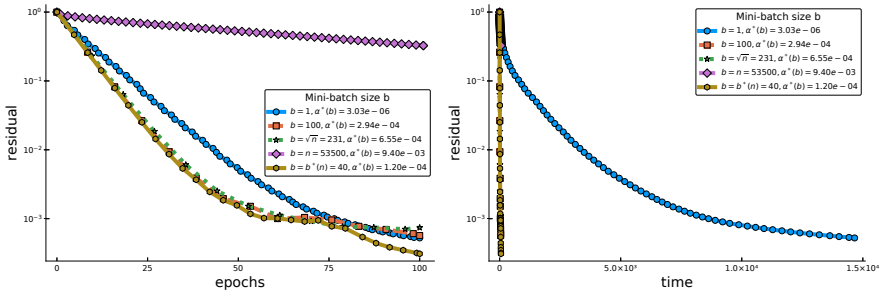



**(b)** $\lambda = 10^{-3}$

**Figure 3.16:** *Optimality of our mini-batch size $b^*$ given in Table 3.1 for Free-SVRG on the YearPredictionMSD data set.*

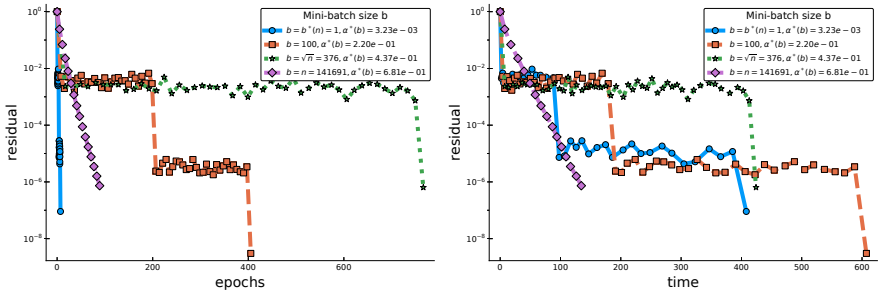



**(a)** $\lambda = 10^{-1}$

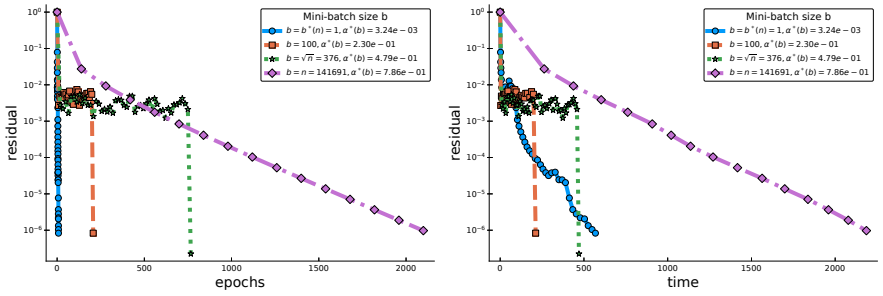



**(b)** $\lambda = 10^{-3}$

**Figure 3.17:** *Optimality of our mini-batch size $b^*$ given in Table 3.1 for Free-SVRG on the slice data set.*

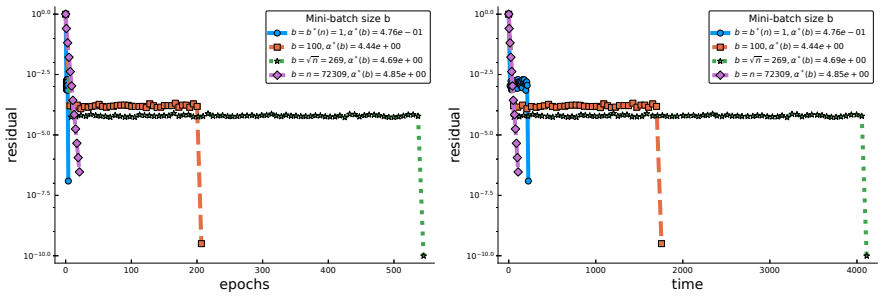



**(a)** $\lambda = 10^{-1}$

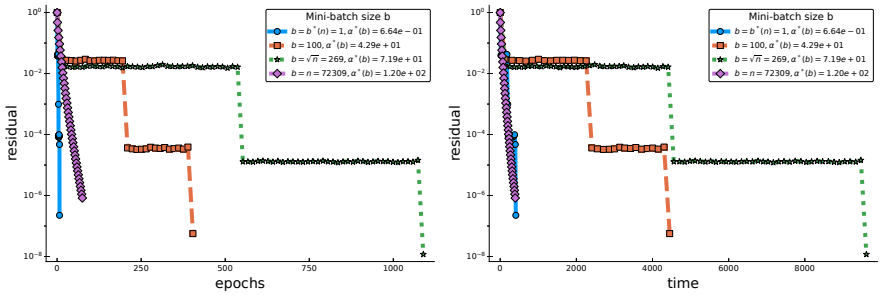



**(b)** $\lambda = 10^{-3}$

**Figure 3.18:** *Optimality of our mini-batch size $b^*$ given in Table 3.1 for Free-SVRG on the ijcnn1 data set.*




**(a)** $\lambda = 10^{-1}$

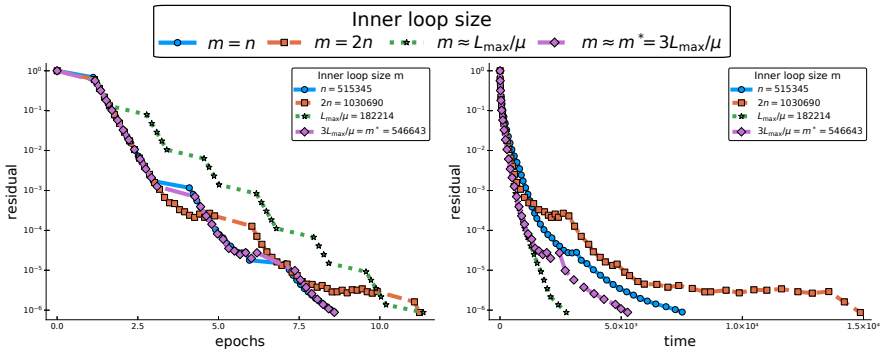



**(b)** $\lambda = 10^{-3}$

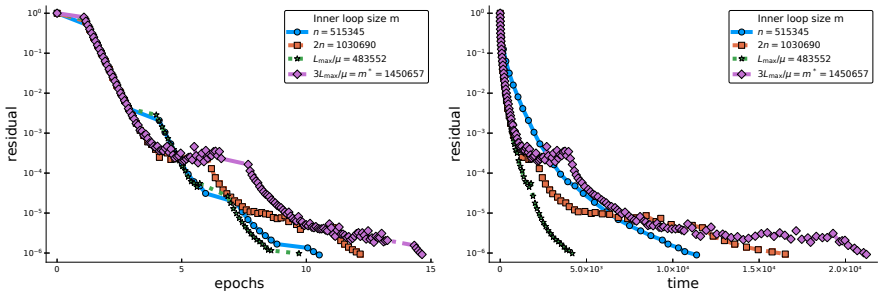
**Figure 3.19:** *Optimality of our mini-batch size $b^*$ given in Table 3.1 for Free-SVRG on the real-sim data set.*

### 3.8.2.2 Experiment 2.b: comparing different choices for the inner loop size

We set $b = 1$ and compare different values for the inner loop size: the optimal one $n$, $2n$, $L_{\max}/\mu$ and $m^* = 3L_{\max}/\mu$ in order to validate our theory in Proposition 3.36, that is, that the overall performance of *Free-SVRG* is not sensitive to the range of values of $m$, so long as $m$ is close to $n$, $L_{\max}/\mu$ or anything in between. And indeed, this is what we confirmed in Figures 3.20, 3.21, 3.22 and 3.23. The choice $m = 2n$ is the one suggested by [71] in their practical SVRG (Option II). We notice that our optimal inner loop size $m^*$ underperforms compared to $n$ or $2n$ only in Figure 3.23a, which is a very rare kind of problem since it is very well conditioned ($L_{\max}/\mu \approx 4$).

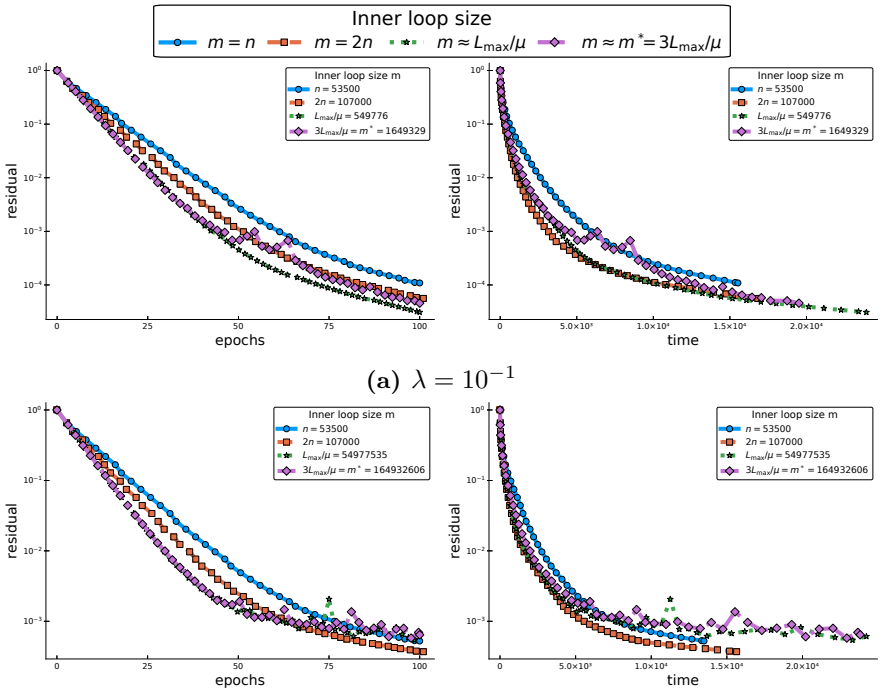



**(a)** $\lambda = 10^{-1}$




**(b)** $\lambda = 10^{-3}$

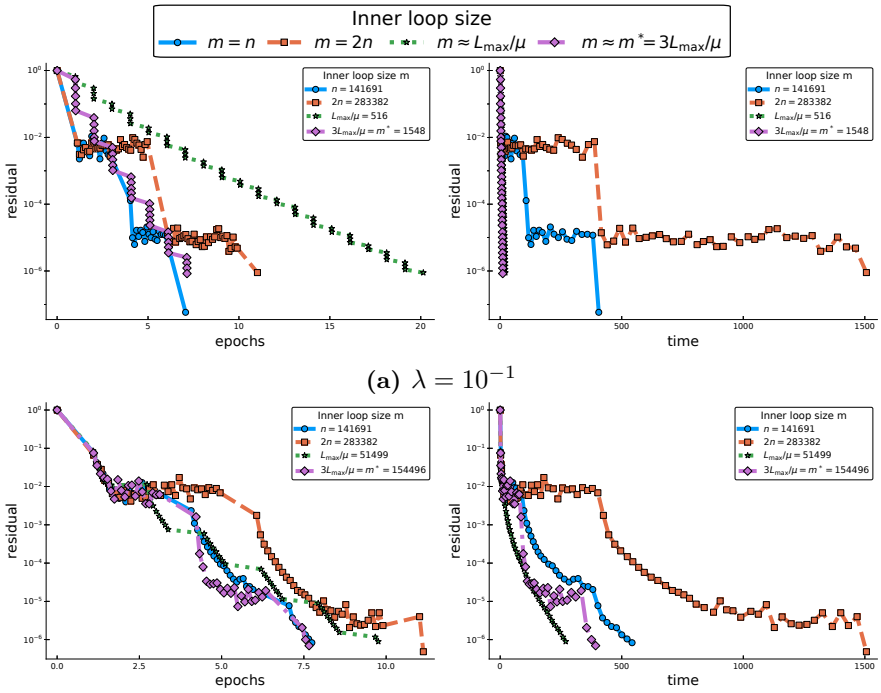
**Figure 3.20:** *Optimality of our inner loop size $m^* = 3L_{\max}/\mu$ for Free-SVRG on the YearPredictionMSD data set.*




**(a)** $\lambda = 10^{-1}$

**(b)** $\lambda = 10^{-3}$

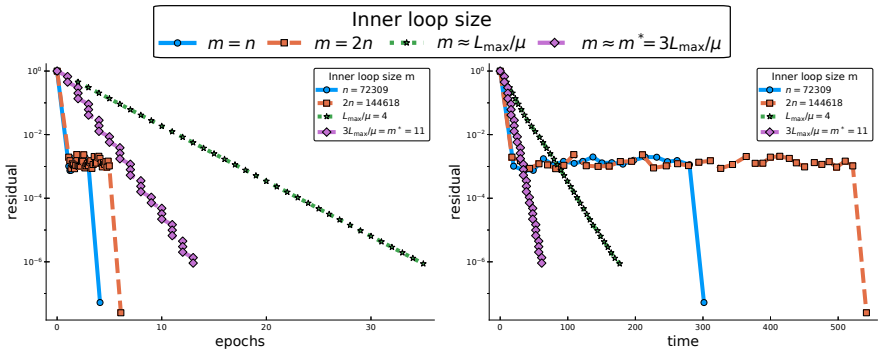
**Figure 3.21:** *Optimality of our inner loop size* $m^* = 3L_{\max}/\mu$ *for Free-SVRG on the slice data set.*



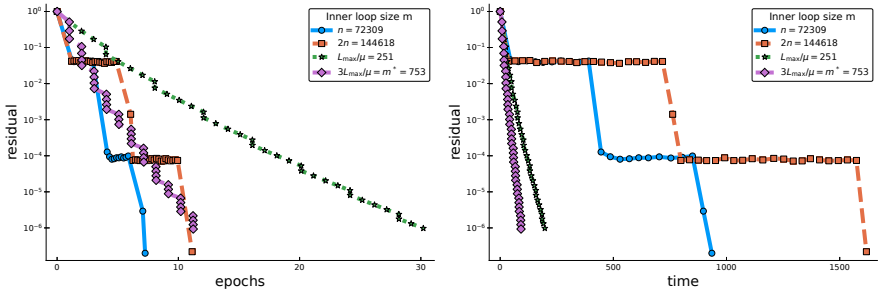

**(a)** $\lambda = 10^{-1}$

**(b)** $\lambda = 10^{-3}$

**Figure 3.22:** *Optimality of our inner loop size* $m^* = 3L_{\max}/\mu$ *for Free-SVRG on the ijcnn1 data set.*




**(a)** $\lambda = 10^{-1}$




**(b)** $\lambda = 10^{-3}$

**Figure 3.23:** *Optimality of our inner loop size $m^* = 3L_{\max}/\mu$ for Free-SVRG on the real-sim data set.*

# Part II

# Extension of sketch-and-project methods for ridge regression and ADI model problem

CHAPTER 4

# `RidgeSketch`: a fast sketching based solver for large scale ridge regression

كلموني ثاني عنك ... فكروني ... فكروني

أم كلثوم – فكروني

We propose new variants of the sketch-and-project method [59] for solving large scale ridge regression problems. Firstly, we propose a new momentum alternative and provide a theorem showing it can speed up the convergence of sketch-and-project, through a fast *sublinear* convergence rate. We carefully delimit under what settings this new sublinear rate is faster than the previously known linear rate of convergence of sketch-and-project without momentum. Secondly, we consider combining the sketch-and-project method with new modern sketching methods such as the Count sketch [29], SubCount sketch (a new method we propose), and subsampled Hadamard transforms [163, 147]. We show experimentally that when combined with the sketch-and-project method, the (sub)count sketch is very effective on sparse data and the standard subsample sketch is effective on dense data. Indeed, we show that these sketching methods, combined with our new momentum scheme, result in methods that are competitive even when compared to the Conjugate Gradient method [64] on real large scale data. On the contrary, we show the subsampled Hadamard transform does not perform well in this setting, despite the use of fast Hadamard transforms [21, 147, 70], and nor do recently proposed acceleration schemes work well in practice. To support all of our experimental findings, and invite the community to validate and extend our results, with this chapter we are also releasing an open source software package: `RidgeSketch`[1]. We designed this object-oriented package in Python for testing sketch-and-project methods and benchmarking ridge regression solvers. `RidgeSketch` is highly modular, and new sketching methods can easily be added as subclasses. We provide code snippets of our package in the appendix.

## 4.1 Introduction

Consider the regression problem given by

$$\min_{w\in\mathbb{R}^d} \frac{1}{2}\|\mathbf{X}w - y\|_2^2 + \frac{\lambda}{2}\|w\|_2^2 \ , \tag{4.1}$$

where $\mathbf{X} \in \mathbb{R}^{n\times d}$ is the feature matrix, $y \in \mathbb{R}^n$ the targets and $\lambda > 0$ the regularization parameter.

The need to solve linear regression such as (4.1) occurs throughout scientific computing, where it is known as ridge regression in the statistics community [132, 155], data assimilation in weather forecasting [82]

[1]Our fully documented and tested package is available at `https://github.com/facebookresearch/RidgeSketch`.

and linear least squares with Tikhonov regularization among numerical analysts [47].

Here we focus on applications where the number of rows and columns of $\mathbf{X} \in \mathbb{R}^{n\times d}$ are both large scale. With datasets now being collected automatically and electronically, there has been a surge of new stochastic incremental methods that can gracefully scale with the dimensions of the data in (4.1). Yet, it is still unclear if the new stochastic methods are capable of outperforming the classic Conjugate Gradient (CG) methods [64], even though the CG method was developed in the 1950's. We address this question, by designing and implementing new stochastic methods based on randomized sketching [116, 117] together with the iterative projection methods known as the sketch-and-project methods [59]. Our main objective can be stated simply as

> **Main objective:** Develop variants of the sketch-and-project method that are competitive in practice, and provide new theoretical support for these variants.

We do this by exploring the use of new sketching transforms such as `Count sketch` [27] and proposing a new momentum variant of the sketch-and-project method. To demonstrate the benefits of our new momentum and sketching variant, we show how our resulting method is competitive with the Conjugate Gradients (CG) methods, and supersedes previous momentum based variants [91]. We will also give several negative results, such as showing how, in our setting, recently developed accelerated variants [150, 50] are not viable in practice, and nor are the fast Johnson-Lindenstrauss (JL) transforms [1].

In the following Section 4.2 we outline some of the background and state our main contributions. In Section 4.3, we describe the sketch-and-project method for solving ridge regression and the different sketching methods we explored in Section 4.4. Then, we give a convergence theory for our method without and with momentum respectively in Sections 4.5 and 4.6, that we specialize in Section 4.7 for single column sketches. To illustrate this theoretical results, we provide large scale numerical experiments highlighting our theoretical findings in Section 4.8 and showing that our momentum sketch-and-project method competes against CG and direct solvers. Finally, in Section 4.9 we present the accelerated version of our method and show its inefficiency in practice since it requires additional overhead costs and spectral information of $\mathbf{A}$ to compute acceleration parameters.

## 4.2 Background and contributions

**Iterative sketching in linear systems.** When the dimensions $n$ and $d$ are large, direct methods for solving (4.1) can be infeasible, and iterative methods are favored. In particular, the Krylov methods including the CG algorithm [64] are the industrial standard so long as one can afford full matrix vector products and the system matrix fits in memory. On the other hand, if a single matrix vector product is considerably expensive, or the data matrix is too large to fit in memory, then iterative methods exploiting only few rows or columns of $\mathbf{A}$ are effective. This includes, for example, randomized Kaczmarz methods [73, 144, 101], its greedy or deterministic variants [115, 38, 31, 62, 11], and Coordinate Descent (CD) method [87, 93, 164]. In [59], the authors united Kaczmarz, CD and host of other randomized iterative methods under the sketch-and-project framework.

*Contributions.* We revisit the sketch-and-project method and examine how to make specialized variants for solving ridge regression that are competitive as compared to conjugate gradients. To do so, we are also releasing a high-quality and modular Python package called `RidgeSketch` for efficiently implementing and testing sketch-and-project methods. More information about the code and how to contribute by adding sketches or datasets are detailed in Section 4.10.

**Momentum.** Recently, a variant of sketch-and-project with momentum was proposed in [91]. The authors of [91] show a theoretical advantage in terms of convergence in expectation (of the first moment only), but do not present any advantage in terms of convergence in $\ell^2$ or high probability. There has even more recent work [97] analysing momentum together with sketch-and-project, and extensions to solving the linear feasibility problem [98, 99]. Yet none of these works show any theoretical benefit of using

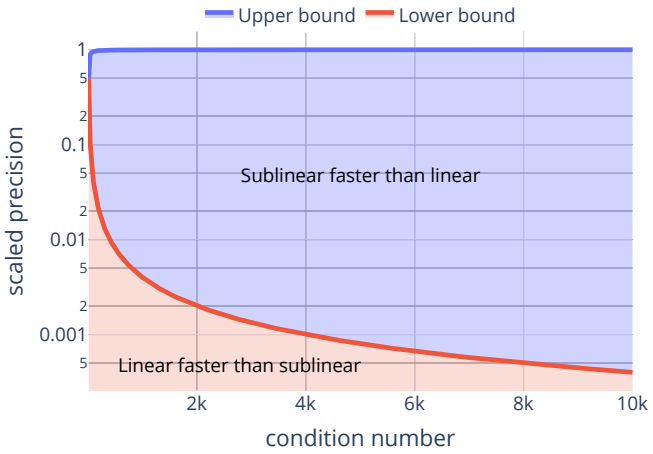



**Figure 4.1:** *Areas of superiority of sublinear convergence of CD with momentum (in blue) and linear convergence of CD without momentum (in red), where the upper and lower bounds are respectively the left and right-hand side of* (4.64)*. Condition number is denoted* $\kappa = \frac{\lambda_{\max}(\mathbf{A})}{\lambda_{\min}(\mathbf{A})}$ *and the scaled precision is defined as* $\hat{\epsilon} = \frac{\epsilon}{\lambda_{\max}(\mathbf{A})}$

momentum over using no momentum at all. This lack of a theoretical benefit of using momentum is echoed throughout stochastic optimization: there are almost no theoretical benefits in using momentum outside of the strongly convex full batch setting [118].

*Contributions.* Here we have establish the first theoretical advantage of using momentum together with sketch-and-project method. We show that when using sketch-and-project with momentum, the last iterate enjoys a fast sublinear convergence. Without momentum, this result does not hold. Instead, without momentum, it has only been shown to hold for the average of the iterates. As depicted in Figure 4.1, we show that the fast sublinear rate of our new momentum variant gives a tighter complexity bound than the previously best known linear rate of convergence [54] when the desired precision is moderate and the condition number of the underlining problem is moderate to large. Our new convergence theory for momentum also suggests completely iteration dependent schedules for setting the momentum parameter. We perform extensive numerical tests showing the superiority of this new scheduling as compared to using a constant momentum.

**Acceleration.** Lee and Sidford [86] show how the Kaczmarz method could be accelerated through its connection to CD, and also compare the resulting accelerated convergence rate to the rate of the Conjugate Gradient algorithm. Though, they provide no numerical experiments leaving it unclear if this form of acceleration can afford any practical advantage. Later, Liu and Wright [90] developed an accelerated Kaczmarz method and show that it can be faster than CG on densely generated artificial data. But they do not provide examples of this on real data or an affordable rule for setting the acceleration parameters. More recently, it was shown that the entire family of the sketch-and-project methods could be accelerated [50]. Yet, experiments of the authors rely on a grid search that would defeat any gains in using acceleration.

*Contributions.* We investigate the possibility of developing a practical setting for the two acceleration parameters proposed in [50] that would result in a robust performance gain over standard sketch-and-project. Unlike Nesterov's acceleration for gradient descent in the convex setting, there are no default parameter settings that work consistently across a significant class of problems. We show through a careful grid search that finding good parameters is like "looking for a needle in the haystack" and that identifying any practical settings for these parameters is virtually impossible. We thus recommend that,

to advance the use of acceleration for linear systems, one would need to study a smaller class of problems and certain spectral bounds to derive a working rule for setting the parameters.

**Johnson-Lindenstrauss sketches.** In [72], the authors show how high dimension data can be projected onto a low dimension subspace using a Gaussian matrix in such a way that it approximately preserves the pairwise distance between points. This result is now known as the celebrated Johnson-Lindenstrauss (JL) Lemma. Since then, many more random transforms have been shown to satisfy this property, such as the Count sketch [29], subsampled Fourier [1] and Hadamard transforms [21, 147, 70]. These JL *transforms* have been used to speed up LAPACK solvers [6], solving linear regression [160] and Newton based methods [117].

*Contributions.* We propose new combinations of the sketch-and-project method with JL sketches such as the Subsampled Randomized Hadamard Transform (SRHT) and a new subsampled count (SubCount) sketch. Our new package `RidgeSketch` is also setup in a way that is easily extensible, where new sketching methods can easily be added as a new instance of a *sketching class* (see Section 4.10). Our results for the subsampled Hadamard sketch are negative. We show, despite the favourable theoretical complexity of using Hadamard sketches, that the overhead costs make them a completely impractical choice for sketch-and-project methods. The SubCount sketch, on the other hand, when combined with sketch-and-project, results in an efficient method.

**Alternative iterative sketching based methods.** A closely related method to the sketch-and-project method is the Iterative Hessian Sketch [117], which makes use of iterative sketching to solve constrained quadratics. In the unconstrained setting such as (4.1), Iterative Hessian Sketch is efficient when the dimension $d$ to be significantly smaller than number of data points $n$. This rules out one of our main applications: kernel ridge regression where $n = d$.

### 4.2.1 Linear system formulation

Since the optimization problem (4.1) is differentiable and without constraints, its solution satisfies the stationarity conditions given by

$$\left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right) w = \mathbf{X}^\top y \ . \tag{4.2}$$

We can also rewrite the above linear system in its *dual form* given by

$$w = \mathbf{X}^\top \alpha \ , \quad \text{where} \quad \left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}\right) \alpha = y \ . \tag{4.3}$$

The equivalence between solving (4.2) and (4.3) is well known and proven in the following lemma for completeness.

**Lemma 4.1.** *The linear systems* (4.2) *and* (4.3) *have the same solution.*

*Proof.* Note that

$$\left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right) \mathbf{X}^\top = \mathbf{X}^\top \left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}\right) \ .$$

Left multiplying the above by the inverse of $\left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)$ and right multiplying by the inverse of $(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I})$ gives

$$\mathbf{X}^\top \left(\mathbf{X}\mathbf{X}^\top + \lambda \mathbf{I}\right)^{-1} = \left(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}\right)^{-1} \mathbf{X}^\top \ .$$

Finally right multiplying by $y$, on the left we have the solution to (4.3) and on the right the solution to (4.2). □

The choice of solving (4.2) or (4.3) will depend on the dimensions of the data. On the one hand, the system in (4.2) involves a $d \times d$ matrix, and thus the *primal form* (4.2) is preferred when $d \leq n$. On the other hand, the system in (4.3) involves a $n \times n$ matrix, and thus solving the *dual form* (4.3) is preferred when $n < d$.

In either case, the bottleneck cost is the solution of a linear system where the system matrix is symmetric and positive definite. To simplify notation we introduce $\mathbf{A} \stackrel{\text{def}}{=} \mathbf{B} + \lambda \mathbf{I}$, and let

$$\mathbf{A}w = b \ , \tag{4.4}$$

where $\mathbf{B} = \mathbf{X}^\top \mathbf{X}$ and $b = \mathbf{X}^\top y$, for the primal form (4.2), or $\mathbf{B} = \mathbf{X}\mathbf{X}^\top$ and $b = y$ for the dual one (4.3). Note that in the later case, a multiplication by $\mathbf{X}^\top$ is required to recover the weights vector. Let $m$ be the dimensions of $\mathbf{A} \in \mathbb{R}^{m \times m}$, thus $m$ equals the smallest dimension of the design matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. Indeed, $m = d$ if we choose to solve the primal version (4.2) and $m = n$ if we choose to solve the dual one (4.3).

Thus solving the ridge regression problem boils down to finding the solution of the linear system (4.4). If the dimension of the squared matrix $\mathbf{A}$, denoted $m$, is not too large, one can solve this problem using a direct solver (for instance through a SVD or a Cholesky decomposition). But when $m$ is large, direct methods become intractable as their computational cost grows with $\mathcal{O}(m^3)$.

### 4.2.2 Using a kernel

We also consider kernel ridge regression, where the feature matrix is the result of applying a feature map. This leads to particular considerations since the resulting feature matrix may have an infinite number of columns.

The idea behind kernel ridge regression is that, instead of learning using the original input (or feature) vectors $\mathbf{X} \stackrel{\text{def}}{=} [x_1, \ldots, x_n]$, we can learn using a high dimensional feature map of the inputs $\phi : \mathbb{R}^d \to \mathbb{R}^r$ where $r > d$ or even an infinite dimensional space. For instance, $\phi(x)$ could encode a high dimensional polynomial. By replacing each $x_i$ with $\phi(x_i)$ in (4.1) we arrive at

$$\min_{w \in \mathbb{R}^r} \frac{1}{2} \sum_{i=1}^{n} (\phi(x_i)w - y_i)^2 + \frac{\lambda}{2} \|w\|_2^2 \ . \tag{4.5}$$

When $r$ is large, or even infinite, solving (4.5) directly can be difficult or intractable. Fortunately, the *dual formulation* of (4.5) is always an $n$–dimensional problem independently of the dimension $r$ [2]. The *dual formulation* of (4.5) is given by

$$\alpha^* = \arg\min_{\alpha \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{K}\alpha - y\|_2^2 + \frac{\lambda}{2} \alpha^\top \mathbf{K} \alpha \ , \tag{4.6}$$

where $\mathbf{K} = (\mathbf{K}(x_i, x_j))_{ij} \stackrel{\text{def}}{=} \langle \phi(x_i), \phi(x_j) \rangle_{ij}$ is the kernel matrix. This is equivalent to solving in $\alpha$ the linear system

$$\mathbf{K} (\mathbf{K} + \lambda \mathbf{I}) \alpha = \mathbf{K} y \ . \tag{4.7}$$

With $\alpha^*$, the solution to the above, we can then predict the output of a new input vector $x$ using

$$\text{predict}(x) \ = \ \sum_{i=1}^{n} \alpha_i^* \langle \phi(x_i), \phi(x) \rangle \ = \ \sum_{i=1}^{n} \alpha_i^* \mathbf{K}(x_i, x) \ = \ \mathbf{K}\alpha^* \ .$$

Consequently to solve (4.6) and make predictions, we only need access to the kernel matrix. Fortunately, there are several feature maps for which the kernel matrix is easily computable including the one we use in our experiments which is the Gaussian Kernel, otherwise known as the Radial Basis Function

$$K(x, x') = \exp\left( \frac{-||x - x'||^2}{2\sigma^2} \right) \ . \tag{4.8}$$

where $\sigma > 0$ is the *kernel* parameter.

[2] This is commonly known as the *kernel trick*, see Chapter 16 in [136].

**Algorithm 11** Sketch-and-project method

**Parameters:** distribution over random $m \times \tau$ matrices in $\mathcal{D}$, tolerance $\epsilon > 0$
Set $w^0 = 0 \in \mathbb{R}^m$ ▷ weights initialization
Set $r^0 = \mathbf{A}w^0 - b = -b \in \mathbb{R}^m$ ▷ residual initialization
$k = 0$
**While** $\left\|r^k\right\|_2 / \left\|r^0\right\|_2 \leq \epsilon$ **do**
    Sample an independent copy $\mathbf{S}_k \sim \mathcal{D}$
    $r^k_{\mathbf{S}} = \mathbf{S}_k^\top r^k$ ▷ compute sketched residual
    $\delta_k = \texttt{least\_norm\_solution}\left(\mathbf{S}_k^\top \mathbf{A}\mathbf{S}_k, r^k_{\mathbf{S}}\right)$ ▷ solve sketched system
    $w^{k+1} = w^k - \mathbf{S}_k\delta_k$ ▷ update the iterates
    $r^{k+1} = r^k - \mathbf{A}\mathbf{S}_k\delta_k$ ▷ update the residual
    $k = k + 1$
**Output:** $w^t$ ▷ return weights vector

Ultimately, despite the addition of a kernel, the resulting problem (4.7) is still a linear system of the form $\mathbf{A}w = b$ where

$$\mathbf{A} = \mathbf{K}^\top(\mathbf{K} + \lambda\mathbf{I}) \quad \text{and} \quad b = \mathbf{K}^\top y \ .$$

The only marked difference now is that $\mathbf{A}$ tends to be dense, and because of this, matrix-vector products are particularly expensive.

## 4.3 The sketch-and-project method

Sketch-and-project is an archetypal algorithm that unifies a variety of randomized iterative methods including both randomized Kaczmarz and CD [59], and all their block and importance sampling variants.

At each iteration, the sketch-and-project methods randomly compressed the linear system using what is known as a *sketching matrix*.

**Definition 4.2.** *Let $\tau \in \mathbb{N}$ and let $\mathcal{D}$ be a distribution over matrices in $\mathbb{R}^{m\times\tau}$. We refer to $\tau$ as the sketch size and to $\mathbf{S} \in \mathbb{R}^{m\times\tau}$ drawn from the distribution $\mathcal{D}$ as a sketching matrix.*

We can use a sketching matrix $\mathbf{S}$ to reduce the number of rows of the linear system (4.4) to $\tau$ rows as follows

$$\mathbf{S}^\top\mathbf{A}w = \mathbf{S}^\top b \ . \tag{4.9}$$

If the sketch size $\tau$ is sufficiently large and the sketching matrix is appropriately chosen, then we can guarantee with high probability that the solution to the *sketched* linear system (4.9) is close to the solution $w^*$ of the original system (4.4) (see [94]). But this *one-shot* sketching approach poses several challenges 1) it may be hard to determine how large $\tau$ should be, 2) the sketched linear system now has multiple solutions and 3) with some low probability the solution to (4.9) could be far from $w^*$. To address these issues, we use an iterative projection scheme.

Let $\mathbf{W}$ be a symmetric positive definite matrix of order $n$ (which will typically be chosen as $\mathbf{A}$), here we project with respect to the $\mathbf{W}$–norm[3] given by $\|\cdot\|_{\mathbf{W}} = \sqrt{\langle\cdot, \mathbf{W}\cdot\rangle}$.

At the $k^{\text{th}}$ iteration of the sketch-and-project algorithm, a sketching matrix $\mathbf{S}_k$ is drawn from $\mathcal{D}$ and the current iterate $w^k$ is projected onto the solution space of the sketched system $\mathbf{S}_k^\top\mathbf{A}x = \mathbf{S}_k^\top b$ with respect to the $\mathbf{W}$–norm, that is

$$w^{k+1} = \underset{w\in\mathbb{R}^m}{\arg\min} \left\|w - w^k\right\|^2_{\mathbf{W}} \quad \text{subject to} \quad \mathbf{S}_k^\top\mathbf{A}w = \mathbf{S}_k^\top b \ . \tag{4.10}$$

[3]Using this norm for symmetric positive definite matrices has shown to result in algorithms with a fast convergence rate [59].

The closed form solution to (4.10) is given by

$$w^{k+1} = w^k - \mathbf{W}^{-1}\mathbf{A}\mathbf{S}_k \left(\mathbf{S}_k^\top \mathbf{A}\mathbf{W}^{-1}\mathbf{A}\mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top (\mathbf{A}w^k - b) \ , \tag{4.11}$$

where $^\dagger$ denotes the pseudoinverse. When $\mathbf{A}$ is known to be positive definite, as in our case, using $\mathbf{W} = \mathbf{A}$ often results in an overall faster convergence of (4.11) as shown in [59]. Using $\mathbf{W} = \mathbf{A}$ in (4.11) gives the updates

$$w^{k+1} = w^k - \mathbf{S}_k \left(\mathbf{S}_k^\top \mathbf{A}\mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top (\mathbf{A}w^k - b) \ . \tag{4.12}$$

We refer to (4.12) as the `RidgeSketch` method since it is specialized for solving ridge regression. Here we give the details on how to efficiently implement the `RidgeSketch` update (4.12), see Algorithm 11.

One practical detail we have added to the pseudocode in Algorithm 11 is a stopping criteria. For any iterative algorithm, it is important to know when to stop. We can do this by monitoring the *residual* $r^k \overset{\text{def}}{=} \mathbf{A}w^k - b$. From an initial residual $r^0 \in \mathbb{R}^m$, when the relative residual $\left\|r^k\right\| / \left\|r^0\right\|$ is below a given tolerance, we stop. We also need this residual for computing the update (4.12). We can efficiently update the residual from one iteration to the next since

$$r^{k+1} = \mathbf{A}w^{k+1} - b = \mathbf{A}(w^k - \mathbf{S}_k\delta_k) - b = r^k - \mathbf{A}\mathbf{S}_k\delta_k \ , \tag{4.13}$$

where $\delta_k \overset{\text{def}}{=} \left(\mathbf{S}_k^\top \mathbf{A}\mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top (\mathbf{A}w^k - b) = \left(\mathbf{S}_k^\top \mathbf{A}\mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top r^k$. Thus we can update the residual at the cost of $O(m\tau)$, that is, multiplying a $m \times \tau$ matrix $\mathbf{A}\mathbf{S}_k$ with the $\tau$ dimensional vector $\delta_k$. Note that $\delta_k$ is can be efficiently computed as the least-norm solution of the following linear system in $x$

$$\mathbf{S}_k^\top \mathbf{A}\mathbf{S}_k x = \mathbf{S}_k^\top r^k \ . \tag{4.14}$$

## 4.4 Sketching methods and matrices

Here we introduce several sketching matrices that can be used in Algorithm 11. The ideal sketch is one that reduces the dimension of the linear system (4.4) as much as possible, while preserving as much information as possible and that can be efficiently implemented. As we discuss throughout this section, there is no sketch that has all three of these qualities, and ultimately, one must make a trade-off between them.

### 4.4.1 Classical sketches

One of the most classical and simple sketching method is the *Gaussian sketch*.

**Definition 4.3.** *A* **Gaussian sketch** *is a random matrix $\mathbf{S} \in \mathbb{R}^{m\times\tau}$ where each element is sampled i.i.d for the standard Gaussian distribution.*

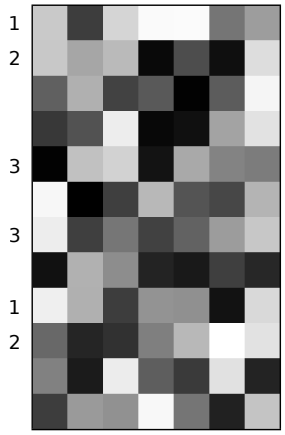
As argued in [116], the resulting sketched matrix $\mathbf{S}^\top\mathbf{A}$ can be a good approximation to the full matrix and is easy to control with probability bounds. Though simple to implement, the cost of forming $\mathbf{S}^\top\mathbf{A}$ is $\mathcal{O}(\tau m^2)$, which is expensive.

A much cheaper option is to use a *Subsampling sketch*.

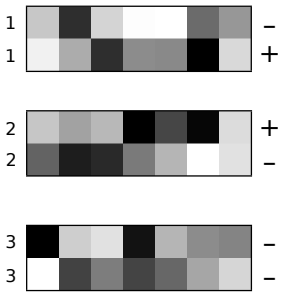
**Definition 4.4.** *A* **Subsampling sketch** *is based on a randomly sampled subset $C \subset \{1,\dots,m\}$ with $|C| = \tau$ elements drawn uniformly on average from all such subsets. Let $\mathbf{I}_C \in \mathbb{R}^{m\times\tau}$ denote the concatenation of the columns of the identity matrix $\mathbf{I} \in \mathbb{R}^{m\times m}$ whose columns are indexed in $C$. We define the subsampling sketch distribution as*

$$\mathbb{P}[\mathbf{S} = \mathbf{I}_C] \ = \ \frac{1}{\binom{m}{\tau}} \ , \quad \textit{for all} \quad C \subset \{1,\dots,m\},\ |C| = \tau \ . \tag{4.15}$$

Subsampling sketches are very cheap to compute, indeed, we need not even compute $\mathbf{S}^\top\mathbf{A}$ since is simply equivalent to fetching the rows of $\mathbf{A}$ indexed by a random subset. This can be done in Python without




**(a)** *Selection of rows with subsampling size* $s = 6$.




**(b)** *Random sign flip of selected rows that will be summed together with sum size* $k = 2$.

**Figure 4.2:** *SubCount sketch example.*

generating any copies of the data by slicing the selected rows. Slicing is very well optimized operation in NumPy [152] and SciPy [154] Compressed Sparse Row (CSR) sparse arrays, which makes it one the fastest sketching method.

Though subsampling sketches are cheap and fast, the sketched matrix $\mathbf{S}^\top\mathbf{A}$ can be a poor approximation of $\mathbf{A}$, since it is always possible that some vital part of $\mathbf{A}$ is "left out" in the rows that were not sampled. Still, the subsampling sketch will prove to work well within the iterative sketch-and-project scheme.

Next we consider a sketch that makes use of subsampling, addition and subtraction of rows.

### 4.4.2 Count and SubCount sketch

In order to avoid losing too much information by just subsampling rows, one can also sum and subtract groups of rows. This is the idea behind `Count sketch` which stems from the streaming data literature [27, 30] and got popularized as a matrix sketching tool by [28]. Count sketch selects rows of $\mathbf{A}$, flips their sign with probability $1/2$ and add it to a random row, sampled uniformly, of the output matrix $\mathbf{S}^\top\mathbf{A}$.

To decrease the overall cost of Count Sketch, we also combined it with a subsampling step. We call the resulting method the `SubCount sketch`. SubCount sketch has two parameters, the subsampling size $s \in \{1, \dots, m\}$ and the sum size $k \in \{1, \dots, s\}$ that must be such that it divides $s$. The SubCount sketch can be broken down into three steps: subsampling $s$ random rows of the input, then randomly flipping their sign and finally summing $k$ contiguous rows together. This can also be illustrated in terms of matrix multiplications as follows.

**Definition 4.5.** *A* **SubCount sketch** *is a matrix* $\mathbf{S} \in \mathbb{R}^{m\times\tau}$ *such that* $\mathbf{S}^\top = \mathbf{\Sigma}\mathbf{D}\mathbf{I}_C$*, where*

- $\mathbf{I}_C \in \mathbb{R}^{s\times m}$ *is a subsampling matrix based on a set* $C \subset \{1, \dots, m\}$ *chosen uniformly at random from all sets with* $s$ *elements*
- $\mathbf{D} \in \mathbb{R}^{s\times s}$ *is a diagonal matrix with elements sampled uniformly from* $\{-1, 1\}$
- $\mathbf{\Sigma} \in \mathbb{R}^{\frac{s}{k}\times s}$ *is a* sum *matrix, that sums every* $k$ *contiguous rows together, that is*

$$\mathbf{\Sigma} = \begin{bmatrix} \underbrace{1\cdots1}_{k} & 0\cdots0 & \cdots & 0\cdots0 \\ 0\cdots0 & \underbrace{1\cdots1}_{k} & \cdots & 0\cdots0 \\ 0\cdots0 & 0\cdots0 & \cdots & \underbrace{1\cdots1}_{k} \end{bmatrix} \tag{4.16}$$

See Figure 4.2 for a depiction of the first two steps of SubCount sketch. The resulting sketch size is given by $\tau = s/k \in \mathbb{N}^*$. In our `RidgeSketch` package, since the sketch size $\tau$ is a parameter selected by the user, we adjusted $s$ and $k$ such that $\tau = s/k$ as follows: If $10\,\tau \leq m$, we set arbitrarily $k$ to 10, else, we fix $k = \lfloor m/\tau \rfloor$ and then compute $s = k\tau$. When we have no subsampling, that is $s = m$, then we simply refer to the SubCount Sketch and Count Sketch[4]. The advantage of Count sketch is that it enables the computation of $\mathbf{S}^\top \mathbf{A}$ with a cost of $\mathcal{O}(nnz(\mathbf{A}))$, where $nnz(\mathbf{A})$ is the number of non-zero values of $\mathbf{A}$. Thus, this sketching method is fast for large sparse matrices, especially for the CSR (column sparse rows) format for which accessing rows is efficient. Moreover, because it linearly combines groups of rows of the input matrix, it avoids the pitfall of leaving out meaningful rows of $\mathbf{A}$, unlike subsampling. This will later be confirmed numerically in Section 4.8.

### 4.4.3 Subsampled randomized Hadamard transform

Here we consider the *Subsampled Randomized Hadamard Transform* (SRHT) [163, 147], which we refer to as `Hadamard sketch` for short.

**Definition 4.6.** *A* **Hadamard sketch or SRHT** *is a matrix $\mathbf{S} \in \mathbb{R}^{m\times\tau}$ such that*

$$\mathbf{S}^\top = \frac{1}{\sqrt{\tau m}} \mathbf{I}_C \mathbf{H}_m \mathbf{D} \ , \tag{4.17}$$

*where we assume there exists $q \in \mathbb{N}^*$ such that $m = 2^q$ with, and*

- $\mathbf{D} \in \mathbb{R}^{m\times m}$ *is a diagonal matrix with elements sampled uniformly from* $\{-1, 1\}$
- $\mathbf{H}_m \in \mathbb{R}^{m\times m}$ *is the Hadamard matrix of order $m$ defined recursively through*

$$\mathbf{H}_1 = \begin{bmatrix} 1 \end{bmatrix}, \quad \mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \ldots, \quad \mathbf{H}_{2^q} = \begin{bmatrix} \mathbf{H}_{2^{q-1}} & \mathbf{H}_{2^{q-1}} \\ \mathbf{H}_{2^{q-1}} & -\mathbf{H}_{2^{q-1}} \end{bmatrix} \in \mathbb{R}^{m\times m} \tag{4.18}$$

- $\mathbf{I}_C \in \mathbb{R}^{\tau\times m}$ *is a subsampling matrix based on a set $C \subset \{1, \ldots, m\}$ chosen uniformly at random from all sets with $\tau$ elements*

The Hadamard Sketch has recently become popular in several applications throughout numerical linear algebra because it satisfies the JL Lemma [21] and it can be computed using $\mathcal{O}(m^2 \log \tau)$ operations using the *Fast Walsh-Hadamard Transform* (FWHT) [42] or the recent *Trimmed Walsh-Hadamard Transform* (FWHT) [2]. Let $\mathbf{H}$ be the Hadamard matrix defined in (4.18) of order $m$. Such methods use a butterfly structure, like the Cooley-Tukey FFT algorithm, to compute $\mathbf{H}y$ in $\mathcal{O}(m \log m)$ time for any $y \in \mathbb{R}^m$, but they require $m$ to be a power of 2.

This last detail is often glossed over in theory, but we find in practice this poses a challenge when $m$ is large. One can remove some rows of $\mathbf{A}$ to meet this assumption, with the risk of losing vital information. Instead, when $m$ is not a power of 2, we need to pad $\mathbf{A}$ with zero rows until the augmented matrix has $m'$ rows, where $m'$ is a power of 2. That is, we need $m' = 2^{\lceil \log_2 m \rceil}$ rows. In the worst case scenario, when $m = 2^q + 1$ for some $q \in \mathbb{N}^*$ then $m' = 2^{q+1}$, effectively doubling the number of rows.

## 4.5 Convergence theory

Here we present theoretical convergence guarantees of `RidgeSketch` method (4.12) for sketch matrices $\mathbf{S} \in \mathbb{R}^{m\times\tau}$ drawn from a fixed distribution $\mathcal{D}$. Later on, in Section 4.6 we provide a new convergence theory for a new momentum variant of the sketch-and-project method. To contextualize our contribution, we will first present the previously known convergence theory of sketch-and-project method.

[4]The standard definition of Count Sketch also shuffles the columns of the summing matrix (4.16). We did not use this shuffling since we found that it had little to no effect on performance in our setting.

### 4.5.1 Convergence of the iterates

The sketch-and-project method (4.12) enjoys a linear convergence in $\ell^2$ given as follows.

**Theorem 4.7** (Convergence of the iterates, [54, 59])**.** *Let $w^*$ be a solution of (4.4) and let $w^0 \in \text{Range}(\mathbf{A})$. Let $t \in \mathbb{N}$ and consider the iterates given in (4.11). It follows that*

$$\mathbb{E}\left[\left\|w^t - w^*\right\|_{\mathbf{A}}^2\right] \leq (1-\rho)^t \left\|w^0 - w^*\right\|_{\mathbf{A}}^2 \quad , \tag{4.19}$$

*where*

$$\rho = \lambda_{\min}^+(\mathbf{A}^{1/2}\mathbb{E}\left[\mathbf{S}(\mathbf{S}^\top\mathbf{A}\mathbf{S})^\dagger\mathbf{S}\right]\mathbf{A}^{1/2}) \ . \tag{4.20}$$

*Consequently, we also have for the residual that*

$$\mathbb{E}\left[\left\|\mathbf{A}w^t - b\right\|^2\right] \leq (1-\rho)^t\lambda_{\max}(\mathbf{A})\left\|w^0 - w^*\right\|_{\mathbf{A}}^2 \quad . \tag{4.21}$$

*Proof.* The proof of (4.19) is given in Theorem 1.1 in [54]. The convergence of the residual in (4.21) follows as a consequence since

$$\left\|\mathbf{A}w^t - b\right\|^2 = \left\|w^t - w^*\right\|_{\mathbf{A}^2}^2 \leq \lambda_{\max}(\mathbf{A})\left\|w^t - w^*\right\|_{\mathbf{A}}^2 \quad . \tag{4.22}$$

□

This linear convergence in $\ell^2$ is often thought of as a gold standard for convergence of stochastic sequences, since it implies convergence in high probability and of all the moments of the sequence. Furthermore, the error decays at an exponential rate determined by $\rho$. Yet the downside of (4.19) and (4.21) is that the rate of convergence $\rho$ can be very small. Next we present a sublinear rate of convergence in $\ell^2$ that has an improved rate of convergence.

### 4.5.2 Convergence of the residuals with step size $0 < \gamma < 1$

The convergence proofs we present next rely on viewing the sketch-and-project method as an instance of *SGD* (stochastic gradient descent) [125, 166]. To establish this SGD viewpoint, we first reformulate the problem of solving (4.4) as the following minimization problem

$$\min_{w\in\mathbb{R}^d} f(w) \overset{\text{def}}{=} \frac{1}{2}\left\|\mathbf{A}w - b\right\|_{\mathbb{E}[\mathbf{H}_\mathbf{S}]}^2 = \mathbb{E}\left[\frac{1}{2}\left\|\mathbf{A}w - b\right\|_{\mathbf{H}_\mathbf{S}}^2\right] =: \mathbb{E}\left[f_\mathbf{S}(w)\right] \quad , \tag{4.23}$$

where

$$\mathbf{H}_\mathbf{S} \overset{\text{def}}{=} \mathbf{S}(\mathbf{S}^\top\mathbf{A}\mathbf{W}^{-1}\mathbf{A}\mathbf{S})^\dagger\mathbf{S}^\top \ . \tag{4.24}$$

Solving our linear systems is now equivalent to solving the stochastic minimization problem (4.23), for which the classic method is SGD. It just so happens that, SGD with a step size of $\gamma = 1$ is exactly the Sketch-and-Project iteration (4.11). Indeed, if we compute the gradients with respect to the $\|w\|_\mathbf{W} = \sqrt{\langle\mathbf{W}w, w\rangle}$ norm, then SGD is given by

$$w^{k+1} = w^k - \gamma\nabla_\mathbf{W} f_\mathbf{S}(w^k) \ , \tag{4.25}$$

where the gradient relative to the weighted inner product $\langle .,.\rangle_\mathbf{W}$ is given by

$$\nabla_\mathbf{W} f_\mathbf{S}(w^k) \overset{\text{def}}{=} \mathbf{W}^{-1}\mathbf{A}\mathbf{H}_\mathbf{S}\left(\mathbf{A}w^k - b\right) \quad , \tag{4.26}$$

where $\mathbf{S} \sim \mathcal{D}$ is sampled i.i.d at each iteration and $\gamma > 0$ is the step size. Since $f_\mathbf{S}(w)$ is an unbiased estimate of $f(w)$ we have that

$$\mathbb{E}\left[\nabla_\mathbf{W} f_\mathbf{S}(w^k)\right] \ = \ \nabla_\mathbf{W} f(w^k). \tag{4.27}$$

Using the interpretation as SGD, we can provide the convergence of $\mathbb{E}\left[f(w^k)\right]$ to zero. We can then relate the convergence of $f(w^k)$ to the convergence of the residual using the following lemma.

**Lemma 4.8.**

$$\lambda_{\min}\left(\mathbb{E}\left[\mathbf{H_S}\right]\right)\left\|\mathbf{A}w-b\right\|^2 \;\leq\; 2f(w)\;. \tag{4.28}$$

*Proof.* This follows from

$$\lambda_{\min}\left(\mathbb{E}\left[\mathbf{H_S}\right]\right)\left\|\mathbf{A}w-b\right\|^2 \;=\; \left\|\mathbf{A}w-b\right\|^2_{\lambda_{\min}(\mathbb{E}[\mathbf{H_S}])\mathbf{I}_m} \;\leq\; \left\|\mathbf{A}w-b\right\|^2_{\mathbb{E}[\mathbf{H_S}]} \;=\; 2f(w)\;.$$

□

First we need the following property of the gradient taken from Lemma 3.1 in [125].

**Lemma 4.9** (Gradient norm – function identity)**.** *It follows that*

$$\left\|\nabla_{\mathbf{W}} f_{\mathbf{S}}(w)\right\|^2_{\mathbf{W}} \;=\; 2f_{\mathbf{S}}(w)\;. \tag{4.29}$$

*Proof.* For completeness we give the proof. By straight forward computation we have that

$$\begin{aligned}
\left\|\nabla_{\mathbf{W}} f_{\mathbf{S}}(w)\right\|^2_{\mathbf{W}} &\overset{(4.26)}{=} \left\langle \mathbf{W}^{-1}\mathbf{A}\mathbf{H_S}\left(\mathbf{A}w-b\right), \mathbf{W}^{-1}\mathbf{A}\mathbf{H_S}\left(\mathbf{A}w-b\right)\right\rangle_{\mathbf{W}} \\
&= \left(\mathbf{A}w-b\right)^\top \mathbf{H_S}\mathbf{A}\mathbf{W}^{-1}\mathbf{A}\mathbf{H_S}\left(\mathbf{A}w-b\right) \\
&\overset{(4.24)}{=} \left(\mathbf{A}w-b\right)^\top \mathbf{H_S}\left(\mathbf{A}w-b\right) \\
&\overset{(4.23)}{=} 2f_{\mathbf{S}}(w)\;.
\end{aligned}$$

where in the third equality we expanded $\mathbf{H_S}$ and applied the identity $\mathbf{M}^\dagger = \mathbf{M}^\dagger\mathbf{M}\mathbf{M}^\dagger$ with $\mathbf{M} = \mathbf{S}^\top\mathbf{A}\mathbf{W}^{-1}\mathbf{A}\mathbf{S}$. □

Furthermore, the functions $f_{\mathbf{S}}$ are convex.

**Lemma 4.10.** *The function $f_{\mathbf{S}}$ is a convex quadratic. Consequently*

$$f_{\mathbf{S}}(y) \geq f_{\mathbf{S}}(x) + \left\langle \nabla_{\mathbf{W}} f_{\mathbf{S}}(x), y-x\right\rangle_{\mathbf{W}}\;. \tag{4.30}$$

*Proof.* Let $w \in \mathbb{R}^d$ and $\mathbf{S} \sim \mathcal{D}$, the gradient relative to the Euclidean inner product $\langle .,. \rangle$ is

$$\nabla f_{\mathbf{S}}(x) = \mathbf{A}\mathbf{H_S}\left(\mathbf{A}x-b\right)\;,$$

and thus the hessian is

$$\nabla^2 f_{\mathbf{S}}(x) = \mathbf{A}\mathbf{H_S}\mathbf{A}\;,$$

which is a semi-definite positive matrix. This implies that $f_{\mathbf{S}}$ is convex. As a consequence (4.30) holds since

$$\begin{aligned}
f_{\mathbf{S}}(y) &\geq f_{\mathbf{S}}(x) + \left\langle \nabla f_{\mathbf{S}}(x), y-x\right\rangle \\
&= f_{\mathbf{S}}(x) + \left\langle \nabla_{\mathbf{W}} f_{\mathbf{S}}(x), y-x\right\rangle_{\mathbf{W}}\;,
\end{aligned}$$

and given that $\left\langle \nabla_{\mathbf{W}} f_{\mathbf{S}}(x), y-x\right\rangle_{\mathbf{W}} = \left\langle \mathbf{W}\mathbf{W}^{-1}\nabla f_{\mathbf{S}}(x), y-x\right\rangle = \left\langle \nabla f_{\mathbf{S}}(x), y-x\right\rangle$. □

Next we establish the sublinear convergence of the *average of the iterates* of Sketch-and-project method. This result is a direct consequence of Theorem 4.10 in [125] and has already been proven in Theorem 3 in [91]. We present the complete statement and proof since it is a warm-up for our forthcoming results, and since the proof is substantially simpler than the one presented in [91].

**Theorem 4.11** (Convergence of the residuals)**.** *Let $\gamma \in (0,1)$. Let $w^*$ be a solution of (4.4), let $t \in \mathbb{N}^*$ and let $(w^k)_{0\leq k\leq t}$ be the iterates given by (4.25). It follows that*

$$f\left(\overline{w}^t\right) \;\leq\; \frac{1}{t}\sum_{k=0}^{t-1} f\left(w^k\right) \;\leq\; \frac{1}{t}\frac{\left\|w^0-w^*\right\|_{\mathbf{W}}^2}{2\gamma(1-\gamma)} \;, \tag{4.31}$$

*where $\overline{w}^t \overset{def}{=} \frac{1}{t}\sum_{k=0}^{t-1} w^k$. Furthermore*

$$\left\|\mathbf{A}\overline{w}^t - b\right\|^2 \;\leq\; \frac{1}{t}\frac{1}{\lambda_{\min}\left(\mathbb{E}\left[\mathbf{H}_{\mathbf{S}}\right]\right)}\frac{\left\|w^0-w^*\right\|_{\mathbf{W}}^2}{2\gamma(1-\gamma)} \;. \tag{4.32}$$

*Proof.*

$$\begin{aligned}\left\|w^{k+1}-w^*\right\|_{\mathbf{W}}^2 &= \left\|w^k-w^*\right\|_{\mathbf{W}}^2 - 2\gamma\left\langle\nabla_{\mathbf{W}} f_{\mathbf{S}}\left(w^k\right), w^k-w^*\right\rangle_{\mathbf{W}} + \gamma^2\left\|\nabla_{\mathbf{W}} f_{\mathbf{S}}\left(w^k\right)\right\|_{\mathbf{W}}^2 \\ &\overset{(4.29)}{=} \left\|w^k-w^*\right\|_{\mathbf{W}}^2 - 2\gamma\left\langle\nabla_{\mathbf{W}} f_{\mathbf{S}}\left(w^k\right), w^k-w^*\right\rangle_{\mathbf{W}} + 2\gamma^2 f_{\mathbf{S}}\left(w^k\right) \\ &\overset{(4.30)}{\leq} \left\|w^k-w^*\right\|_{\mathbf{W}}^2 - 2\gamma(1-\gamma) f_{\mathbf{S}}\left(w^k\right) \;.\end{aligned}$$

Re-arranging, and dividing through by $\gamma(1-\gamma) > 0$ we have that

$$f_{\mathbf{S}}\left(w^k\right) \;\leq\; \frac{1}{2\gamma(1-\gamma)}\left(\left\|w^k-w^*\right\|_{\mathbf{W}}^2 - \left\|w^{k+1}-w^*\right\|_{\mathbf{W}}^2\right) \;.$$

Summing up over both sides for $k = 0,\ldots,t-1$ and using telescopic cancellation we have that

$$\sum_{k=0}^{t-1} f_{\mathbf{S}}\left(w^k\right) \;\leq\; \frac{1}{2\gamma(1-\gamma)}\left(\left\|w^0-w^*\right\|_{\mathbf{W}}^2 - \left\|w^t-w^*\right\|_{\mathbf{W}}^2\right) \;\leq\; \frac{\left\|w^0-w^*\right\|_{\mathbf{W}}^2}{2\gamma(1-\gamma)} \;. \tag{4.33}$$

By applying the Jensen's inequality to $f_{\mathbf{S}}$, which is convex since it is a quadratic form,

$$f_{\mathbf{S}}\left(\frac{1}{t}\sum_{k=0}^{t-1} w^k\right) \leq \frac{1}{t}\sum_{k=0}^{t-1} f_{\mathbf{S}}\left(w^k\right) \;, \tag{4.34}$$

Now, by denoting $\overline{w}^t \overset{\text{def}}{=} \frac{1}{t}\sum_{k=0}^{t-1} w^k$, the convergence result (4.31) follows by dividing (4.33) by $t$ and using (4.34). Finally, the convergence of the residual in (4.32) follows from (4.28). □

The weakness of Theorem 4.11 is that it describes how the average of the iterates $\overline{w}^t$ converge instead of the last one $w^t$. This type of convergence is problematic since $\overline{w}^t$ gives as much importance, a $1/t$ weight, to the initial point $w^0$ as to the last one $w^t$. Since $w^0$ is often chosen arbitrarily, the average of the iterates can converge only as fast as $w^0$ is forgotten. That is, the average cannot converge faster than $1/t$. This is apparent in experiments, where using averaging from the start results in a slow convergence. In practice, averaging only the last few iterates works substantially better, but it is not supported in theory. To resolve this issue, we will replace this *equal* averaging with a *weighted* average that gives more weight to recent iterates, and forgets the initial conditions exponentially fast.

## 4.6 Momentum

A common variant of SGD is to add momentum. Since the sketch-and-project method can be interpreted as SGD (4.25), we can add momentum. Let $\gamma_k \in [0,\ 1]$ and $\beta_k \in [0,\ 1]$ be respectively the step size and the momentum parameter. The heavy ball formulation of momentum is given by

$$w^{k+1} = w^k - \gamma_k \nabla_{\mathbf{W}} f_{\mathbf{S}}\left(w^k\right) + \beta_k\left(w^k - w^{k-1}\right) \;. \tag{4.35}$$

Note that we have now allowed for a step size $\gamma_k$ that is iteration dependent.

This same heavy ball formulation was considered in [91], where the authors propose a precise analysis and show no benefit using momentum. But, all of their analysis assumes that the momentum parameter is constant. It turns out, that by allowing $\beta_k$ to be iteration dependent, we can do better. But first, we need the *iterative averaging viewpoint* of momentum.

### 4.6.1 Iterate averaging viewpoint

Recently, a new *iterate averaging* parametrization of the momentum method was proposed in [135]. This iterative averaging parametrization is given by

$$z^k = z^{k-1} - \eta_k \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) \tag{4.36}$$

$$w^{k+1} = \left(1 - \frac{1}{\zeta_{k+1}+1}\right) w^k + \frac{1}{\zeta_{k+1}+1} z^k \ , \tag{4.37}$$

where we have introduced two new parameter sequences $\eta_k$ and $\zeta_k$ that map back to the $\gamma_k$ and $\beta_k$ parameters via

$$\gamma_k = \frac{\eta_k}{\zeta_{k+1}+1} \quad \text{and} \quad \beta_k = \frac{\zeta_k}{\zeta_{k+1}+1} \ . \tag{4.38}$$

Next we show that (4.37) produces the same $w_k$ iterates as (4.35).

**Proposition 4.12** (Equivalent formulations)**.** *Let $t \in \mathbb{N}^*$. The steps given in* (4.35) *and in* (4.36)–(4.37) *generate the same iterates $(w_k)_{0 \le k \le t}$.*

*Proof.* First, let us expand the *iterate averaging* parametrization

$$\begin{aligned} w^{k+1} &\overset{(4.37)}{=} \left(1 - \frac{1}{\zeta_{k+1}+1}\right) w^k + \frac{1}{\zeta_{k+1}+1} z^k \\ &\overset{(4.36)}{=} w^k - \frac{1}{\zeta_{k+1}+1} w^k + \frac{1}{\zeta_{k+1}+1} \left(z^{k-1} - \eta \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k)\right) \\ &= w^k - \frac{\eta}{\zeta_{k+1}+1} \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) + \frac{1}{\zeta_{k+1}+1} \left(z^{k-1} - w^k\right) \\ &= w^k - \gamma_k \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) + \frac{1}{\zeta_{k+1}+1} \left(z^{k-1} - w^k\right) \ , \end{aligned} \tag{4.39}$$

where in the last step we used the left-hand side relation in (4.38). To conclude the proof, we need only show that

$$\frac{1}{\zeta_{k+1}+1} \left(z^{k-1} - w^k\right) = \beta_k (w_k - w_{k-1}) \ .$$

Indeed, this follows $w^k = \left(1 - \frac{1}{\zeta_k+1}\right) w^{k-1} + \frac{1}{\zeta_k+1} z^{k-1}$, which can be rearranged in $z^{k-1} - w^k = \zeta_k \left(w^k - w^{k-1}\right)$, consequently

$$\frac{1}{\zeta_{k+1}+1} \left(z^{k-1} - w^k\right) = \frac{\zeta_k}{\zeta_{k+1}+1} \left(w^k - w^{k-1}\right) \overset{(4.38)}{=} \beta_k (w_k - w_{k-1}) \ ,$$

which together with (4.39) concludes the proof. □

Next we show how to leverage the iterative averaging viewpoint (4.36)–(4.37) to prove the convergence of the last iterates of sketch-and-project with momentum (4.35).

### 4.6.2 Convergence theorem

**Theorem 4.13.** *Consider the iterates* (4.36)–(4.37) *with* $w^0 = w^{-1}$. *Let* $\eta_k$ *be a sequence of parameters with* $0 < \eta_k < 1$. *If*

$$\zeta_0 = 0 \quad \textit{and} \quad \zeta_k = \frac{1}{\eta_k} \sum_{t=0}^{k-1} \eta_t (1 - \eta_t) \ , \quad \textit{for all } k \geq 1 \ . \tag{4.40}$$

*then*

$$\left\| \mathbf{A} w^k - b \right\|^2 \ \leq \ \frac{1}{\lambda_{\min}\left(\mathbb{E}\left[\mathbf{H}_{\mathbf{S}}\right]\right)} \frac{\left\| w^0 - w^* \right\|_{\mathbf{W}}^2}{\sum_{t=0}^{k} \eta_t (1 - \eta_t)} \ . \tag{4.41}$$

*Proof.* This proof is based on Theorem 3.1 in [135] for SGD applied to convex and smooth functions. Consider the Lyapunov function

$$L_k \ = \ \mathbb{E}\left[ \left\| z^k - w^* \right\|_{\mathbf{W}}^2 \right] + 2\eta_k \zeta_k \mathbb{E}\left[ f(w^k) \right] \ . \tag{4.42}$$

First note that

$$\begin{aligned}
\left\| z^k - w^* \right\|_{\mathbf{W}}^2 &\overset{(4.36)}{=} \left\| z^{k-1} - \eta_k \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) - w^* \right\|_{\mathbf{W}}^2 \\
&= \left\| z^{k-1} - w^* \right\|_{\mathbf{W}}^2 - 2\eta_k \left\langle z^{k-1} - w^*, \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) \right\rangle_{\mathbf{W}} + \eta_k^2 \left\| \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) \right\|_{\mathbf{W}}^2 \\
&\overset{(4.37)}{=} \left\| z^{k-1} - w^* \right\|_{\mathbf{W}}^2 - 2\eta_k \left\langle w^k - w^* + \zeta_k (w^k - w^{k-1}, \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) \right\rangle_{\mathbf{W}} + \eta_k^2 \left\| \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) \right\|_{\mathbf{W}}^2 \\
&= \left\| z^{k-1} - w^* \right\|_{\mathbf{W}}^2 - 2\eta_k \left\langle w^k - w^*, \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) \right\rangle_{\mathbf{W}} - 2\eta_k \zeta_k \left\langle w^k - w^{k-1}, \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) \right\rangle_{\mathbf{W}} \\
&\quad + \eta_k^2 \left\| \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) \right\|_{\mathbf{W}}^2 \\
&\overset{(4.29)}{=} \left\| z^{k-1} - w^* \right\|_{\mathbf{W}}^2 + 2\eta_k^2 f_{\mathbf{S}}(w^k) - 2\eta_k \left\langle w^k - w^*, \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) \right\rangle_{\mathbf{W}} \\
&\quad - 2\eta_k \zeta_k \left\langle w^k - w^{k-1}, \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) \right\rangle_{\mathbf{W}} \ .
\end{aligned}$$

The above holds to equality. Now we introduce the first inequality by calling upon (4.30) so that

$$\begin{aligned}
\left\| z^k - w^* \right\|_{\mathbf{W}}^2 &\overset{(4.30)}{\leq} \left\| z^{k-1} - w^* \right\|_{\mathbf{W}}^2 + 2\eta_k^2 f_{\mathbf{S}}(w^k) - 2\eta_k f_{\mathbf{S}}(w^k) - 2\eta_k \zeta_k (f_{\mathbf{S}}\left(w^k\right) - f_{\mathbf{S}}(w^{k-1})) \\
&= \left\| z^{k-1} - w^* \right\|_{\mathbf{W}}^2 - 2\eta_k (1 + \zeta_k - \eta_k) f_{\mathbf{S}}(w^k) + 2\eta_k \zeta_k f_{\mathbf{S}}(w^{k-1}) \ .
\end{aligned} \tag{4.43}$$

The restriction on the parameters in (4.40) was designed so that

$$\eta_k (1 + \zeta_k - \eta_k) \ = \ \eta_{k+1} \zeta_{k+1} \ . \tag{4.44}$$

Indeed, from (4.40) we have that

$$\zeta_{k+1} = \frac{1}{\eta_{k+1}} \sum_{t=0}^{k} \eta_t (1 - \eta_t) \ = \frac{\eta_k}{\eta_{k+1}} \frac{1}{\eta_k} \sum_{t=0}^{k} \eta_t (1 - \eta_t) \ = \frac{\eta_k}{\eta_{k+1}} \left( \zeta_k + 1 - \eta_k \right) \ .$$

Using (4.44) in (4.43) and taking expectation gives

$$\mathbb{E}\left[ \left\| z^k - w^* \right\|_{\mathbf{W}}^2 \right] + 2\eta_{k+1} \zeta_{k+1} \mathbb{E}\left[ f(w^k) \right] \ \leq \ \mathbb{E}\left[ \left\| z^{k-1} - w^* \right\|_{\mathbf{W}}^2 \right] + 2\eta_k \zeta_k \mathbb{E}\left[ f(w^{k-1}) \right] \ . \tag{4.45}$$

Summing up both sides from $k = 0, \ldots, t$ and using telescopic cancellation gives

$$\mathbb{E}\left[ \left\| z^t - w^* \right\|_{\mathbf{W}}^2 \right] + 2\eta_{t+1} \zeta_{t+1} \mathbb{E}\left[ f(w^t) \right] \ \leq \ \mathbb{E}\left[ \left\| z^{-1} - w^* \right\|_{\mathbf{W}}^2 \right] + 2\eta_0 \zeta_0 \mathbb{E}\left[ f(w^{-1}) \right] \ . \tag{4.46}$$

Using that $z^{-1} = w^0$ from (4.37), $\zeta_0 = 0$ and re-arranging gives

$$\mathbb{E}\left[\left\|z^t - w^*\right\|_{\mathbf{W}}^2\right] + 2\eta_{t+1}\zeta_{t+1}\mathbb{E}\left[f(w^t)\right] \quad \leq \quad \mathbb{E}\left[\left\|w^0 - w^*\right\|_{\mathbf{W}}^2\right] \ . \tag{4.47}$$

Substituting the definition of $\zeta_{t+1}$ from (4.40) gives

$$2\mathbb{E}\left[f(w^t)\right] \quad \leq \quad \frac{1}{\eta_{t+1}\zeta_{t+1}}\mathbb{E}\left[\left\|w^0 - w^*\right\|_{\mathbf{W}}^2\right] = \frac{1}{\sum_{k=0}^{t}\eta_k(1-\eta_k)}\mathbb{E}\left[\left\|w^0 - w^*\right\|_{\mathbf{W}}^2\right] \ . \tag{4.48}$$

The final step is a result of using (4.28) to lower bound $\mathbb{E}\left[f(w^t)\right]$. □

Theorem 4.13 shows that the convergences of the iterates $w^t$ depend on a sequence of parameters $\eta_k$. Next we give a corollary that shows that by simply choosing $\eta_k \equiv \eta < 1$ the iterates $w^t$ enjoy a fast sublinear convergence.

**Corollary 4.14.** *Consider the setting of Theorem 4.13. Let $\eta_k \equiv \eta < 1$ and thus*

$$\zeta_0 = 0 \quad \textit{and} \quad \zeta_k = k(1-\eta) \ , \quad \textit{for all } k \geq 1 \ , \tag{4.49}$$

*and*

$$\gamma_k = \frac{\eta}{(k+1)(1-\eta)+1} \quad \textit{and} \quad \beta_k = 1 - \frac{2-\eta}{(k+1)(1-\eta)+1} \ , \tag{4.50}$$

*then*

$$\left\|\mathbf{A}w^k - b\right\|^2 \quad \leq \quad \frac{1}{\lambda_{\min}\left(\mathbb{E}\left[\mathbf{H}_{\mathbf{S}}\right]\right)}\frac{1}{k}\frac{\left\|w^0 - w^*\right\|_{\mathbf{W}}^2}{\eta(1-\eta)} \ . \tag{4.51}$$

*Consequently, for $\eta = \frac{1}{2}$ and for a given tolerance $\epsilon > 0$, the iteration complexity of minimizing the residual is given by*

$$t \ \geq \ \frac{4\left\|w^0 - w^*\right\|_{\mathbf{W}}^2}{\lambda_{\min}\left(\mathbb{E}\left[\mathbf{H}_{\mathbf{S}}\right]\right)}\frac{1}{\epsilon} \ , \tag{4.52}$$

*to reach a desired precision*

$$\mathbb{E}\left[\left\|\mathbf{A}w^t - b\right\|^2\right] < \epsilon \ .$$

Corollary 4.14 shows that the *last iterate* of sketch-and-project with momentum converges at the same rate as the average of the iterates of sketch-and-project (see Theorem 4.11). This is the first tangible theoretical advantage of using momentum in this setting.

In Section 4.7, through a more specialized setting for ridge regression, we show that the complexity given in (4.52) can be even tighter than the previously known linear convergence in Theorem 4.7. But first, we present a practical pseudo-code implementation of sketch-and-project with momentum.

### 4.6.3 Implementation

In Algorithm 12 we have the pseudo-code of Sketch-and-Project with momentum (4.35) for solving ridge regression (4.1) where $\mathbf{W} = \mathbf{A}$.

The residual $r^k \overset{\text{def}}{=} \left\|\mathbf{A}w^k - b\right\|^2$ is required for computing the update like in Algorithm 11. Fortunately we can efficiently compute the residual at step $k+1$ by storing the residuals at two steps $k-1$ and $k$

**Algorithm 12** Sketch-and-project method with momentum

**Parameters:** distribution over random $m \times \tau$ matrices in $\mathcal{D}$, tolerance $\epsilon > 0$, momentum parameters $\eta_k, \zeta_k \in [0, 1]$
Set $w^{-1} = w^0 = 0 \in \mathbb{R}^m$ ▷ weights initialization
Set $r^{-1} = r^0 = \mathbf{A}w^0 - b = -b \in \mathbb{R}^m$ ▷ residual initialization
$k = 0$
**While** $\left\|r^k\right\|_2 / \left\|r^0\right\|_2 \leq \epsilon$ **do**
  $\gamma_k = \frac{\eta_k}{1+\zeta_{k+1}}$ and $\beta_k = \frac{\zeta_k}{1+\zeta_{k+1}}$
  Sample an independent copy $\mathbf{S}_k \sim \mathcal{D}$
  $r^k_{\mathbf{S}} = \mathbf{S}_k^\top r^k$ ▷ compute sketched residual
  $\delta_k = \texttt{least\_norm\_solution}\left(\mathbf{S}_k^\top \mathbf{A}\mathbf{S}_k, r^k_{\mathbf{S}}\right)$ ▷ solve sketched system
  $w^{k+1} = (1+\beta_k)w^k - \beta_k w^{k-1} - \gamma_k \mathbf{S}_k \delta_k$ ▷ update the iterates
  $r^{k+1} = (1+\beta_k)r^k - \beta_k r^{k-1} - \gamma_k \mathbf{A}\mathbf{S}_k \delta_k$ ▷ update the residual
  $k = k + 1$
**Output:** $w^t$ ▷ return weights vector

since

$$\begin{aligned} r^{k+1} &= \mathbf{A}w^{k+1} - b \\ &\overset{(4.35)}{=} \mathbf{A}((1+\beta_k)w^k - \beta_k w^{k-1} - \gamma_k \nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k)) - b \\ &\overset{(4.26)}{=} \mathbf{A}((1+\beta_k)w^k - \beta_k w^{k-1} - \gamma_k \mathbf{S}_k \delta_k) - b \\ &\overset{(4.26)}{=} (1+\beta_k)r^k - \beta_k r^{k-1} - \gamma_k \mathbf{A}\mathbf{S}_k \delta_k \ , \end{aligned} \tag{4.53}$$

where

$$\delta_k = \left(\mathbf{S}_k^\top \mathbf{A}\mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top r^k = \texttt{least\_norm\_solution}\left(\mathbf{S}_k^\top \mathbf{A}\mathbf{S}_k, r^k_{\mathbf{S}}\right) \ ,$$

and where we used that since $\mathbf{W} = \mathbf{A}$ we have that

$$\nabla_{\mathbf{W}} f_{\mathbf{S}}(w^k) \overset{(4.26)}{=} \mathbf{W}^{-1}\mathbf{A}\mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{W}^{-1}\mathbf{A}\mathbf{S})^\dagger \mathbf{S}^\top \left(\mathbf{A}w^k - b\right) = \mathbf{S}_k \left(\mathbf{S}_k^\top \mathbf{A}\mathbf{S}_k\right)^\dagger \mathbf{S}_k^\top r^k \ .$$

Thus, for the momentum version of our algorithm we can keep the residual update at the cost of $O(m\tau)$ by just storing the residual $r^{k-1}$ at the previous time step.

## 4.7 Specialized convergence theory for single column sketches

Here we take a closer look at the rates of convergence given by Theorem 4.7 and Corollary 4.14 by considering a specialized setting of ridge regression ($\mathbf{W} = \mathbf{A}$) and single column sketches. That is, in this section we use a discrete distribution for $\mathbf{S}$ given by

$$\mathbb{P}[\mathbf{S} = s_i] = p_i, \quad \text{for } i = 1, \ldots, q \ , \tag{4.54}$$

where $s_1, \ldots, s_q \in \mathbb{R}^m$ are a fixed collection of vectors and $\sum_{i=1}^q p_i = 1$.

To better understand the complexity (4.52) and (4.22) , we first need to find a lower bound for $\lambda_{\min}\left(\mathbb{E}\left[\mathbf{H}_{\mathbf{S}}\right]\right)$ where

$$\mathbb{E}\left[\mathbf{H}_{\mathbf{S}}\right] = \mathbb{E}\left[\mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{S})^\dagger \mathbf{S}^\top\right] \ . \tag{4.55}$$

By using a special choice for the probabilities in (4.54) , we are able to give a convenient lower bound for $\mathbb{E}\left[\mathbf{H}_{\mathbf{S}}\right]$ in the following lemma.

**Lemma 4.15.** *Let* $\mathbf{S}$ *have a discrete distribution according to*

$$\mathbb{P}[\mathbf{S} = s_i] \;=\; \frac{s_i^\top \mathbf{A} s_i}{\sum_{j=1}^q s_j^\top \mathbf{A} s_j} \;, \quad \text{for } i = 1, \dots, q \;, \tag{4.56}$$

*where* $s_1, \dots, s_q \in \mathbb{R}^m$ *are unit column vectors. Let* $\mathbf{F} \overset{def}{=} [s_1, \dots, s_q] \in \mathbb{R}^{m \times q}$. *It follows that*

$$\lambda_{\min}(\mathbb{E}\left[\mathbf{H}_{\mathbf{S}}\right]) \;=\; \frac{\lambda_{\min}(\mathbf{F}\mathbf{F}^\top)}{\sum_{j=1}^q s_j^\top \mathbf{A} s_j} \;. \tag{4.57}$$

*Proof.* This *convenient* probability distribution (4.56) was already considered in [59] in Section 5.2. But there in, the authors used these probabilities to study a different spectral quantity, thus for completion we adapt their proof to our setting. First note that

$$\mathbb{E}\left[\mathbf{H}_{\mathbf{S}}\right] \overset{(4.55)}{=} \sum_{i=1}^q \frac{p_i}{s_i^\top \mathbf{A} s_i} s_i s_i^\top \overset{(4.56)}{=} \frac{\sum_{i=1}^q s_i s_i^\top}{\sum_{j=1}^q s_j^\top \mathbf{A} s_j} \;.$$

Consequently, the smallest eigenvalue is given by

$$\lambda_{\min}(\mathbb{E}\left[\mathbf{H}_{\mathbf{S}}\right]) \;=\; \lambda_{\min}\left(\frac{\sum_{i=1}^q s_i s_i^\top}{\sum_{j=1}^q s_j^\top \mathbf{A} s_j}\right) \;=\; \frac{\lambda_{\min}(\mathbf{F}\mathbf{F}^\top)}{\sum_{j=1}^q s_j^\top \mathbf{A} s_j} \;.$$

□

### 4.7.1 Complexity of coordinate descent with and without momentum

Here we compare the fast sublinear convergence of `RidgeSketch` with momentum given in Theorem (4.13) to the linear convergence of `RidgeSketch` without momentum given in Theorem 4.7. Though linear convergence is generally preferred, we will show here that our new sublinear rate of convergence can be faster. To illustrate this, we will focus on the special case of Coordinate Descent (CD).

The CD method is the result of applying `RidgeSketch` when the sketching matrices are unit coordinate vectors. That is, when $s_i = e_i \in \mathbb{R}^m$ is the $i$-th column of the identity matrix and

$$\mathbb{P}[s_i = e_i] \;=\; \frac{\mathbf{A}_{ii}}{\text{Trace}\left(\mathbf{A}\right)} \;, \quad \text{for } i = 1, \dots, m \;. \tag{4.58}$$

This particular nonuniform sampling in (4.58) was first given in [87]. With this sketch, the `RidgeSketch` method with momentum (4.35) becomes CD with momentum which is given by

$$w^{k+1} = w^k - \gamma_k \frac{\mathbf{A}_{i:} w^k - b_i}{\mathbf{A}_{ii}} e_i + \beta_k (w^k - w^{k-1}) \;. \tag{4.59}$$

The CD method with constant momentum $\beta_k \equiv \beta$ is known to converge linearly [87, 91]. But the linear convergence of CD with momentum is always slower than CD without momentum, see Theorem 1 in [91]. Here we present the first convergence rate of CD with momentum that can be faster than CD without momentum. But first, we need the following corollary.

**Corollary 4.16.** *Let* $\epsilon > 0$. *If we set the momentum parameters* $\gamma_k$ *and* $\beta_k$ *according to* (4.50) *with* $\eta \equiv 0.5$ *then the iterates of CD with momentum* (4.59) *satisfy*

$$t \geq 4 \frac{\text{Trace}\left(\mathbf{A}\right)}{\epsilon} \quad \Longrightarrow \quad \frac{\mathbb{E}\left[\left\|\mathbf{A}w^t - b\right\|^2\right]}{\left\|w^0 - w^*\right\|_{\mathbf{A}}^2} < \epsilon \;. \tag{4.60}$$

*Alternatively, if we use no momentum (setting $\gamma_k = 1$ and $\beta_k = 0$) then the iterates* (4.59) *satisfy*

$$t \;\geq\; \frac{\mathrm{Trace}\,(\mathbf{A})}{\lambda_{\min}(\mathbf{A})}\log\left(\frac{\lambda_{\max}(\mathbf{A})}{\epsilon}\right) \;\;\Longrightarrow\;\; \frac{\mathbb{E}\left[\|\mathbf{A}w^t - b\|^2\right]}{\|w^0 - w^*\|^2_{\mathbf{A}}} < \epsilon \;\; . \tag{4.61}$$

*Proof.* Consider the sampling given by (4.58). From (4.57), since $\mathbf{F} = \mathbf{I}_m$, we have that

$$\lambda_{\min}(\mathbb{E}\left[\mathbf{H}_{\mathbf{S}}\right]) \;\geq\; \frac{1}{\mathrm{Trace}\,(\mathbf{A})} \;\; . \tag{4.62}$$

Consequently, using (4.62) together with the complexity bound (4.52) and setting $\mathbf{W} = \mathbf{A}$ we have (4.60).

The linear complexity (4.61) follows from a special case of Theorem 4.7. Indeed, by using (4.62) we have that

$$\rho = \frac{\lambda_{\min}(\mathbf{A})}{\mathrm{Trace}\,(\mathbf{A})} \;\; .$$

The resulting complexity (4.61) follows by standard manipulations of logarithm. □

This linear convergence (4.61) is generally preferred because of the resulting logarithmic dependency of $\epsilon$. But, as we show next, the sublinear complexity given in (4.60) can be tighter when $\epsilon$ is not too small.

**Corollary 4.17** (Domain of superiority of the sublinear over the linear convergence)**.** *Consider the setting of Corollary 4.17. Let $\hat{\epsilon} \overset{def}{=} \frac{\epsilon}{\lambda_{\max}(\mathbf{A})}$ be the* scaled precision *and let use denote $\kappa \overset{def}{=} \frac{\lambda_{\max}(\mathbf{A})}{\lambda_{\min}(\mathbf{A})}$ the* condition number*. If*

$$\hat{\epsilon}(1-\hat{\epsilon}) \;\geq\; \frac{4}{\kappa} \;\; , \tag{4.63}$$

*then the complexity bound of momentum* (4.60) *is tighter than the bound in* (4.61)*.*

*Furthermore, if $\kappa \geq 16$, the solutions to* (4.63) *in $\hat{\epsilon}$ are given by*

$$\frac{1}{2} - \frac{1}{2}\sqrt{1 - \frac{16}{\kappa}} \leq \hat{\epsilon} \leq \frac{1}{2} + \frac{1}{2}\sqrt{1 - \frac{16}{\kappa}} \;\; . \tag{4.64}$$

*Proof.* The complexity bound in (4.60) is tighter than the bound in (4.61) if

$$\frac{\mathrm{Trace}\,(\mathbf{A})}{\lambda_{\min}(\mathbf{A})}\log\left(\frac{\lambda_{\max}(\mathbf{A})}{\epsilon}\right) \;\geq\; 4\frac{\mathrm{Trace}\,(\mathbf{A})}{\epsilon} \;\; .$$

Substituting $\hat{\epsilon} \overset{\text{def}}{=} \frac{\epsilon}{\lambda_{\max}(\mathbf{A})}$ and re-arranging the above gives

$$\hat{\epsilon}\log\left(\frac{1}{\hat{\epsilon}}\right) \;\geq\; \frac{4}{\kappa} \;\; . \tag{4.65}$$

To further bound the above we use the following standard logarithm bound

$$\frac{1}{1-x}\log\left(\frac{1}{x}\right) \geq 1 \;\; , \quad \forall x \in (0,\, 1) \;\; ,$$

which after manipulations gives

$$x\log\left(\frac{1}{x}\right) \geq x(1-x) \;\; .$$

Assuming that $\hat{\epsilon} = \epsilon/\lambda_{\max}(\mathbf{A}) \leq 1$ and using this bound with $x = \hat{\epsilon}$ we have that if (4.63) holds then

$$\hat{\epsilon}\log\left(\frac{1}{\hat{\epsilon}}\right) \;\geq\; \hat{\epsilon}(1-\hat{\epsilon}) \;\geq\; \frac{4}{\kappa} \;\; .$$

Thus (4.65) holds. □

Using Corollary 4.17 we can deduce several regimes where the sublinear momentum bound in (4.60) is tighter than the bound in (4.61). As illustrated in Figure 4.1, when the condition number is moderate to large or the scaled precision is moderate, the sublinear bound (4.60) is often tighter. On the other hand, when $\hat{\epsilon}$ is very small, then linear rates such as (4.61) are generally preferred.

The regime where sketch-and-projecting methods are interesting is when $\epsilon$ is moderate, and $\mathbf{A}$ has large dimensions. Furthermore, in the large dimensional setting, the condition number of $\mathbf{A}$ can also be very large, thus (4.63) is likely to hold.

## 4.8 RidgeSketch momentum experiments

Our first experiment explores the efficiency of the momentum version of our `RidgeSketch` method. We then compare the different sketches described in Section 4.4. Finally, we prove the efficiency of our method on large scale real datasets and show it is competitive with CG and direct solvers.

**Datasets.** In what follows, we test our algorithms on the datasets with different number of data samples $n$ and number of features $d$: `California Housing` ($n = 20,640$, $d = 8$), `Boston` ($n = 506$, $d = 13$), `RCV1` ($n = 804,414$, $d = 47,236$) fetched from `sklearn`[5] [114] and `Year Prediction MSD` ($n = 515,345$, $d = 90$) from the UCI repository[6]. We converted `RCV1` into a regression task by transforming multi-class labels into integers.

### 4.8.1 Experiment 1: comparison of different momentum settings

In this section, we compare the sketch-and-project method with momentum for three settings: our new iteration dependent parameters given by (4.38), the constant $\beta$ setting proposed in [91] ($\gamma_k = 1$, $\beta_k = 0.5$) and no momentum at all ($\gamma_k = 1$, $\beta_k = 0$). We report iteration plots since the sketch-and-project methods with or without momentum have almost the same iteration cost. Indeed in either case, this cost is dominated by sketching $\mathbf{A}$, computing the residual $r^k$ and solving the sketched system (4.14). We report error areas (1st and 3rd quartiles) computed over 10 runs each.

**Increasing momentum.** As suggested by Theorem 4.13, our momentum Algorithm 12 requires choosing the sequence $\eta_k$, after which $\gamma_k$ and $\beta_k$ are set using (4.40) and (4.38). After running several benchmarks tests, we identified the following *theoretical* rule for setting $\eta_k$

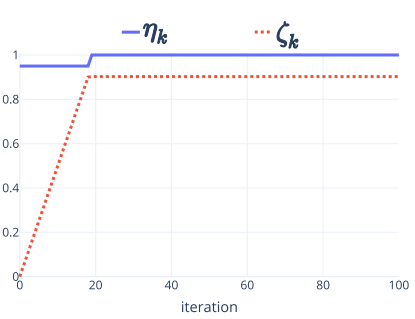
$$\eta_k = \begin{cases} 0.995 & \text{if } \beta_k < 0.5 \\ 1 & \text{if } \beta_k \geq 0.5 \end{cases} . \tag{4.66}$$

We call this parameter choice increasing momentum as it allows $\beta_k$ to increase from 0 to 0.5 while the step size $\gamma_k$ decreases from 1 to 0.5, as showed in Figure 4.3. Yet, we found in our experiments that this theoretical setting (4.66) closely matches the version without momentum of Algorithm 11. We suppose that the gain of the increasing momentum is lost by an excessively rapid drop of the step size to 0. This is why we introduce the following *heuristic* setting that keeps the step size to 1 and still uses the theoretical setting for momentum when $\eta_k \equiv \eta$ given by (4.50), that is

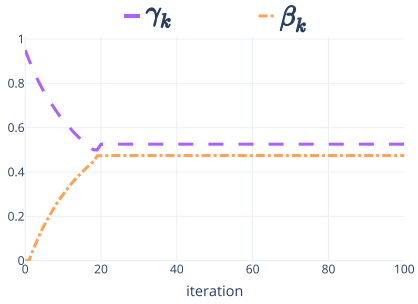
$$\gamma_k \equiv 1 \quad \text{and} \quad \beta_k = 1 - \frac{2-\eta}{(k+1)(1-\eta)+1} . \tag{4.67}$$

We tested it for increasing momentum on the `Boston` and `RCV1` datasets with different sketches and sketch sizes, see Figures 4.4 and 4.5. We found that this heuristic setting (4.67) had the *best of both worlds*, in that in the first iterations, when $\gamma_k \approx 1$ and $\beta_k \approx 0$, it benefits from the fast initial decrease

[5] `https://scikit-learn.org/stable/modules/generated/sklearn.datasets`.
[6] `https://archive.ics.uci.edu/ml/datasets/yearpredictionmsd`.

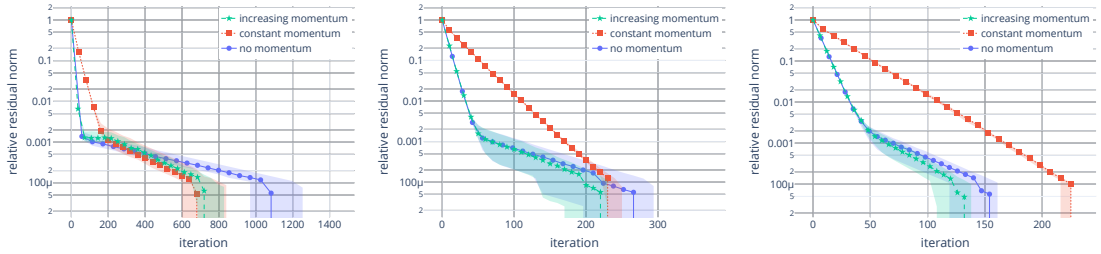



**(a)** *Parameters* $\eta_k$ *and* $\zeta_k$




**(b)** *Step size and momentum parameter*

**Figure 4.3:** *Momentum parameters for the theoretical rule:* $\eta_k \equiv 0.5$ *while* $\beta_k < 0.5$*, then* $\eta_k \equiv 1$.




**(a)** $\lambda = 0$ **(b)** $\lambda = 0.7$ **(c)** $\lambda = 70000$

**Figure 4.4:** *Effect of regularizer* $\lambda$ *on increasing momentum (green), constant momentum (red) and no-momentum (blue) (*`Boston` *dataset with kernel,* $m = 506$*,* $\tau = \lfloor m/4 \rfloor = 126$*,* `Count sketch`*).*

of the no-momentum version. Then, in later iterations, it exploits the fast asymptotic convergence of momentum since $\beta_k \approx 0.5$.

**Regularization test.** Using the heuristic setting (4.67), we tested the impact of using a small, medium and larger regularization parameter $\lambda$ on the performance of momentum, see Figure 4.4. In this figure, we can see that constant momentum is less effective as $\lambda$ increases, and the no momentum variant is more effective when $\lambda$ is small. Moreover, we observe the robustness of our heuristic increasing momentum since it performs well for all regularizers.

**Sketch size.** We tested different values of the sketch size, namely $\tau = 10\%, 50\%$ and $90\%$ of $m$, and reported the run time to reach a tolerance of $10^{-4}$ for each method. In Figure 4.5, we observe that constant momentum is very affected by the sketch size and is always the slowest method. For intermediate sketch sizes, like $\tau = m/2$, our increasing momentum competes with no-momentum. We also see that $\tau$ should not be set too small nor too large. Indeed, larger sketch sizes lead to better estimates of the initial system (4.4) by (4.9). But if the sketch size is too large, solving the sketched system (4.14) becomes very slow.

**Conclusions.** We highlighted that momentum sketch-and-project is more efficient for small regularizers $\lambda$ as opposed to the vanilla method. Also, we showed that the run time decreases then increases as a function of the sketch size $\tau$. Thus $\tau$ should be set to an intermediate value, *e.g.*, $\tau = m^{\frac{2}{3}}$, so that the cost of solving the sketched system (4.14) is manageable. Finally, the main conclusion of this experiment is the overall robustness (across values of $\lambda$ and $\tau$) and faster convergence of our heuristic increasing momentum setting.

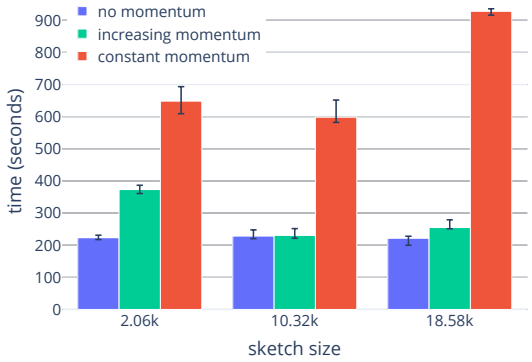
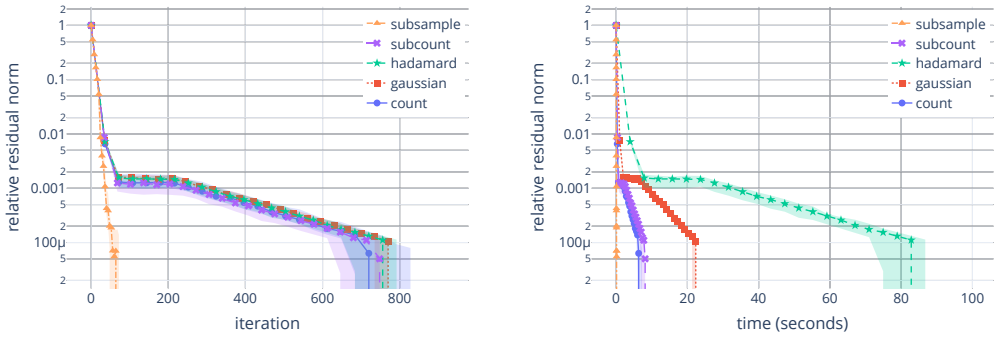



**Figure 4.5:** *Effect of sketch size on increasing momentum (green), constant momentum (red) and no-momentum (blue) (`California Housing` with kernel, $m = 20,640$, $\lambda = 10^{-6}$, `Subsample sketch`).*




**(a)** *Iterations* **(b)** *Time*

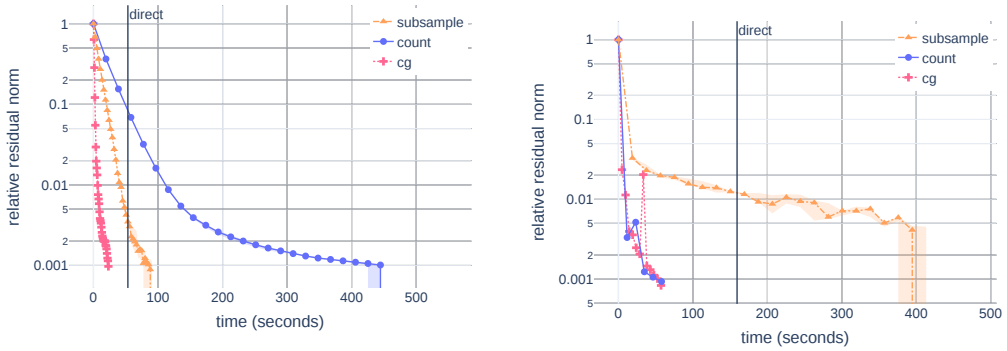
**Figure 4.6:** *Comparison of sketch methods for heuristic increasing momentum on kernel ridge regression problem applied to the `Boston` dataset ($\lambda = 10^{-6}$, $m = 506$ and $\tau = m/4 = 126$).*

### 4.8.2 Experiment 2: comparison of different types of sketches

In this experiment we compare the performance of different sketching methods presented in Section 4.4 when using our heuristic increasing momentum setting (4.67). In Figure 4.6, we monitor both the number of iterations and the time taken, since different sketching methods take different amounts of time per iteration. We see in this figure that there is a clear ranking between sketch methods in terms of run time:

1. `Subsample` is the most efficient on dense data (see also Figure 4.7a)
2. `Count` and `SubCount` are the most efficient on sparse data (see Figure 4.7b) and have very similar performance
3. `Gaussian` is slow because of the cost of dense matrix-matrix multiplications
4. `Hadamard` is extremely slow because of the size of the padded matrix and of the preprocessing time it requires

**Conclusions.** For dense datasets, the `Subsample sketch` is the fastest because it only requires slicing operations, which are very well optimized (especially for NumPy arrays). For sparse problems, the `Count sketch` is to be preferred since it *densifies* just enough sketched matrices to extract information out of $\mathbf{A}$. We find that computing `Gaussian` and `Hadamard sketch` is very time demanding. Furthermore, the cost associated to the padding step in `Hadamard sketch` is detrimental, especially for large $m$, which often makes it the slowest method.




**(a)** *Kernel ridge regression applied to* `California Housing` *($\tau = m/4 = 5,160$).*

**(b)** *Ridge regression applied to* `RCV1` *($\tau = \lceil m^{2/3} \rceil = 1,307$).*

**Figure 4.7:** *Time comparison of best* `RidgeSketch` *methods (heuristic increasing momentum) against direct and CG solvers ($\lambda = 10^{-9}$).*

### 4.8.3 Experiment 3: comparison against direct solver and conjugate gradients

We now compare `RidgeSketch` with our heuristic increasing momentum setting (4.67) with the two best sketches, `Subsample` and `Count sketch`, against a direct solver and *Conjugate Gradients* (CG) [64]. The direct solver we used was LAPACK's `gesv` routine [5] for solving positive definite linear systems. Here we tested our code on

- A kernel ridge regression problem (4.7) on the dataset `California Housing` ($m = n = 20,640$).
- A large and sparse dataset: `RCV1` ($m = d = 47,236$), with only 0.16% of non-zeros.

Figure 4.7 highlights a clear benefit of iterative methods like CG and sketch-and-project for solving large scale ridge problems. Moreover, this experiment on `RCV1` shows that in the large scale sparse setting, `Count sketch` is competitive as compared to CG.

## 4.9 Acceleration

Recently, it was shown that the convergence rate of sketch-and-project can be improved by using *acceleration* [150, 50]. See Algorithm 13 for our pseudo-code of the accelerated sketch-and-project method. In [50] it was shown that, by using specific parameter settings, the accelerated method enjoys a linear convergence with a rate that can be an order of magnitude better than rate given in Theorem 4.7.

Despite this strong theoretical advantage of the accelerated method, it is not clear if this translates into a practical advantage because 1) the additional overhead costs of the method may outweigh the benefits of the improved iteration complexity and 2) the accelerated method relies on knowing beforehand spectral properties of the matrix $\mathbf{A}$ that are expensive to compute. Here we show that 1) can be remedied by a careful implementation and 2) is indeed a fundamental issue that prevented us from developing a practical method.

### 4.9.1 From theory to practical implementation of acceleration

**Additional overhead: pseudo-code and efficient implementation.** The accelerated version in Algorithm 13 has three $(w^k, z^k, v^k)$ sequences of iterates. The bottleneck costs of Algorithm 13 are the same as the standard sketch-and-project method in Algorithm 11, which are the sketching operations on line 12. Indeed, the only additional computations in Algorithm 13 as compared to Algorithm 11 are lines 14 and 16 which cost $O(m)$. The other additional overhead is how to monitor

**Algorithm 13** Sketch-and-project method with acceleration

1: **Parameters:** distribution over random $m \times \tau$ matrices in $\mathcal{D}$, tolerance $\epsilon > 0$, acceleration parameters $\mu \in [0,\, 1]$, $\nu \in \left[1,\, \frac{1}{\mu}\right]$
2: Set $w^0 = v^0 = 0 \in \mathbb{R}^m$ ▷ weights initialization
3: Set $r_z^0 = r_v^0 = r_w^0 = \mathbf{A}w^0 - b \in \mathbb{R}^m$ ▷ residual initialization
4: Set $\beta = 1 - \sqrt{\frac{\mu}{\nu}}$
5: Set $\gamma = \sqrt{\frac{1}{\mu\nu}}$
6: Set $\alpha = \frac{1}{1+\sqrt{\frac{\nu}{\mu}}}$
7: $k = 0$
8: **While** $\left\|r_v^k\right\|_2 / \left\|r_v^0\right\|_2 \leq \epsilon$ **do**
9: Sample $\mathbf{S}_k \sim \mathcal{D}$ i.i.d
10: Compute and store $\mathbf{AS}_k$
11: $\delta^k = \texttt{least\_norm\_solution}\left(\mathbf{S}_k^\top \mathbf{AS}_k, \mathbf{S}_k^\top r_z^k\right)$ ▷ solve sketched system
12: $g^k = \mathbf{S}_k \delta^k$
13: $\mathbf{A}g^k = (\mathbf{AS}_k)\delta^k$
14: $z^k = \alpha v^k + (1-\alpha)w^k$ ▷ update the iterates
15: $w^{k+1} = z^k - g^k$
16: $v^{k+1} = \beta v^k + (1-\beta)z^k - \gamma g^k$
17: $r_z^k = \alpha r_v^k + (1-\alpha)r_w^k$ ▷ update the residuals
18: $r_w^{k+1} = r_z^k - \mathbf{A}g^k$
19: $r_v^{k+1} = \beta r_v^k + (1-\beta)r_z^k - \gamma \mathbf{A}g^k$
20: $k = k + 1$
21: **Output:** $w^t$ ▷ return weights vector

the residual so as to know when to stop the algorithm. We found that for the residual to be efficiently maintained and updated, we had to monitor three residual vectors $r_z^k \overset{\text{def}}{=} \mathbf{A}z^k - b$, $r_w^k \overset{\text{def}}{=} \mathbf{A}w^k - b$ and $r_v \overset{\text{def}}{=} \mathbf{A}v^k - b$. These residual vectors can be updated efficiently since

$$r_w^{k+1} = \mathbf{A}w^{k+1} - b = \mathbf{A}(z^k - g^k) - b = r_z^k - \mathbf{A}g^k = r_z^k - \mathbf{AS}_k\delta^k \ .$$

Since we have already pre-computed $\mathbf{AS}_k$ and $\delta^k$, the additional cost is $O(m\tau)$. Furthermore from lines 14 and 16 we have that

$$r_z^{k+1} = \alpha\mathbf{A}v^k + (1-\alpha)\mathbf{A}w^k - b = \alpha r_v^k + (1-\alpha)r_w^k \ ,$$

and

$$r_v^{k+1} = \beta\mathbf{A}v^k + (1-\beta)\mathbf{A}z^k - b - \gamma\mathbf{A}g^k = \beta r_v^k + (1-\beta)r_z^k - \gamma\mathbf{AS}_k\delta^k \ .$$

Thus the residuals $r_v^k$ and $r_v^k$ can be updated at an additional $O(m)$ cost to perform the above vector additions and scalar multiplications.

**Setting the acceleration parameters with spectral properties.** The main issue with the accelerated version is that it introduces two new hyperparameters $\mu$ and $\nu$ which have to be estimated. In theory [50], by setting these two parameters according to

$$\mu \overset{\text{def}}{=} \inf_{x \in \text{Range}(\mathbf{A}^\top)} \frac{\langle \mathbb{E}\left[\mathbf{Z}\right] x, x\rangle}{\langle x, x\rangle} \qquad \text{and} \qquad \nu \overset{\text{def}}{=} \sup_{x \in \text{Range}(\mathbf{A}^\top)} \frac{\left\langle \mathbb{E}\left[\mathbf{Z}\mathbb{E}\left[\mathbf{Z}\right]^\dagger \mathbf{Z}\right] x, x\right\rangle}{\langle \mathbb{E}\left[\mathbf{Z}\right] x, x\rangle} \ . \tag{4.68}$$

where

$$\mathbf{Z} \overset{\text{def}}{=} \mathbf{A}^\top \mathbf{S}^\top (\mathbf{S}^\top \mathbf{A} \mathbf{S})^\dagger \mathbf{S}^\top \mathbf{A} \ , \tag{4.69}$$

we can guarantee an accelerated rate of convergence. The issue is that the theory in [50] requires that these parameters be set exactly using (4.68) and computing (4.68) is more costly then solving the original linear system! So this leads us to the following practical question.

> Is there a **rule of thumb setting** for the acceleration parameters $\mu$ and $\nu$ such that the accelerated sketch-and-project method is consistently better than the standard sketch-and-project method?

In [150] the authors propose some settings for $\nu$ and $\mu$ when the sketch size is large. But there is currently no practical rule for setting these parameters in general. In theory, we know that

$$0 \ \leq \ \mu \ \leq \ \frac{1}{\nu} \ \leq \ 1 \ ,$$

as proven in Lemma 2 in [50]. Furthermore the extreme case where $\mu = \nu = 1$ corresponds to the standard sketch-and-project method, as can be seen by induction on Algorithm 13 since $z^k = v^k = w^k$ for all iterations, and $w^k$'s are thus equivalent to the $w^k$'s in Algorithm 11. We now look at some other extreme cases to better understand these parameters.

**Single row sampling.** For this special case of subsample sketchs with $\tau = 1$, that is $\mathbf{S} = e_i$, where we recall that $(e_i)_{1 \leq i \leq m}$ are the canonical basis vectors of $\mathbb{R}^m$, with probability $\frac{1}{m}$ we know that

$$\mu = \frac{\lambda_{\min}(\mathbf{A})}{\text{Trace}\,(\mathbf{A})} \qquad \text{and} \qquad \nu = \frac{\text{Trace}\,(\mathbf{A})}{\min_{i=1,\ldots,m} \mathbf{A}_{ii}} \ . \tag{4.70}$$

Consequently, if the eigenvalues of $\mathbf{A}$ are concentrated with $\lambda_{\min}(\mathbf{A})$ close to $\lambda_{\max}(\mathbf{A})$ then we have that $\mu \approx \frac{1}{m}$ and $\nu \approx m$. Alternatively, if the eigenvalues of $\mathbf{A}$ are far apart, then it may be that $\mu \approx 0$ and $\nu \approx \infty$.

**No sketching.** When $\mathbf{S} = \mathbf{I}$ then $\mathbf{Z} = \mathbf{A}$ since $\mathbf{A}$ is invertible. Consequently

$$\mu = \inf_{x \in \text{Range}(\mathbf{A}^\top)} \frac{\langle \mathbf{A}x, x \rangle}{\langle x, x \rangle} = \lambda_{\min}(\mathbf{A}), \qquad \nu = \sup_{x \in \text{Range}(\mathbf{A}^\top)} \frac{\langle \mathbb{E}\left[\mathbf{A}\right] x, x \rangle}{\langle \mathbb{E}\left[\mathbf{A}\right] x, x \rangle} = 1 \ . \tag{4.71}$$

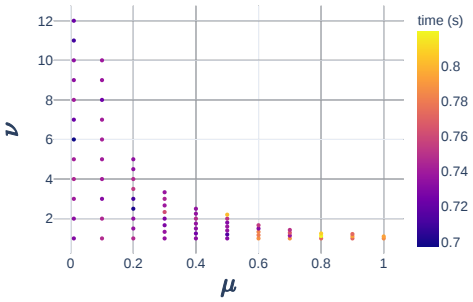
In either of these two extremes, we need the smallest eigenvalue of $\mathbf{A}$ to set $\mu$ and $\nu$, which is a prohibitive cost. In Section 4.9.2 we show that finding a setting for $\mu$ and $\nu$ that outperforms the standard sketch-and-project method is difficult, and akin to finding a needle in a haystack.

### 4.9.2 Experiments setting the acceleration parameters

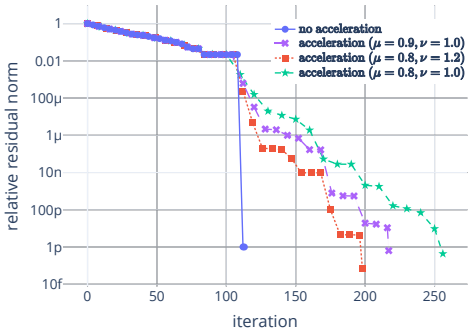
Here we would like to verify if there exists a default setting for the acceleration parameters $\mu$ and $\nu$ that results in consistently faster execution than the non-accelerated version. In Figure 4.8a, we show the results of an extensive grid search for trying to identify a suitable $\mu$ and $\nu$ setting. This figure shows the time taken to reach a $\epsilon = 10^{-4}$ solution of the relative residual for different pairs of $\mu$ and $\nu$ such that

$$0 \ \leq \ \mu \ \leq \ \frac{1}{\nu} \ \leq \ 1 \ .$$

The problem we considered here is kernel ridge regression, with a RBF kernel with $\sigma = 0.5$. The data $\mathbf{X}$ is a random $n \times d$ sparse CSC matrix with density 0.25, and the regularizer is set to $\lambda = 1/n$. From Figure 4.8a, there is no clear pair of parameters $(\mu, \nu)$ leading to an improvement in convergence. Even if a finer grid search might allow to find optimal parameters, the gain in convergence is so marginal that it makes acceleration impractical compared to the version without acceleration, see Figure 4.8b.




**(a)** *Average time taken over* 100 *runs, for different acceleration parameters, to solve the problem with tolerance* $10^{-4}$.




**(b)** *Comparison of no-acceleration and different acceleration parameter settings.*

**Figure 4.8:** *Grid time plot (left) and single run (right) to solve a RBF kernel regression* ($\sigma = 0.5$, $\lambda = 1/n$) *for random sparse CSC matrix* $\mathbf{X} \in \mathbb{R}^{n \times d}$ ($n = m = 2000$, $d = 500$) *with density* 0.25. *We used a* `Subsample sketch` *method and the sketch size is set to* $\tau = \lfloor m^{2/3} \rfloor = 158$.

We conclude that there is yet no known way to set these acceleration parameters in practice: the theory might be too loose to set them and looking empirically on a grid search for optimal points is too cumbersome.

## 4.10 `RidgeSketch` package

Our `RidgeSketch` Python package is designed to be easily augmented by new contributions. Users are encouraged to add new sketches, new parameter settings (*e.g.,* for momentum) and new datasets (see `datasets/data_loaders.py`). They can also easily compare methods using the command:

```
$ python benchmarks.py [options] [name of config]
```

`RidgeSketch` comes with two tutorial Jupyter Notebooks: one for fitting a `RidgeSketch` model, and another for adding new sketches and benchmarks. Next we present some snippets of code.

### 4.10.1 Solving the ridge regression problem

In Code 4.1, we provide an example creating a ridge regression model and solving the fitting problem with our `Subsample sketch` solver.

### 4.10.2 Adding sketching methods

We built our package so that new sketching methods can easily be added by contributors. To do this, one should open the `sketching.py` file and create a new sketching subclass that inherits from the `Sketch` class. In Code 4.2 we provide the example of the `Subsample` sketching method.

```python
from datasets.data_loaders import BostonDataset
from ridge_sketch import RidgeSketch

dataset = BostonDataset()
X, y = dataset.load_X_y()  # data loader

model = RidgeSketch(
    alpha=1.0,  # regularizer
    algo_mode="mom",  # momentum version
    solver="subsample",  # sketch, cg, direct or sklearn solver
    sketch_size=10)
model.fit(X, y)  # solve the fitting problem
```

**Code 4.1:** *Example of how to build a* `RidgeSketch` *model and fit it to a dataset.*

```python
import numpy as np

class SubsampleSketch(Sketch):
    def __init__(self, A, b, sketch_size):
        """Initializes the Sketch object and defines its attributes"""
        super().__init__(A, b, sketch_size)
        self.sample_indices = None

    def set_sketch(self):
        """Generates subsampling indices representing the sketching matrix"""
        self.sample_indices = np.random.choice(
            self.m, size=self.sketch_size, replace=False).tolist()

    def sketch(self, r):
        """Generates sketched system"""
        self.set_sketch()
        SA = self.A.get_rows(self.sample_indices)
        SAS = SA[:, self.sample_indices]
        rs = r[self.sample_indices]
        return SA, SAS, rs

    def update_iterate(self, w, lmbda, step_size=1.0):
        """Updates the weights by solving the sketched system"""
        w[self.sample_indices] -= step_size * lmbda
        return w
```

**Code 4.2:** *Implementation of the* `Subsample sketch` *subclass.*

CHAPTER 5

# Randomized ADI methods

لو أني أعرف أن الحب خطيرٌ جدًا ما أحببت
لو أني أعرف أن البحر عميقٌ جدًا ما أبحرت
لو أني أعرف خاتمتي ما كنت بدأت

عبد الحليم حافظ – رسالة من تحت الماء

## 5.1 Introduction

The Alternating-Direction Implicit (ADI) methods are iterative methods solving linear systems of the form

$$(\mathbf{H} + \mathbf{V})u = s \ , \tag{5.1}$$

where $\mathbf{H}, \mathbf{V} \in \mathbb{R}^{d\times d}$ are symmetric matrices.

A naive way of solving such a problem would be to sum together $\mathbf{H}$ and $\mathbf{V}$ and then use a classical method such as a direct solver or Conjugate Gradients. Yet, in some applications the full matrix $\mathbf{\Sigma} \stackrel{\text{def}}{=} \mathbf{H} + \mathbf{V}$ is too difficult to invert directly [89, 41]. In such cases *operator splitting methods* [40], including the PR method, are preferred. They rely on a fundamental assumption: $(\mathbf{\Sigma} + r\mathbf{I}_m)$ is not easily invertible, for $r \in \mathbb{R}$, which precludes the application of the *proximal point algorithm* [128] Nevertheless, let us suppose that $(\mathbf{H} + p\mathbf{I}_m)$ and $(\mathbf{V} + q\mathbf{I}_m)$ are relatively easy to invert, for some $p, q \in \mathbb{R}$. Then, by combining cleverly *forward* steps, *i.e.,* apply $(\mathbf{H} + p\mathbf{I}_m)$ or $(\mathbf{V} + q\mathbf{I}_m)$, and *backward* steps, *i.e.,* apply $(\mathbf{H} + p\mathbf{I}_m)^{-1}$ or $(\mathbf{V} + q\mathbf{I}_m)^{-1}$, one will approach a direct application of $(\mathbf{\Sigma} + r\mathbf{I}_m)^{-1}$.

In other applications, such as the resolution of Sylvester matrix equation introduced below in (5.7), the problem is not originally formulated as a linear systems. As showed later in Lemma 5.1, it can be rewritten such as in (5.1) at the expense of costly computations and of a huge dimension increase. Hence $\mathbf{H}$ and $\mathbf{V}$ cannot simply be summed together.

### 5.1.1 The Peaceman-Rachford method

**Origins of ADI methods.** In 1955, the ADI method was introduced by Peaceman and Rachford [113] and numerically implemented by Douglas and Rachford [36] to solve the 2D heat flow equation, which is a discretized parabolic Partial Differential Equation (PDE) [151]. The *Peaceman-Rachford (PR) method* solving (5.1) starts with an initial vector (usually full of zeros) $u^0 \in \mathbb{R}^n$ and performs the following two-step iterations:

- Solve for $u^{k+1/2}$

$$(\mathbf{H} + p_k\mathbf{I}_d)u^{k+1/2} = s + (p_k\mathbf{I}_d - \mathbf{V})u^k \ . \tag{5.2}$$

- Solve for $u^{k+1}$

$$(\mathbf{V} + q_k \mathbf{I}_d)u^{k+1} = s + (q_k \mathbf{I}_d - \mathbf{H})u^{k+1/2} \ . \tag{5.3}$$

The sequences of real numbers $(p_k)_k$ and $(q_k)_k$ are carefully selected algorithmic parameters, usually called *shift parameters*. The ADI method is a particularly efficient solver for (5.1) if the linear systems involving $\mathbf{H} - p_k \mathbf{I}_d$ and $\mathbf{V} - q_k \mathbf{I}_d$ can be rapidly solved and the spectrum of $\mathbf{H}$ and $\mathbf{V}$ are well-separated. When the PR method is run with constant shifts, it is often referred to as Smith's method [143].

**Deriving the Peaceman-Rachford method.** To comprehend the *Peaceman-Rachford method (PR) method*, we derive it again in a principled way. Let $\mathbf{H}, \mathbf{V} \in \mathbb{R}^{d \times d}$. For this, we split the variables $u = u_{\mathbf{H}} = u_{\mathbf{V}}$ and add the constraint $u_{\mathbf{H}} = u_{\mathbf{V}}$ so that solving our original problem (5.1) is equivalent to solving in $u_{\mathbf{H}}$ and $u_{\mathbf{V}} \in \mathbb{R}^d$

$$\begin{cases} \mathbf{H}u_{\mathbf{H}} + \mathbf{V}u_{\mathbf{V}} = s \\ u_{\mathbf{H}} = u_{\mathbf{V}} \end{cases} \tag{5.4}$$

Let $p, q \in \mathbb{R}$. Adding $pu_{\mathbf{H}} = pu_{\mathbf{V}}$ to both sides of (5.4) gives

$$\begin{cases} (\mathbf{H} + p\mathbf{I}_d)u_{\mathbf{H}} + \mathbf{V}u_{\mathbf{V}} = s + pu_{\mathbf{V}} \\ u_{\mathbf{H}} = u_{\mathbf{V}} \end{cases} \tag{5.5}$$

Similarly, Adding $qu_{\mathbf{H}} = qu_{\mathbf{V}}$ to both sides of (5.4) gives

$$\begin{cases} \mathbf{H}u_{\mathbf{H}} + (\mathbf{V} + q\mathbf{I}_d)u_{\mathbf{V}} = s + qu_{\mathbf{H}} \\ u_{\mathbf{H}} = u_{\mathbf{V}} \end{cases} \tag{5.6}$$

These two equivalent formulations (5.5) and (5.6) suggest a method. By alternating between solving (5.5) in $u_{\mathbf{H}}$, $u_{\mathbf{H}} = u_{\mathbf{V}}$ and (5.6) in $u_{\mathbf{V}}$ we have the resulting iterative method.

$$\begin{aligned} \text{Solve for } u_{\mathbf{H}}^{k+1} \quad &: \quad (\mathbf{H} + p_k \mathbf{I}_d)u_{\mathbf{H}}^{k+1} = s + (p_k \mathbf{I}_d - \mathbf{V})u_{\mathbf{V}}^{k} \\ \text{Set} \quad &: \quad u_{\mathbf{V}}^{k} \leftarrow u_{\mathbf{H}}^{k+1} \\ \text{Solve for } u_{\mathbf{V}}^{k+1} \quad &: \quad (\mathbf{V} + q_k \mathbf{I}_d)u_{\mathbf{V}}^{k+1} = s + (q_k \mathbf{I}_d - \mathbf{H})u_{\mathbf{H}}^{k+1} \end{aligned}$$

where $p_k, q_k \in \mathbb{R}$. Which can be simplified to

$$\begin{aligned} \text{Solve for } u^{k+1/2} \quad &: \quad (\mathbf{H} + p_k \mathbf{I}_d)u^{k+1/2} = s + (p_k \mathbf{I}_d - \mathbf{V})u^{k} \\ \text{Solve for } u^{k+1} \quad &: \quad (\mathbf{V} + q_k \mathbf{I}_d)u^{k+1} = s + (q_k \mathbf{I}_d - \mathbf{H})u^{k+1/2} \end{aligned}$$

and leads us back the PR updates from (5.2)–(5.3).

### 5.1.2 ADI methods for matrix equations

The PR method has been generalized to three dimensions [36, 23] and the stability and the convergence of the have has been proved. To find optimal parameters, the convergence analysis of ADI methods led to a Chebyshev minimax problem. Lately, it was found that this problem had been solved by Zolotarev in 1877 [168]. The early convergence theory of ADI is presented in the survey published in 1962 by Birkhoff, Varga and Young [18] and in [159]. We also refer to the excellent book of Wachspress [156] summarizing 50 years of ADI theory and applications.

**Link with Sylvester matrix equations.** Almost 30 years after the introduction of the Peaceman-Rachford method, it was showed that ADI methods were applicable to another class of problems : Sylvester matrix equations. Sylvester equations are matrix equations of the form

$$\mathbf{AX} - \mathbf{XB} = \mathbf{F} \ , \tag{5.7}$$

where $\mathbf{X} \in \mathbb{R}^{n\times n}$ is the unknown and $\mathbf{A}, \mathbf{B}$ and $\mathbf{F} \in \mathbb{R}^{n\times n}$. These equations can easily be extended to rectangular matrices, however, in this chapter we focus on square matrices of the same size for the sake of simplicity and clarity.

When $\mathbf{B} = -\mathbf{A}^\top$, we call (5.7) a Lyapunov equation. Next lemma shows how to transform (5.7) into (5.1) and gives an example where the linear operators $\mathbf{H}$ and $\mathbf{V}$ are only accessible separately, respectively through $\mathbf{A}$ and $\mathbf{B}$.

**Lemma 5.1** (Sylvester equation as an ADI problem)**.** *Sylvester equation* (5.7) *can be rewritten as a Peaceman-Rachford problem* (5.1) *with*

$$\mathbf{H} = \mathbf{I}_n \otimes \mathbf{A} \in \mathbb{R}^{n^2\times n^2} \quad \textit{and} \quad \mathbf{V} = -\mathbf{B}^\top \otimes \mathbf{I}_n \in \mathbb{R}^{n^2\times n^2} \ , \tag{5.8}$$

*where* $\otimes$ *denotes the Kronecker product, and*

$$s = vec\,(\mathbf{F}) = \begin{bmatrix} \mathbf{F}_{:1} \\ \vdots \\ \mathbf{F}_{:n} \end{bmatrix} \in \mathbb{R}^{n^2} \quad \textit{and} \quad u = vec\,(\mathbf{X}) = \begin{bmatrix} \mathbf{X}_{:1} \\ \vdots \\ \mathbf{X}_{:n} \end{bmatrix} \in \mathbb{R}^{n^2} \ , \tag{5.9}$$

*where* $vec\,(\cdot)$ *denotes the vectorization operator which stacks vertically the columns of a given matrix.*

Such matrix equations are ubiquitous in applied mathematics, both in theory and in practice [15, 17, 142]. Thus, this link between Sylvester equations and ADI methods revived the interest the field.

**ADI method for Sylvester equations.** One could solve (5.7) by, first, building matrices $\mathbf{H}$ and $\mathbf{V}$ and the vector $s$ according to (5.8) and (5.9) and then, by apply Peaceman-Rachford method presented in (5.2)–(5.3). Yet, when the dimension $n$ is very large, the building phase becomes cumbersome since it requires computing and storing $n^2 \times n^2$ matrices and the half-steps require inverting $n^2 \times n^2$ linear systems. Instead, one should apply the equivalent method for Sylvester equations[1]. The *ADI method* for Sylvester equations [92] solving (5.7) starts from an initial matrix (usually full of zeros) $\mathbf{X}^{(0)} \in \mathbb{R}^{n\times n}$ and performs the following two-step iterations:

- Solve for $\mathbf{X}^{(k+1/2)}$
$$(\mathbf{A} - p_k\mathbf{I}_n)\mathbf{X}^{(k+1/2)} = \mathbf{X}^{(k)}(\mathbf{B} - p_k\mathbf{I}_n) + \mathbf{F} \ . \tag{5.10}$$
- Solve for $\mathbf{X}^{(k+1)}$
$$\mathbf{X}^{(k+1)}(\mathbf{B} - q_k\mathbf{I}_n) = (\mathbf{A} - q_k\mathbf{I}_n)\mathbf{X}^{(k+1/2)} - \mathbf{F} \ . \tag{5.11}$$

This ADI method is more practical than the one described above since it only requires to store $n \times n$ matrices and to invert $n \times n$ linear systems. The performance of this algorithm highly depends on the *shift parameters* $p_k$, $q_k$ selected at the beginning of each iteration. When precise spectral information is available on $\mathbf{A}$ and $\mathbf{B}$, efficient shifts can lead to fast convergence, see Section 2.2 of [131]. Such details we be recalled in our analysis.

**Discretized Poisson equation.** As we said above, solving such equations is a common problem in many applied mathematics problems such as in control theory [25], signal processing [65] or in the discretization of PDEs. Here we give an example of PDE discretization that will be tackled in our numerical experiments in Section 5.5. Consider the 2D Poisson's equation in $u$ with zero homogeneous Dirichlet conditions:

$$\begin{aligned} &\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u(x,y) = f(x,y), \quad \forall (x,y) \in [-1,1]^2 \\ &u(x,y) = 0, \quad \forall x, y : |x| = 1 \vee |y| = 1 \end{aligned}$$

[1] More precisely, both methods are equivalent if $p_k$ is replaced by $-p_k$ and if the second half-step system is multiplied on both sides by $-1$.

where $f$ is a known continuous function. The discretized version is the Lyapunov equation below:

$$\mathbf{D}_2\mathbf{X} + \mathbf{X}\mathbf{D}_2^T = \mathbf{F} \ . \tag{5.12}$$

where $\mathbf{D}_2 \in \mathbb{R}^{n\times n}$ is a second-order discrete differentiation matrix, $\mathbf{F} \in \mathbb{R}^{n\times n}$ is the known function applied at the grid points and $\mathbf{X} \in \mathbb{R}^{n\times n}$ is the unknown. Using finite differences to form this problem, $\mathbf{D}_2$ becomes tridiagonal but it can be dense if other differentiation methods are used, for instance a Chebyshev differentiation matrix.

The standard approach to solve (5.12) is the Bartels-Stewart (BS) algorithm [12, 46] when the dimensions are moderate. Yet, this direct method has a computational cost of $O(n^3)$ (except if matrix-vector products can be computed efficiently) which is prohibitive for large systems. One could also try applying naively an iterative method like Conjugate Gradients (CG) [64] without exploiting the Kronecker structure to solve the equivalent formulation of (5.12):

$$(\mathbf{D}_2 \otimes \mathbf{I}_n + \mathbf{I}_n \otimes \mathbf{D}_2)\text{vec}\,(\mathbf{X}) = \text{vec}\,(\mathbf{F}) \ .$$

But once again, this method would cost $O(n^4)$ which is impractical. This is why, in large dimension solving Sylvester equations is challenging and usually iterative methods like ADI are preferred. In some applications where $\mathbf{F}$ admits a low-rank factorization, a factored ADI (fADI) [15] method can be efficient in such a case since $\mathbf{D}_2$ is symmetric positive definite (SPD) and tridiagonal, thus very sparse. However, if $\mathbf{D}_2$ is dense and if the dimension $n$ is large, *e.g.,* when spatial discretization step of the PDE is small, traditional ADI methods also suffer from the space needed in RAM to store the matrices and cannot apply efficiently direct solver to invert the linear systems in the half-steps (5.10)–(5.11). In such cases, it is natural to wonder if randomization of the ADI method can be beneficial.

**Randomization of the ADI method.** In what follows, we consider the very general case of dense matrices and do not assume any particular structure on the matrices $\mathbf{A}$, $\mathbf{B}$ or $\mathbf{F}$. As exposed before, when $n$ is very large, performing the linear systems inversions becomes intractable. Recent works on randomization of Kaczmarz or related methods [144, 104, 105, 59] show their efficiency for inverting matrices or solving large linear systems. Indeed, randomized iterative methods have many advantages over deterministic methods, especially practical aspects. They are often easy to implement, require less memory, perform cheaper iterations and are sometimes easier to parallelize.

**Contributions.** The goal of this last chapter is to develop and analyse a randomized version of the ADI method to solve large scale Peaceman-Rachford problems and Sylvester matrix equations.

In Section 5.2, we apply the *sketch-and-project* method developed in [59, 60] to solve each of the linear systems in the half-steps of the PR method and present the resultant algorithms. In Section 5.4, we specialize these sketched updates to the ADI method solving the Sylvester equation. Our new methods are respectively called the *Sketched Peaceman-Rachford (SPR) method* and the *Sketched ADI (SADI) method*.

In Section 5.3, we analyse the convergence of the SPR algorithm in the particular case of single row sketching.

Finally, in Section 5.5 we compare our randomized ADI methods against deterministic ones. We show both on artificial and real problems the numerical efficiency of our methods on both PR problems and Sylvester equations.

## 5.2 Sketched Peaceman-Rachford method

As discussed in the introduction, in large scale settings, it might be advantageous to only solve a sketch of (5.2) and (5.3). Indeed, solving a rough approximation of the systems (5.2) and (5.3) can be much faster than solving the entire systems while still converging in expectation. Thus, we try to apply the sketch-and-project method [59] to each of the half-step of the PR method.

### 5.2.1 A randomized Peaceman-Rachford method

**Sketched Peaceman-Rachford method.** In this section, we describe our new proposed method in a very general version[2], that we call *sketched Peaceman-Rachford (SPR) method*. Assuming that $\mathbf{H}+\mathbf{V}$ is SPD, let us denote $u^*$ the solution to the Peaceman-Rachford problem (5.1). Let $\mathbf{M_H}, \mathbf{M_V} \in \mathbb{R}^{d\times d}$ be two SPD matrices, let $\tau \in \mathbb{N}$ be the sketch size and let $\mathcal{D}$ be a distribution over matrices $\mathbb{R}^{d\times\tau}$. At the $k$–th iteration, let us sample two sketching matrices $\mathbf{S}_{1/2}, \mathbf{S}_1 \sim \mathcal{D}$ and apply the following updates

$$\begin{aligned} u^{k+1/2} &= \arg\min_u \left\|u-u^k\right\|_{\mathbf{M_H}} \\ &\quad \text{subject to} \quad \mathbf{S}_{1/2}^\top(\mathbf{H}+p_k\mathbf{I}_d)u = \mathbf{S}_{1/2}^\top s + \mathbf{S}_{1/2}^\top(p_k\mathbf{I}_d-\mathbf{V})u^k \ , \qquad (5.13)\\ u^{k+1} &= \arg\min_u \left\|u-u^{k+1/2}\right\|_{\mathbf{M_V}} \\ &\quad \text{subject to} \quad \mathbf{S}_1^\top(\mathbf{V}+q_k\mathbf{I}_d)u = \mathbf{S}_1^\top s + \mathbf{S}_1^\top(q_k\mathbf{I}_d-\mathbf{H})u^{j+1/2} \ . \qquad (5.14)\end{aligned}$$

In general, $\mathbf{M_H}$ should be chosen as a SPD matrix such that $(\mathbf{H}+p_k\mathbf{I}_d)\mathbf{M_H}^{1/2}$ is well conditioned. This suggests that the *metric matrices* $\mathbf{M_H}$ and $\mathbf{M_V}$ should change at each iteration. In particular if $\mathbf{H}+p_k\mathbf{I}_d$ (resp. $\mathbf{V}+q_k\mathbf{I}_d$) is SPD then choosing $\mathbf{M_H} \approx (\mathbf{H}+p_k\mathbf{I}_d)^{-1}$ (resp. $\mathbf{M_V} \approx (\mathbf{V}+q_k\mathbf{I}_d)^{-1}$) will result in a more efficient method (both in terms of theory and practice).

The closed form solution to (5.13) is given by

$$\begin{aligned} u^{k+1/2} &= u^k - \mathbf{M_H}^{-1}(\mathbf{H}+p_k\mathbf{I}_d)^\top\mathbf{S}_{1/2}\left(\mathbf{S}_{1/2}^\top(\mathbf{H}+p_k\mathbf{I}_d)\mathbf{M_H}^{-1}(\mathbf{H}+p_k\mathbf{I}_d)^\top\mathbf{S}_{1/2}\right)^\dagger\mathbf{S}_{1/2}^\top(\mathbf{H}u^k+\mathbf{V}u^k-s)\\ &= u^k - \mathbf{M_H}^{-1}\mathbf{Z_H}(\mathbf{H}+p_k\mathbf{I}_d)^{-1}R(u^k) \ , \qquad (5.15)\end{aligned}$$

and the solution to (5.14) is given by

$$\begin{aligned} u^{k+1} &= u^{k+1/2} - \mathbf{M_V}^{-1}(\mathbf{V}+q_k\mathbf{I}_d)^\top\mathbf{S}_1\left(\mathbf{S}_1^\top(\mathbf{V}+q_k\mathbf{I}_d)\mathbf{M_V}^{-1}(\mathbf{V}+q_k\mathbf{I}_d)^\top\mathbf{S}_1\right)^\dagger\mathbf{S}_1^\top(\mathbf{H}u^{k+1/2}+\mathbf{V}u^{k+1/2}-s)\\ &= u^{k+1/2} - \mathbf{M_V}^{-1}\mathbf{Z_V}(\mathbf{V}+q_k\mathbf{I}_d)^{-1}R(u^{k+1/2}) \ , \qquad (5.16)\end{aligned}$$

where

$$\mathbf{Z_H} \overset{\text{def}}{=} (\mathbf{H}+p_k\mathbf{I}_d)^\top\mathbf{S}_{1/2}\left(\mathbf{S}_{1/2}^\top(\mathbf{H}+p_k\mathbf{I}_d)\mathbf{M_H}^{-1}(\mathbf{H}+p_k\mathbf{I}_d)^\top\mathbf{S}_{1/2}\right)^\dagger\mathbf{S}_{1/2}^\top(\mathbf{H}+p_k\mathbf{I}_d) \ , \qquad (5.17)$$

$$\mathbf{Z_V} \overset{\text{def}}{=} (\mathbf{V}+q_k\mathbf{I}_d)^\top\mathbf{S}_1\left(\mathbf{S}_1^\top(\mathbf{V}+q_k\mathbf{I}_d)\mathbf{M_V}^{-1}(\mathbf{V}+q_k\mathbf{I}_d)^\top\mathbf{S}_1\right)^\dagger\mathbf{S}_1^\top(\mathbf{V}+q_k\mathbf{I}_d) \ , \qquad (5.18)$$

and where the *residual* is given by

$$R(u) \overset{\text{def}}{=} (\mathbf{H}+\mathbf{V})u - s = (\mathbf{H}+\mathbf{V})(u-u^*) \ . \qquad (5.19)$$

**Remark 5.2.** *Note that matrices $\mathbf{Z_H}$ and $\mathbf{Z_V}$ can depend on the shift parameters if they are not constant.*

The implementation of the SPR method is given in details in Algorithm 14.

**Update of the residual.** Exactly like for *RidgeSketch* algorithm in Section 4.3 of Chapter 4, one observes that the SPR method builds its half-iterates in (5.15)–(5.16) using the *residual* at the current iterate: $r^k \overset{\text{def}}{=} R(u^k) = (\mathbf{H}+\mathbf{V})u^k - s$. The latter can be computed efficiently if it is properly updated after each half-step. For instance, after the first half-step,

$$r^{k+1/2} = R(u^{k+1/2}) = (\mathbf{H}+\mathbf{V})u^{k+1/2} - s \overset{(5.15)}{=} R(u^k) - (\mathbf{H}+\mathbf{V})\mathbf{M_H}^{-1}(\mathbf{H}+p_k\mathbf{I}_d)^\top\mathbf{S}_{1/2}y^k \qquad (5.20)$$

[2]With general sketching and weight matrices. In next sections, the latter are dropped for the sake of clarity.

**Algorithm 14** Sketched Peaceman-Rachford (SPR)

**Input:** shift parameters $(p_k)_k, (q_k)_k$, weight matrices $\mathbf{M_H}, \mathbf{M_V}$, sketching distribution $\mathcal{D}$

**Initialize:** $u^0 = 0_d$

**for** $k = 0, 1, \ldots, K-1$ **do**

  Sample $\mathbf{S}_{1/2}, \mathbf{S}_1 \sim \mathcal{D}$ ▷ sketching matrices

  $r_{\mathbf{S}}^k = \mathbf{S}_{1/2}^\top R(u^k)$ ▷ 1st sketched residual

  $y^k = \texttt{least\_norm\_solution}\left(\left(\mathbf{S}_{1/2}^\top(\mathbf{H}+p_k\mathbf{I}_d)\mathbf{M}_{\mathbf{H}}^{-1}(\mathbf{H}+p_k\mathbf{I}_d)^\top\mathbf{S}_{1/2}\right), r_{\mathbf{S}}^k\right)$

  $u^{k+1/2} = u^k - \mathbf{M}_{\mathbf{H}}^{-1}(\mathbf{H}+p_k\mathbf{I}_d)^\top\mathbf{S}_{1/2}y^k$ ▷ 1st half-step

  $r_{\mathbf{S}}^{k+1/2} = \mathbf{S}_1^\top R(u^{k+1/2})$ ▷ 2nd sketched residual

  $y^{k+1/2} = \texttt{least\_norm\_solution}\left(\left(\mathbf{S}_1^\top(\mathbf{V}+q_k\mathbf{I}_d)\mathbf{M}_{\mathbf{V}}^{-1}(\mathbf{V}+q_k\mathbf{I}_d)^\top\mathbf{S}_1\right), r_{\mathbf{S}}^{k+1/2}\right)$

  $u^{k+1} = u^{k+1/2} - \mathbf{M}_{\mathbf{V}}^{-1}(\mathbf{V}+q_k\mathbf{I}_d)^\top\mathbf{S}_1y^{k+1/2}$ ▷ 2nd half-step

**Output:** $u^K$

where $y^k \overset{\text{def}}{=} \left(\mathbf{S}_{1/2}^\top(\mathbf{H}+p_k\mathbf{I}_d)\mathbf{M}_{\mathbf{H}}^{-1}(\mathbf{H}+p_k\mathbf{I}_d)^\top\mathbf{S}_{1/2}\right)^\dagger r_{\mathbf{S}}^k$ and $r_{\mathbf{S}}^k \overset{\text{def}}{=} \mathbf{S}_{1/2}^\top R(u^k)$ denotes the sketch residual. Moreover, $y^k$ can be computed as the least-norm solution of the following linear system in $x$

$$\mathbf{S}_{1/2}^\top(\mathbf{H}+p_k\mathbf{I}_d)\mathbf{M}_{\mathbf{H}}^{-1}(\mathbf{H}+p_k\mathbf{I}_d)^\top\mathbf{S}_{1/2}x = r_{\mathbf{S}}^k \ . \tag{5.21}$$

**Remark 5.3** (Weight matrices and the ADI structure)**.** *Let us assume that $\mathbf{H}+\mathbf{V}$ is a SPD matrix. One can simplify the residual update* (5.20) *by setting $\mathbf{M_H} = \mathbf{H}+\mathbf{V}$ which gives*

$$r^{k+1/2} = r^k - (\mathbf{H}+p_k\mathbf{I}_d)^\top\mathbf{S}_{1/2}y^k \ .$$

*But it does not lead to the same simplifications as the one in Chapter 4, where the weight matrix was set to $\mathbf{W} = \mathbf{A}^{-1}$. Indeed, the first update* (5.15) *becomes*

$$u^{k+1/2} = u^k - \underbrace{(\mathbf{H}+\mathbf{V})^{-1}(\mathbf{H}+p_k\mathbf{I}_d)^\top}_{\text{no cancellation}}\mathbf{S}_{1/2}y^k \ .$$

*This lack of simplification is inherent to the fact that we do not sketch the entire system matrix $\mathbf{H}+\mathbf{V}$, which is prohibited by the ADI structure.*

### 5.2.2 Error recurrence

By combining the two updates (5.16) and (5.15), we get the following error recurrence.

**Lemma 5.4** (Error recurrence)**.** *Consider the iterates $(u^k)_k$ given by* (5.16) *and* (5.15)*. If the weight matrices are set to identity $\mathbf{M_H} = \mathbf{M_V} = \mathbf{I}_d$ then*

$$u^{k+1} - u^* = \mathbf{T}_k(u^k - u^*), \tag{5.22}$$

*where*

$$\mathbf{T}_k \overset{def}{=} \left(\mathbf{I}_d - \mathbf{Z_V}(\mathbf{V}+q_k\mathbf{I}_d)^{-1}(\mathbf{H}+\mathbf{V})\right)\left(\mathbf{I}_d - \mathbf{Z_H}(\mathbf{H}+p_k\mathbf{I}_d)^{-1}(\mathbf{H}+\mathbf{V})\right) \ , \tag{5.23}$$

*or equivalently*

$$\begin{aligned}\mathbf{T}_k = &\left[(\mathbf{I}_d - \mathbf{Z_V})(\mathbf{H}-q_k\mathbf{I}_d)^{-1}(\mathbf{V}+q_k\mathbf{I}_d) - \mathbf{Z_V}\right](\mathbf{V}+q_k\mathbf{I}_d)^{-1}(\mathbf{H}-q_k\mathbf{I}_d)\\ &\times\left[(\mathbf{I}_d - \mathbf{Z_H})(\mathbf{V}-p_k\mathbf{I}_d)^{-1}(\mathbf{H}+p_k\mathbf{I}_d) - \mathbf{Z_H}\right](\mathbf{H}+p_k\mathbf{I}_d)^{-1}(\mathbf{V}-p_k\mathbf{I}_d) \ .\end{aligned} \tag{5.24}$$

With a closed form expression for the error recursion, we will then be able to analyse the rate of convergence.

*Proof.* Let $\delta_k \overset{\text{def}}{=} u^k - u^*$ be the gap between the $k$-th iterate and the solution. The first form for $\mathbf{T}_k$ in (5.23) follow immediately by plugging (5.15) into (5.16). The second form (5.24) follows since

$$\begin{aligned}
\delta_{k+1/2} \overset{\text{def}}{=} u^{k+1/2} - u^* &\overset{(5.15)}{=} u^k - u^* - \mathbf{Z_H}(\mathbf{H} + p_k\mathbf{I}_d)^{-1}R(u^k) \\
&\overset{(5.19)}{=} \delta_k - \mathbf{Z_H}(\mathbf{H} + p_k\mathbf{I}_d)^{-1}(\mathbf{H} + \mathbf{V})\delta_k \\
&= \left(\mathbf{I}_d - \mathbf{Z_H}(\mathbf{H} + p_k\mathbf{I}_d)^{-1}(\mathbf{H} + \mathbf{V})\right)\delta_k \\
&= \left[\mathbf{I}_d - \mathbf{Z_H}\left(\mathbf{I}_d + (\mathbf{H} + p_k\mathbf{I}_d)^{-1}(\mathbf{V} - p_k\mathbf{I}_d)\right)\right]\delta_k \\
&= \left[(\mathbf{I}_d - \mathbf{Z_H})(\mathbf{V} - p_k\mathbf{I}_d)^{-1}(\mathbf{H} + p_k\mathbf{I}_d) - \mathbf{Z_H}\right](\mathbf{H} + p_k\mathbf{I}_d)^{-1}(\mathbf{V} - p\mathbf{I}_d)\delta_k \ .
\end{aligned}$$

Analogously, the second sketched half-step can be written

$$\begin{aligned}
\delta_{k+1} = u^{k+1} - u^* &\overset{(5.16)}{=} u^{k+1/2} - u^* - \mathbf{Z_V}(\mathbf{V} + q\mathbf{I}_d)^{-1}R(u^{k+1/2}) \\
&= \left[(\mathbf{I}_d - \mathbf{Z_V})(\mathbf{H} - q_k\mathbf{I}_d)^{-1}(\mathbf{V} + q_k\mathbf{I}_d) - \mathbf{Z_V}\right](\mathbf{V} + q_k\mathbf{I}_d)^{-1}(\mathbf{H} - q_k\mathbf{I}_d)\delta_{k+1/2} \ ,
\end{aligned}$$

which leads to the factorized version of the reduction operator in (5.24). □

### 5.2.3 Particular sketches

#### 5.2.3.1 No sketching: recovering the Peaceman-Rachford method

In this section, we verify that we recover the original PR method when we do not sketch, *i.e.,* $\mathbf{S}_{1/2} = \mathbf{S}_1 = \mathbf{I}_d$. We set $\mathbf{M_H} = \mathbf{M_V} = \mathbf{I}_d$. Here, the operators $\mathbf{Z_H}, \mathbf{Z_V}$ boil down to identity matrices of order $d$. And since we solve the entire linear system at each half-step we recover the original Peaceman-Rachford method as stated in the following lemma.

**Lemma 5.5** (Recovering Peaceman-Rachford matrix)**.** *If we run the SPR method presented in* (5.13) *and* (5.14) *with full sketching at each half-step. Then, we recover the* Peaceman-Rachford matrix *[for $p = q$, see eq (15.3) in [18], eq (7.13) in [153], eq (4) p.3 in [156]] at the $k$-th step*

$$\mathbf{T}_k \overset{def}{=} (\mathbf{V} + q_k\mathbf{I}_d)^{-1}(\mathbf{H} - q_k\mathbf{I}_d)(\mathbf{H} + p_k\mathbf{I}_d)^{-1}(\mathbf{V} - p_k\mathbf{I}_d) \ . \tag{5.25}$$

*Proof.* Since sketch matrices are always the identity matrix, $\mathbf{Z_H}$ and $\mathbf{Z_V}$ also equal to the identity because

$$\begin{aligned}
\mathbf{Z_H} &= (\mathbf{H} + p_k\mathbf{I}_d)^\top \left((\mathbf{H} + p_k\mathbf{I}_d)(\mathbf{H} + p_k\mathbf{I}_d)^\top\right)^\dagger (\mathbf{H} + p_k\mathbf{I}_d) \\
&= (\mathbf{H} + p_k\mathbf{I}_d)^\top \left((\mathbf{H} + p_k\mathbf{I}_d)^\top\right)^\dagger (\mathbf{H} + p_k\mathbf{I}_d)^\dagger (\mathbf{H} + p_k\mathbf{I}_d) \\
&= \mathbf{I}_d \ .
\end{aligned}$$

So we have,

$$\begin{aligned}
\mathbf{T}_k &= \mathbf{I}_d - \left[(\mathbf{H} + p_k\mathbf{I}_d)^{-1} + (\mathbf{V} + q_k\mathbf{I}_d)^{-1} - (\mathbf{V} + q_k\mathbf{I}_d)^{-1}(\mathbf{H} + \mathbf{V})(\mathbf{H} + p_k\mathbf{I}_d)^{-1}\right](\mathbf{H} + \mathbf{V}) \\
&= (\mathbf{V} + q_k\mathbf{I}_d)^{-1}\left\{\mathbf{V} + q_k\mathbf{I}_d - \left[(\mathbf{V} + q_k\mathbf{I}_d)(\mathbf{H} + p_k\mathbf{I}_d)^{-1} + \mathbf{I}_d - (\mathbf{H} + \mathbf{V})(\mathbf{H} + p_k\mathbf{I}_d)^{-1}\right](\mathbf{H} + \mathbf{V})\right\} \\
&= (\mathbf{V} + q_k\mathbf{I}_d)^{-1}\left\{\mathbf{V} + q_k\mathbf{I}_d - (\mathbf{V} + q_k\mathbf{I}_d + \mathbf{H} + p_k\mathbf{I}_d - \mathbf{H} - \mathbf{V})(\mathbf{H} + p_k\mathbf{I}_d)^{-1}(\mathbf{H} + \mathbf{V})\right\} \\
&= (\mathbf{V} + q_k\mathbf{I}_d)^{-1}\left\{\mathbf{V} + q_k\mathbf{I}_d - (p_k + q_k)(\mathbf{H} + p_k\mathbf{I}_d)^{-1}(\mathbf{H} + p_k\mathbf{I}_d + \mathbf{V} - p_k\mathbf{I}_d)\right\} \\
&= (\mathbf{V} + q_k\mathbf{I}_d)^{-1}\left\{\mathbf{V} + q_k\mathbf{I}_d - (p_k + q_k)\mathbf{I}_d - (p_k + q_k)(\mathbf{H} + p_k\mathbf{I}_d)^{-1}(\mathbf{V} - p_k\mathbf{I}_d)\right\} \\
&= (\mathbf{V} + q_k\mathbf{I}_d)^{-1}\left\{\mathbf{I}_d - (p_k + q_k)(\mathbf{H} + p_k\mathbf{I}_d)^{-1}\right\}(\mathbf{V} - p_k\mathbf{I}_d) \\
&= (\mathbf{V} + q_k\mathbf{I}_d)^{-1}(\mathbf{H} - q_k\mathbf{I}_d)(\mathbf{H} + p_k\mathbf{I}_d)^{-1}(\mathbf{V} - p_k\mathbf{I}_d) \ .
\end{aligned}$$

where we used in the last equality the fact that

$$(\mathbf{H} - q_k\mathbf{I}_d)(\mathbf{H} + p_k\mathbf{I}_d)^{-1} = (\mathbf{H} + p\mathbf{I}_d - (p_k + q_k)\mathbf{I}_d)(\mathbf{H} + p\mathbf{I}_d)^{-1} = \mathbf{I}_d - (p_k + q_k)(\mathbf{H} + p\mathbf{I}_d)^{-1} \ .$$

□

**Peaceman-Rachford minimax problem.** The Peaceman-Rachford matrix (5.25) controls the error recurrence, presented in its sketched version in (5.22), thus it controls the convergence of the method.

**Lemma 5.6** (PR minimax problem (eq (6) of [156], eq (5) of [158]))**.** *Let* $\mathbf{H}$ *and* $\mathbf{V}$ *be* $d \times d$ *be SPD matrices such that they commute,* i.e., $\mathbf{HV} = \mathbf{VH}$*. Let* $(\lambda_i)_{i=1}^d$ *and* $(\gamma_i)_{i=1}^d$ *be the eigenvalues of* $\mathbf{H}$ *and* $\mathbf{V}$*, respectively. Let us assume that we know bounds of the spectra of* $\mathbf{H}$ *and* $\mathbf{V}$*, that is* $\sigma(\mathbf{H}) \subset [a, b]$ *and* $\sigma(\mathbf{H}) \subset [\alpha, \beta]$ *where* $0 < a \leq b$ *and* $0 < \alpha \leq \beta$*.*

*After* $K \in \mathbb{N}^*$ *iterations, the error is given by* $u^K - u^* \leq \mathbf{T}_K(u^0 - u^*)$ *where* $\mathbf{T}_K \overset{def}{=} \prod_{k=1}^K \mathbf{T}_k$*. And the Peaceman-Rachford method in* (5.2)–(5.3) *converges according to the following relation*

$$\left\|u^K - u^*\right\| \leq \|\mathbf{T}_K\| \left\|u^0 - u^*\right\| \ , \tag{5.26}$$

*where* $\|\mathbf{T}_K\|$ *denotes the spectral norm of* $\mathbf{T}_K$ *and equals*

$$\|\mathbf{T}_K\| \overset{def}{=} \max_{\substack{\lambda \in \sigma(\mathbf{H}) \\ \gamma \in \sigma(\mathbf{V})}} \left| \prod_{k=1}^K \frac{(q_k - \lambda)(p_k - \gamma)}{(p_k + \lambda)(q_k + \gamma)} \right| \ . \tag{5.27}$$

*Furthermore,* $\|\mathbf{T}_K\|$ *can be upper bounded by the following convergence rate*

$$\|\mathbf{T}_K\| \leq \rho_K \overset{def}{=} \max_{\substack{\lambda \in [a,b] \\ \gamma \in [\alpha,\beta]}} \left| \prod_{k=1}^K \frac{(q_k - \lambda)(p_k - \gamma)}{(p_k + \lambda)(q_k + \gamma)} \right| \ . \tag{5.28}$$

In Lemma 5.6, we can clearly see the importance of the shift parameters in the convergence of the PR method. Indeed, the convergence rate $\rho_K$ can be minimized by selecting $p_k, q_k$ that optimize the rate of convergence in (5.28). This new optimization problem is called the *minimax Peaceman-Rachford problem*:

$$\{p_k^*\}_{k=1}^K, \{q_k^*\}_{k=1}^K \in \underset{\{p_k\}_{k=1}^K, \{q_k\}_{k=1}^K}{\arg\min} \max_{\substack{\lambda \in [a,b] \\ \gamma \in [\alpha,\beta]}} \left| \prod_{k=1}^K \frac{(q_k - \lambda)(p_k - \gamma)}{(p_k + \lambda)(q_k + \gamma)} \right| \ . \tag{5.29}$$

**Remark 5.7** (Spectral information and shift parameters)**.** *It's worth remarking that the minimax problem presented above requires the number of iterations* $K$ *to be known in advance. More rigorously, the choice of* $K$*,* $p_k$*'s and* $q_k$*'s are interdependent. Whereas for classical iterative algorithms, we prefer to have hyper-parameters that can be set independently of the convergence rate. Hence, the number of iterations is then deduced from the theoretical complexity required to achieve* $\epsilon$ *tolerance/suboptimality.*

#### 5.2.3.2 Single row sketching: $\mathbf{S}_{1/2} = e^i$ and $\mathbf{S}_1 = e^j$

Here we explore what we get if we use single-row sketching at each iteration, *i.e.,* $\mathbf{S}_{1/2} = e^i$, where $e^i$ denotes the $i$-th basis vector of $\mathbb{R}^d$, with $i$ sampled uniformly at random in $[d] \overset{\text{def}}{=} \{1, \ldots, d\}$ (same procedure for $\mathbf{S}_1$). Studying first single-row subsampling is justified because it can easily be extended to block subsampling, where multiple rows are sampled at once. It is a verify easy sketching method to implement since it only requires slicing operations which are often numerically very efficient (like with `NumPy` arrays [152]).

We consider independent row sketching at each half-step: $\mathbf{S}_{1/2} = e^i$ and $\mathbf{S}_1 = e^j$, where $j$ does not necessarily equal to $i$. The shift parameters are assumed to be constant $p, q \in \mathbb{R}$ and we set weight

**Algorithm 15** Block Sketched Peaceman-Rachford (BSPR)

**Input:** shift parameters $(p_k)_k, (q_k)_k$, sketch size $\tau$

**Initialize:** $u^0 = 0_d$

**for** $k = 0, 1, \ldots, K-1$ **do**

  Sample $C_{1/2}, C_1 \subset [d]$ s.t. $|C_{1/2}| = |C_1| = \tau$ ▷ sampled row indices

  $r_{\mathbf{S}}^k = \mathbf{H}_{C_{1/2}} u^k + \mathbf{V}_{C_{1/2}} u^k - s_{C_{1/2}}$ ▷ 1st sketched residual

  $\tilde{\mathbf{H}}_{C_{1/2}:} = \mathbf{H}_{C_{1/2}:} + p_k \mathbf{I}_{C_{1/2}:}$ ▷ sketched shifted $\mathbf{H}$

  $u^{k+1/2} = u^k - (\tilde{\mathbf{H}}_{C_{1/2}:})^\top \left( \tilde{\mathbf{H}}_{C_{1/2}:} (\tilde{\mathbf{H}}_{C_{1/2}:})^\top \right)^\dagger r_{\mathbf{S}}^k$ ▷ 1st half-step

  $r_{\mathbf{S}}^{k+1/2} = \mathbf{H}_{C_1:} u^{k+1/2} + \mathbf{V}_{C_1:} u^{k+1/2} - s_{C_1}$ ▷ 2nd sketched residual

  $\tilde{\mathbf{V}}_{C_1:} = \mathbf{V}_{C_1:} + q_k \mathbf{I}_{C_1:}$ ▷ sketched shifted $\mathbf{V}$

  $u^{k+1} = u^{k+1/2} - (\tilde{\mathbf{V}}_{C_1:})^\top \left( \tilde{\mathbf{V}}_{C_1:} (\tilde{\mathbf{V}}_{C_1:})^\top \right)^\dagger r_{\mathbf{S}}^{k+1/2}$ ▷ 2nd half-step

**Output:** $u^K$

matrices to identity, *i.e.,* $\mathbf{M_H} = \mathbf{M_V} = \mathbf{I}_d$. The following lemma gives the simplified $\mathbf{Z_H}$ and $\mathbf{Z_V}$ operators when using row sketching.

**Lemma 5.8** ($\mathbf{Z}$ matrices with row sketching)**.** *Let $(e^k)_{1 \le k \le d}$ be the canonical basis of $\mathbb{R}^d$. If the sketching matrices are $\mathbf{S}_{1/2} = e^i$ and $\mathbf{S}_1 = e^j$, then, the $\mathbf{Z_H}$ and $\mathbf{Z_V}$ matrices of the SPR method defined in* (5.17)-(5.18) *become respectively*

$$\mathbf{Z_H} = \frac{1}{\|(\mathbf{H}_{i:})^\top + p e^i\|_2^2} (\mathbf{H} + p\mathbf{I})^\top e^i \left(e^i\right)^\top (\mathbf{H} + p\mathbf{I}) \tag{5.30}$$

*and*

$$\mathbf{Z_V} = \frac{1}{\|(\mathbf{V}_{j:})^\top + q e^j\|_2^2} (\mathbf{V} + q\mathbf{I})^\top e^j \left(e^j\right)^\top (\mathbf{V} + q\mathbf{I}) \ , \tag{5.31}$$

*where $\mathbf{H}_{i:}$ (resp. $\mathbf{V}_{j:}$) denotes the $i$-th (resp. $j$-th) row of matrix $\mathbf{H}$ (resp. $\mathbf{V}$), thus of dimension $1 \times d$.*

*Proof.* Equation (5.17) becomes

$$\begin{aligned} \mathbf{Z_H} &= (\mathbf{H} + p\mathbf{I}_d)^\top e^i \underbrace{\left( \left(e^i\right)^\top (\mathbf{H} + p\mathbf{I}_d)(\mathbf{H} + p\mathbf{I}_d)^\top e^i \right)^\dagger}_{\in \mathbb{R}} \left(e^i\right)^\top (\mathbf{H} + p\mathbf{I}_d) \\ &= \frac{1}{\|(\mathbf{H}_{i:})^\top + p e^i\|_2^2} (\mathbf{H} + p\mathbf{I}_d)^\top e^i \left(e^i\right)^\top (\mathbf{H} + p\mathbf{I}_d) \ . \end{aligned} \tag{5.32}$$

The same computation applies for $\mathbf{Z_V}$. □

Half-updates (5.15) – (5.16) then become

$$u^{k+1/2} = u^k - \frac{1}{\|(\mathbf{H}_{i:})^\top + p e^i\|_2^2} (\mathbf{H} + p\mathbf{I}_d)^\top e^i \left(e^i\right)^\top R(u^k) \ , \tag{5.33}$$

$$u^{k+1} = u^{k+1/2} - \frac{1}{\|(\mathbf{V}_{j:})^\top + q e^j\|_2^2} (\mathbf{V} + q\mathbf{I}_d)^\top e^j \left(e^j\right)^\top R(u^{k+1/2}) \ , \tag{5.34}$$

where $\mathbf{V}_{j:}$ denotes the $j$-th row of matrix $\mathbf{V}$. We call the SPR method with block subsampling the *Block Sketched Peaceman-Rachford (BSPR)* method, and give its implementation in Algorithm 15.

**Importance sampling** To simplify the analysis of this method in Section 5.3, we suggest not sampling uniformly the rows out of $[d]$. Rather, for every $i \in [d]$ let

$$\mathbb{P}\left[\mathbf{S}_{1/2} = e^i\right] = \frac{\left\|(\mathbf{H}_{i:})^\top + pe^i\right\|_2^2}{\sum_{l=1}^d \left\|(\mathbf{H}_{l:})^\top + pe^l\right\|_2^2} = \frac{\left\|(\mathbf{H}_{i:})^\top + pe^i\right\|_2^2}{\left\|\mathbf{H}^\top + p\mathbf{I}_d\right\|_F^2} \ , \tag{5.35}$$

and analogously for the second sketch matrix

$$\mathbb{P}\left[\mathbf{S}_1 = e^j\right] = \frac{\left\|(\mathbf{V}_{j:})^\top + qe^j\right\|_2^2}{\sum_{l=1}^d \left\|(\mathbf{V}_{l:})^\top + qe^l\right\|_2^2} = \frac{\left\|(\mathbf{V}_{j:})^\top + qe^j\right\|_2^2}{\left\|\mathbf{V}^\top + q\mathbf{I}_d\right\|_F^2} \ . \tag{5.36}$$

Taking expectation over (5.32), and analogously for $\mathbb{E}\left[\mathbf{Z}_\mathbf{V}\right]$, gives

$$\mathbb{E}\left[\mathbf{Z}_\mathbf{H}\right](\mathbf{H} + p\mathbf{I}_d)^{-1} = \frac{(\mathbf{H}^\top + p\mathbf{I}_d)}{\left\|\mathbf{H}^\top + p\mathbf{I}_d\right\|_F^2} \ , \tag{5.37}$$

$$\mathbb{E}\left[\mathbf{Z}_\mathbf{V}\right](\mathbf{V} + q\mathbf{I}_d)^{-1} = \frac{(\mathbf{V}^\top + q\mathbf{I}_d)}{\left\|\mathbf{V}^\top + q\mathbf{I}_d\right\|_F^2} \ . \tag{5.38}$$

**Remark 5.9** (Importance samlping with moving shift parameters)**.** *If one deals with moving shift parameters $(p_k)_k$ (resp. $(q_k)_k$), it is required to update the probability distribution of the row sketches after each step. Thus from step $k$ to step $k+1$, the probability of sampling the vector $e^i$ needs to be updated from*

$$\mathbb{P}\left[\mathbf{S}_{1/2}^k = e^i\right] \propto \left\|(\mathbf{H}_{i:})^\top + p_k e^i\right\|_2^2 = \left\|(\mathbf{H}_{i:})^\top\right\|_2^2 + p_k^2 + 2p_k\mathbf{H}_{ii}$$

*to*

$$\mathbb{P}\left[\mathbf{S}_{1/2}^{k+1} = e^i\right] \propto \left\|(\mathbf{H}_{i:})^\top + p_{k+1} e^i\right\|_2^2 = \left\|(\mathbf{H}_{i:})^\top\right\|_2^2 + p_{k+1}^2 + 2p_{k+1}\mathbf{H}_{ii} \ .$$

*So, from step $k$ to step $k+1$ we update the probabilities the following way:*

$$\mathbb{P}\left[\mathbf{S}_{1/2}^{k+1} = e^i\right] \leftarrow \mathbb{P}\left[\mathbf{S}_{1/2}^{k} = e^i\right] + p_{k+1}^2 - p_k^2 + 2(p_{k+1} - p_k)\mathbf{H}_{ii} \ ,$$

*and then perform a normalization of the resulting probability vector $\left(\mathbb{P}\left[\mathbf{S}_{1/2}^{k+1} = e^i\right]\right)_{i=1\dots d} \in \mathbb{R}^d$. This is basically a sum of the $d$ entries of the vector followed by an element-wise division. The same applies to $\mathbb{P}\left[\mathbf{S}_1^{k+1} = e^j\right]$.*

With importance sampling taking expectation over (5.33) and (5.34), and using (5.38), gives the *expected half-steps* of SPR (with constant shift parameters $p, q$):

$$\mathbb{E}\left[u^{k+1/2} - u^*\right] = \mathbb{E}\left[u^k - u^*\right] - \frac{\mathbf{H}^\top + p\mathbf{I}_d}{\left\|\mathbf{H}^\top + p\mathbf{I}_d\right\|_F^2}(\mathbf{H} + \mathbf{V})\mathbb{E}\left[u^k - u^*\right] \ , \tag{5.39}$$

$$\mathbb{E}\left[u^{k+1} - u^*\right] = \mathbb{E}\left[u^{k+1/2} - u^*\right] - \frac{\mathbf{V}^\top + q\mathbf{I}_d}{\left\|\mathbf{V}^\top + q\mathbf{I}_d\right\|_F^2}(\mathbf{H} + \mathbf{V})\mathbb{E}\left[u^{k+1/2} - u^*\right] \ . \tag{5.40}$$

Theses relations will be of great use in our convergence analysis in Section 5.3.

## 5.3 Convergence analysis of the SPR method

In this section, we set $\mathbf{M}_\mathbf{H} = \mathbf{M}_\mathbf{V} = \mathbf{I}_d$ and use constant shifts $p, q \in \mathbb{R}$ throughout to simplify the calculations. We will now analyse the convergence of the random iterates generated by the SPR method.

### 5.3.1 Convergence in expectation for single row sketching

In this section, we study the convergence of the SPR method with single row sketching as described in Section 5.2.3.2 with sampling probabilities set in (5.35)–(5.36). Taking expectation over the two half-steps update given in (5.33) and (5.34), and using expectations (5.37) and (5.38), gives

$$\mathbb{E}\left[u^{k+1/2} - u^*\right] = \mathbb{E}\left[u^k - u^*\right] - \frac{\mathbf{H}^\top + p\mathbf{I}_d}{\|\mathbf{H}^\top + p\mathbf{I}_d\|_F^2}(\mathbf{H}+\mathbf{V})\mathbb{E}\left[u^k - u^*\right]$$
$$\mathbb{E}\left[u^{k+1} - u^*\right] = \mathbb{E}\left[u^{k+1/2} - u^*\right] - \frac{\mathbf{V}^\top + q\mathbf{I}_d}{\|\mathbf{V}^\top + q\mathbf{I}_d\|_F^2}(\mathbf{H}+\mathbf{V})\mathbb{E}\left[u^{k+1/2} - u^*\right] \ .$$

Putting the two together we get

$$\mathbb{E}\left[u^{k+1} - u^*\right] = \left(\mathbf{I}_d - \frac{\mathbf{V}^\top + q\mathbf{I}_d}{\|\mathbf{V}^\top + q\mathbf{I}_d\|_F^2}(\mathbf{H}+\mathbf{V})\right)\left(\mathbf{I}_d - \frac{\mathbf{H}^\top + p\mathbf{I}_d}{\|\mathbf{H}^\top + p\mathbf{I}_d\|_F^2}(\mathbf{H}+\mathbf{V})\right)\mathbb{E}\left[u^k - u^*\right] \ . \quad (5.41)$$

Furthermore, we have that the expected residual depends on almost[3] the same operator:

$$\begin{aligned}\mathbb{E}\left[R(u^{k+1})\right] &= \mathbb{E}\left[(\mathbf{H}+\mathbf{V})(u^{k+1} - u^*)\right] \\ &\overset{(5.41)}{=} (\mathbf{H}+\mathbf{V})\left(\mathbf{I}_d - \frac{\mathbf{V}^\top + q\mathbf{I}_d}{\|\mathbf{V}^\top + q\mathbf{I}_d\|_F^2}(\mathbf{H}+\mathbf{V})\right)\left(\mathbf{I}_d - \frac{\mathbf{H}^\top + p\mathbf{I}_d}{\|\mathbf{H}^\top + p\mathbf{I}_d\|_F^2}(\mathbf{H}+\mathbf{V})\right)\mathbb{E}\left[u^k - u^*\right] \\ &\overset{(5.19)}{=} \left(\mathbf{I}_d - (\mathbf{H}+\mathbf{V})\frac{\mathbf{V}^\top + q\mathbf{I}_d}{\|\mathbf{V}^\top + q\mathbf{I}_d\|_F^2}\right)\left(\mathbf{I}_d - (\mathbf{H}+\mathbf{V})\frac{\mathbf{H}^\top + p\mathbf{I}_d}{\|\mathbf{H}^\top + p\mathbf{I}_d\|_F^2}\right)\mathbb{E}\left[R(u^k)\right] \ .\end{aligned}$$

Now it is much easier to study the convergence rate of the expected iterates in the above. The next theorem follows closely assumptions and proofs given in [18] without leveraging multiple shift parameters.

**Theorem 5.10** (Convergence in expectation in the commutative case)**.** *Let* $\mathbf{H}$ *and* $\mathbf{V}$ *be* $d \times d$ *SPD matrices of order* $d$ *that commute with each other,* i.e., $\mathbf{HV} = \mathbf{VH}$*. Furthermore, we assume that we know bounds of the spectra of* $\mathbf{H}$ *and* $\mathbf{V}$*,* e.g., $0 < a \leq \lambda_i \leq b$ *and* $0 < \alpha \leq \gamma_i \leq \beta$ *for all* $i \in [d]$*, where* $\lambda_i$*'s and* $\gamma_i$*'s are respectively the eigenvalues of* $\mathbf{H}$ *and* $\mathbf{V}$*.*
*Then, after* $K \in \mathbb{N}^*$ *steps of the SPR method (for* $\mathbf{M_H} = \mathbf{M_V} = \mathbf{I}_d$*) with row sketching presented in* (5.33)–(5.34) *and constant shift parameters* $p \geq \frac{1}{2}(\beta - a)$ *and* $q \geq \frac{1}{2}(b - \alpha)$*, the expected iterates converge according to the following relation*

$$\left\|\mathbb{E}\left[u^K - u^*\right]\right\| \leq \rho_E^K \left\|\mathbb{E}\left[u^0 - u^*\right]\right\| \ , \quad (5.42)$$

*where the convergence rate is denoted*

$$\rho_E \overset{def}{=} \max_{i=1,\dots,d}\left\{\left|1 - \frac{(\gamma_i + q)(\gamma_i + \lambda_i)}{\sum_{j=1}^d (\gamma_j + q)^2}\right| \cdot \left|1 - \frac{(\lambda_i + p)(\lambda_i + \gamma_i)}{\sum_{j=1}^d (\lambda_j + p)^2}\right|\right\} < 1 \ . \quad (5.43)$$

*Proof.* From Theorem 7.2 of [153], we know that if $\mathbf{H}$ and $\mathbf{V}$ are symmetric matrices such that $\mathbf{HV} = \mathbf{VH}$, then there exists a common orthonormal basis of eigenvectors $(z_i)_{i=1}^d$. For $i \in [d]$, we denote $\lambda_i$ (resp. $\gamma_i$) the eigenvalue of $\mathbf{H}$ (resp. $\mathbf{V}$) corresponding to the $i$-th eigenvector $z_i$.

Using the fact that $\mathbf{H}^\top = \mathbf{H}$ and $\mathbf{V}^\top = \mathbf{V}$ and by taking norms in (5.41), we get at step $k+1 \in \mathbb{N}^*$

$$\begin{aligned}\left\|\mathbb{E}\left[u^{k+1} - u^*\right]\right\| &\leq \left\|\left(\mathbf{I}_d - \frac{\mathbf{V} + q\mathbf{I}_d}{\|\mathbf{V} + q\mathbf{I}_d\|_F^2}(\mathbf{H}+\mathbf{V})\right)\left(\mathbf{I}_d - \frac{\mathbf{H} + p\mathbf{I}_d}{\|\mathbf{H} + p\mathbf{I}_d\|_F^2}(\mathbf{H}+\mathbf{V})\right)\right\| . \left\|\mathbb{E}\left[u^k - u^*\right]\right\| \\ &\overset{\text{def}}{=} \|\mathbf{T}\| . \left\|\mathbb{E}\left[u^k - u^*\right]\right\|\end{aligned}$$

[3] Almost, since in Theorem 5.10 we assume the commutativity of $\mathbf{H}$ and $\mathbf{V}$.

where $\|.\|$ designates either the Euclidean vector norm or the corresponding subordinate matrix norm. Clearly, the family $(z_i)_{i=1}^n$ is also a basis of eigenvectors for the matrix $\mathbf{T}$. Let $i \in [d]$, then

$$\begin{aligned}\mathbf{T}z_i &= \left(\mathbf{I}_d - \frac{\mathbf{V}+q\mathbf{I}_d}{\|\mathbf{V}+q\mathbf{I}_d\|_F^2}(\mathbf{H}+\mathbf{V})\right)\left(\mathbf{I}_d - \frac{\mathbf{H}+p\mathbf{I}_d}{\|\mathbf{H}+p\mathbf{I}_d\|_F^2}(\mathbf{H}+\mathbf{V})\right) z_i \\ &= \left(1 - \frac{(\gamma_i+q)(\lambda_i+\gamma_i)}{\|\mathbf{V}+q\mathbf{I}_d\|_F^2}\right)\left(1 - \frac{(\lambda_i+p)(\lambda_i+\gamma_i)}{\|\mathbf{H}+p\mathbf{I}_d\|_F^2}\right) z_i \\ &= \left(1 - \frac{(\gamma_i+q)(\lambda_i+\gamma_i)}{\sum_{j=1}^d(\gamma_j+q)^2}\right)\left(1 - \frac{(\lambda_i+p)(\lambda_i+\gamma_i)}{\sum_{j=1}^d(\lambda_j+p)^2}\right) z_i \ ,\end{aligned}$$

where the last equality comes from Definition 5.23 (of the Frobenius norm) and Lemma 5.24 [4]. Moreover, since $\mathbf{H}$ and $\mathbf{V}$ are symmetric and commutative matrices, the left-hand and right-hand side matrices forming $\mathbf{T}$ are symmetric since

$$\begin{aligned}&\left(\mathbf{I}_d - \frac{(\mathbf{H}+p\mathbf{I}_d)}{\|\mathbf{H}+p\mathbf{I}_d\|_F^2}(\mathbf{H}+\mathbf{V})\right)^\top \overset{\substack{\mathbf{H}^\top=\mathbf{H}\\ \mathbf{V}^\top=\mathbf{V}}}{=} \left(\mathbf{I}_d - \frac{\mathbf{H}+\mathbf{V}}{\|\mathbf{H}+p\mathbf{I}_d\|_F^2}(\mathbf{H}+p\mathbf{I}_d)\right) \\ &\qquad = \left(\mathbf{I}_d - \frac{1}{\|\mathbf{H}+p\mathbf{I}_d\|_F^2}(\mathbf{H}^2+\mathbf{V}\mathbf{H}+p\mathbf{H}+p\mathbf{V})\right) \\ &\qquad \overset{\mathbf{HV}=\mathbf{VH}}{=} \left(\mathbf{I}_d - \frac{1}{\|\mathbf{H}+p\mathbf{I}_d\|_F^2}(\mathbf{H}^2+\mathbf{H}\mathbf{V}+p\mathbf{H}+p\mathbf{V})\right) \\ &\qquad = \left(\mathbf{I}_d - \frac{(\mathbf{H}+p\mathbf{I}_d)}{\|\mathbf{H}+p\mathbf{I}_d\|_F^2}(\mathbf{H}+\mathbf{V})\right) \ ,\end{aligned}$$

and analogously for the left-hand side. Moreover, those matrices commute because

$$(\mathbf{V}+q\mathbf{I}_d)(\mathbf{H}+\mathbf{V})(\mathbf{H}+p\mathbf{I}_d) = (\mathbf{H}+p\mathbf{I}_d)(\mathbf{H}+\mathbf{V})(\mathbf{V}+q\mathbf{I}_d) \ .$$

We then apply Lemma 5.18, and deduce that $\mathbf{T}$ is also symmetric.[5] From this observation, we deduce that the convergence of our SPR algorithm is given by the spectral radius of $\mathbf{T}$, that is

$$\rho_{\mathrm{E}} \overset{\mathrm{def}}{=} \|\mathbf{T}\| \overset{\text{Lemma 5.22}}{=} \rho(\mathbf{T}) = \max_{i\in[d]}\left\{\left|1 - \frac{(\gamma_i+q)(\gamma_i+\lambda_i)}{\sum_{j=1}^d(\gamma_j+q)^2}\right| \cdot \left|1 - \frac{(\lambda_i+p)(\lambda_i+\gamma_i)}{\sum_{j=1}^d(\lambda_j+p)^2}\right|\right\} \ . \tag{5.44}$$

The next step is to find conditions on $p$ and $q$ to ensure that $\rho(\mathbf{T})$ is strictly smaller than one, *i.e.,* the matrix $\mathbf{T}$ is contracting. Thus the convergence of the SPR method would be guaranteed. Let $i \in [d]$, let us find a condition on $q$ such that

$$\begin{aligned}\left|1 - \frac{(\gamma_i+q)(\gamma_i+\lambda_i)}{\sum_{j=1}^d(\gamma_j+q)^2}\right| < 1 \iff 0 < \frac{(\gamma_i+q)(\gamma_i+\lambda_i)}{\sum_{j=1}^d(\gamma_j+q)^2} < 2 \\ \iff 0 < (\gamma_i+q)(\gamma_i+\lambda_i) < 2\sum_{j=1}^d(\gamma_j+q)^2 \ .\end{aligned} \tag{5.45}$$

Like in section 7 of [18], we assume that we know bounds of the spectra of $\mathbf{H}$ and $\mathbf{V}$ such that for all $i \in [d]$, $0 < a \leq \lambda_i \leq b$ and $0 < \alpha \leq \gamma_i \leq \beta$.

[4] The squared Frobenius norm is the sum of the squared singular values and that the singular values of a symmetric positive definite matrix match its eigenvalues.

[5] Simpler argument: let $\mathbf{U}$ be the unitary matrix, *i.e.,* $\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}_d$, formed by the orthonormal vectors $(z_i)_{i=1}^d$. Then, $\mathbf{U}\mathbf{H}\mathbf{U}^\top = \mathbf{diag}\,(\lambda_i)$ and $\mathbf{U}\mathbf{V}\mathbf{U}^\top = \mathbf{diag}\,(\gamma_i)$ which implies that $\mathbf{U}\mathbf{T}\mathbf{U}^\top$ is also diagonal. This implies the matrix $\mathbf{T}$ is symmetric.

If $q > -\alpha$, then the left-hand side inequality of (5.45) is satisfied for all $i \in [d]$. Deducing a necessary and sufficient condition on $q$ for the right-hand side inequality is too difficult, instead we only look for a sufficient condition.

If $q \geq \frac{1}{2}(b-\alpha)$, which always satisfies the previous condition, then for all $i \in [d]$,

$$\lambda_i - \gamma_i \leq b - \alpha \implies \lambda_i + \gamma_i \leq 2(\gamma_i + q) \ .$$

And consequently,

$$(\gamma_i + q)(\gamma_i + \lambda_i) \leq 2(\gamma_i + q)^2 < 2\sum_{j=1}^{d} \underbrace{(\gamma_j + q)}_{>0}^2 \ .$$

and thus the right-hand side of (5.45) is verified. The same reasoning applies to $p \geq \frac{1}{2}(\beta - a)$.

Finally, we have proved that if $p \geq \frac{1}{2}(\beta - a)$ and $q \geq \frac{1}{2}(b-\alpha)$, then

$$\left|1 - \frac{(\gamma_i + q)(\gamma_i + \lambda_i)}{\sum_{j=1}^{d}(\gamma_j + q)^2}\right| \cdot \left|1 - \frac{(\lambda_i + p)(\lambda_i + \gamma_i)}{\sum_{j=1}^{d}(\lambda_j + p)^2}\right| < 1 \qquad \forall i \in [d] \ , \tag{5.46}$$

which implies that the convergence rate $\rho_{\mathrm{E}}$ given in (5.44) is strictly smaller than one, *i.e.,* matrix $\mathbf{T}$ is contracting. □

**Remark 5.11** (Leveraging moving shift parameters)**.** *It worths noting that, contrary to the classical ADI convergence theory, we cannot prove any benefit of using different shifts $p_k, q_k$'s at each iteration. Indeed, ADI theory studies the full operator $\mathbf{T}_K$ from step 1 to step $K$, as described briefly in the minimax PR problem in Lemma 5.6. Whereas here for SPR, we focus on bounding the spectral radius of the contracting operator $\mathbf{T}$ between two steps $k$ and $k+1$. Also note that SPR is not a randomized Smith's method [143],where $p = q$ along the iterations, since we use two different shifts $p$ and $q$.*

### 5.3.2 Sketch of convergence in $\ell^2$–norm

Here, we study the convergence in $\ell^2$–norm of the SPR method for a general sketching method. We are interested in this type of convergence since it is stronger than convergence in expectation, as stated in the following lemma.

**Lemma 5.12** (Lemma 10 of [52])**.** *The convergence in $\ell^2$–norm implies the convergence of the norm of the expected error as can be seen through the following equality. Let $(\delta^k)_k$ be a sequence of random vectors. It follows,*

$$\mathbb{E}\left[\left\|\delta^k\right\|_2^2\right] = \left\|\mathbb{E}\left[\delta^k\right]\right\|_2^2 + \mathbb{E}\left[\left\|\delta^k - \mathbb{E}\left[\delta^k\right]\right\|_2^2\right] \geq \left\|\mathbb{E}\left[\delta^k\right]\right\|_2^2 \ .$$

To study the $\ell^2$–norm convergence of the SPR method, we start back from Lemma 5.4 which gives that

$$\begin{aligned}
\delta_{k+1} &\overset{\text{def}}{=} u^{k+1} - u^* \\
&= \left(\mathbf{I}_d - \mathbf{Z}_{\mathbf{V}}(\mathbf{V} + q\mathbf{I}_d)^{-1}(\mathbf{H} + \mathbf{V})\right)\left(\mathbf{I}_d - \mathbf{Z}_{\mathbf{H}}(\mathbf{H} + p\mathbf{I}_d)^{-1}(\mathbf{H} + \mathbf{V})\right)(u^k - u^*) \qquad (5.47) \\
&= (\mathbf{I}_d - \mathbf{\Phi})\,\delta_k \ , \qquad (5.48)
\end{aligned}$$

where

$$\mathbf{\Phi} \overset{\text{def}}{=} \left(\mathbf{Z}_{\mathbf{V}}(\mathbf{V} + q\mathbf{I}_d)^{-1} - \mathbf{Z}_{\mathbf{H}}(\mathbf{H} + p\mathbf{I}_d)^{-1} + \mathbf{Z}_{\mathbf{V}}(\mathbf{V} + q\mathbf{I}_d)^{-1}(\mathbf{H} + \mathbf{V})\mathbf{Z}_{\mathbf{H}}(\mathbf{H} + p\mathbf{I}_d)^{-1}\right)(\mathbf{H} + \mathbf{V}) \ . \tag{5.49}$$

Hence (5.48) indicates that the convergence is controlled by the spectrum of the random operator $\mathbf{\Phi}$. The following theorem gives the first step of the convergence proof in $\ell^2$–norm.

**Theorem 5.13** (Convergence in $\ell^2$–norm)**.** *Let* $\mathbf{H}$ *and* $\mathbf{V}$ *be* $d \times d$ *matrices. Let us define*

$$\mathbf{\Psi} \overset{def}{=} \mathbb{E}\left[\mathbf{\Phi}\right]^\top + \mathbb{E}\left[\mathbf{\Phi}\right] - \mathbb{E}\left[\mathbf{\Phi}^\top \mathbf{\Phi}\right] \quad , \tag{5.50}$$

*where* $\mathbf{\Phi} \in \mathbb{R}^{d\times d}$ *is given by* (5.49)*. And let us assume that the eigenvalues of* $\mathbf{\Psi}$ *belong to* $]0, 1]$*.*

*After* $K \in \mathbb{N}^*$ *steps of SPR method (for* $\mathbf{M_H} = \mathbf{M_V} = \mathbf{I}_d$*) and with constant shift parameters* $p, q \in \mathbb{R}$ *presented in* (5.15)–(5.16)*, the norm of the error converges according to the following relation*

$$\mathbb{E}\left[\left\|u^K - u^*\right\|_2^2\right] \leq \rho_{L2}^K \, \mathbb{E}\left[\left\|u^0 - u^*\right\|_2^2\right] \quad , \tag{5.51}$$

*where the convergence rate is*

$$\rho_{L2} \overset{def}{=} 1 - \lambda_{\min}\left(\mathbf{\Psi}\right) \quad . \tag{5.52}$$

*Proof.* Taking squared norm on both sides of the error recurrence written in (5.48) gives

$$\begin{aligned}\left\|\delta_{k+1}\right\|_2^2 &= \left\langle \left(\mathbf{I}_d - \mathbf{\Phi}\right)^\top \left(\mathbf{I}_d - \mathbf{\Phi}\right) \delta_k, \delta_k \right\rangle \\ &= \left\|\delta_k\right\|_2^2 - \left\langle \left(\mathbf{\Phi}^\top + \mathbf{\Phi} - \mathbf{\Phi}^\top \mathbf{\Phi}\right) \delta_k, \delta_k \right\rangle .\end{aligned}$$

Then we take conditional expectation with respect to $\delta_k$ and then expectation with respect to $\mathbf{S}_1$ and $\mathbf{S}_{1/2}$ so that

$$\begin{aligned}\mathbb{E}\left[\left\|\delta_{k+1}\right\|_2^2 \mid \delta_k\right] &\overset{(5.50)}{=} \left\|\delta_k\right\|_2^2 - \left\langle \mathbf{\Psi}\delta_k, \delta_k \right\rangle \\ &= \left\|\delta_k\right\|_2^2 - \lambda_{\min}\left(\mathbf{\Psi}\right) \left\|\delta_k\right\|_2^2 \quad .\end{aligned}$$

where the second inequality comes from the fact that $\mathbf{\Psi} \succ 0_{d\times d}$. Finally, we take expectation and finish proving that the SPR method converges in $\ell^2$–norm according to

$$\mathbb{E}\left[\left\|\delta_{k+1}\right\|_2^2\right] \leq \rho_{\mathrm{L2}} \, \mathbb{E}\left[\left\|\delta_k\right\|_2^2\right] \quad .$$

□

The natural question arising is

*Does the operator* $\mathbf{\Psi}$ *satisfy the assumption of having its eigenvalues between* 0 *and* 1 *?*

We tried to prove that this assumption holds in a very general case, without any success. In future work, we will try to specialize this theorem for simple sketching methods, for instance single row sketching, or to enforce hard assumptions like positive definiteness or commutativity of $\mathbf{H}$ and $\mathbf{V}$ as we did in Section 5.3.1. We also plan to observe numerically the spectrum of $\mathbf{\Psi}$ in case where it is not too costly to compute to see if this assumption is met or not.

## 5.4 Sketched ADI method for Sylvester matrix equations

Now we extend our randomization of the Peaceman-Rachford method to the ADI method for solving Sylvester matrix equations. We first present the general assumptions made on Sylvester matrix equations and develop rapidly how they are related to the convergence of the general ADI method. Then, we present our sketch-and-project version of the ADI method, that we call the *Sketched ADI (SADI)* method. Like the SPR method, we give an implementation in the special case of row block subsampling.

As we have seen in Lemma 5.1, Sylvester equations are equivalent to PR problems. Thus the convergence analysis developed in Section 5.3 also applies to our SADI method.

### 5.4.1 Solution to the Sylvester equation and ADI error

Before introducing our new method, let us first discuss the existence of a solution $\mathbf{X}^* \in \mathbb{R}^{n\times n}$ to the Sylvester equation and the convergence of the classical ADI method.

**Proposition 5.14** (Solution of a Sylvester equation)**.** *Let* $\mathbf{A}, \mathbf{B}$ *and* $\mathbf{F} \in \mathbb{R}^{n\times n}$*. There exists a unique solution* $u^*$ *to the Sylvester matrix equation* (5.7) *that we recall here:*

$$\mathbf{AX} - \mathbf{XB} = \mathbf{F} \ ,$$

*if and only if* $\mathbf{A}$ *and* $\mathbf{B}$ *have no common eigenvalue,* i.e., *their spectra are disjoint:* $\sigma(\mathbf{A}) \cap \sigma(\mathbf{B}) = \emptyset$*.*

*Proof.* We recall that the Sylvester equation can be equivalently written through (5.8) and (5.9) in a linear system of the form

$$(\mathbf{H} + \mathbf{V})\, u = s \ ,$$

where $\mathbf{H} = \mathbf{I}_n \otimes \mathbf{A}$, $\mathbf{V} = -\mathbf{B}^\top \otimes \mathbf{I}_n$, $u = \mathrm{vec}(\mathbf{X})$ and $s = \mathrm{vec}(\mathbf{F})$. Thus the dimension $d$ of previous sections equals $n^2$. Hence, the Sylvester equation has a unique solution if and only if $0 \notin \sigma(\mathbf{H} + \mathbf{V}) = \sigma\left(\mathbf{I}_n \otimes \mathbf{A} - \mathbf{B}^\top \otimes \mathbf{I}_n\right)$. Theorem 13.16 of [84] directly implies that the latter condition is equivalent to $\mathbf{A}$ and $\mathbf{B}$ not having common eigenvalues, which can be rephrased as "the spectrum of $\mathbf{A}$ and $\mathbf{B}$ are disjoint". □

Hence, we see that there exist a solution to (5.7) if spectra of $\mathbf{A}$ and $-\mathbf{B}$ are separated. Moreover, there is a link between how separated are these spectra and ADI efficiency: the more these spectra are separated, the easier the problem is to solve.

Indeed, let $\mathbf{X}^*$ be a solution of the Sylvester matrix equation (5.7). Let $\mathbf{A}$ and $\mathbf{B}$ be normal matrices and suppose $E, G$ are two sets such that $\sigma(\mathbf{A}) \subset E$ and $\sigma(\mathbf{B}) \subset G$. Then Theorem 2 of [162] states that after $K$ steps of ADI method (with specific shift parameters) described in (5.10)–(5.11) the ADI error is given by

$$\left\|\mathbf{X}^{(K)} - \mathbf{X}^*\right\|_2 \leq Z_K(E,G)\, \|\mathbf{X}^*\|_2 \ , \tag{5.53}$$

where $Z_K(E,G)$ is the $K$-th Zolotarev number defined by

$$Z_K(E,G) \stackrel{\text{def}}{=} \inf_{r\in\mathcal{R}^K} \frac{\sup_{z\in E} |r(z)|}{\inf_{z\in G} |r(z)|} \ ,$$

where $\mathcal{R}^K$ is the space of rational functions where both the numerator and the denominator of degree less or equal than $K$. Hence, the separation between $\sigma(\mathbf{A})$ and $\sigma(-\mathbf{B})$ is a key information because Zolotarev numbers are known to decrease when $E$ and $G$ are more and more separated. Also, the upper-bound on the error in (5.53) allows to choose efficient shift parameters to "zeros and poles of the Zolotarev rational function associated with $Z_K(E,G)$" and thus reduce even more the error. In our experiments in Section 5.5, we will compare our new sketched ADI algorithms with state of the art methods which makes use of these "optimal shift parameters", which have the disadvantage of requiring the entire spectral information of $\mathbf{A}$ and $\mathbf{B}$.

In this thesis, we do not go further into the detail of the study of the Zolotarev function and this type of convergence theory for ADI methods. For further information, we refer to the introduction of [162], where the author clearly unfolds the link between Zolotarev number and the convergence of ADI. Section 1.6 of [162] gives also a good overview of when a Sylvester problem is "ADI-friendly".

### 5.4.2 Sketch-and-project ADI method

In this section, we apply exactly the same reasoning as in Section 5.2.1 and apply the sketch-and-project method to each half-step of the ADI algorithm. The only difference between the PR and the ADI method is that instead of solving a linear system, here we solve a matrix equation (which is indeed equivalent to a large linear system as pointed out in Lemma 5.1).

**Sketched ADI method.** In what follows, we describe our new method that we call *Sketched Alternating-Direction Implicit (SADI)*[6] *method*. For every $\mathbf{X} \in \mathbb{R}^{n\times n}$ and $\mathbf{M} \in \mathbb{R}^{n\times n}$ be a SPD matrix and let $\|\mathbf{X}\|^2_{F(\mathbf{M})} := \left\|\mathbf{M}^{1/2}\mathbf{X}\mathbf{M}^{1/2}\right\|_F^2 = \mathbf{Tr}\left(\mathbf{X}^\top\mathbf{M}\mathbf{X}\mathbf{M}\right)$ denote the weighted Frobenius norm. Let $\mathbf{M_A}, \mathbf{M_B} \in \mathbb{R}^{n\times n}$ be two given SPD matrices and let $\mathbf{X}^*$ denote the solution to the Sylvester equation (5.7).

Let $\tau \in \mathbb{N}$ be the sketch size and let $\mathcal{D}$ be a distribution over matrices $\mathbb{R}^{n\times\tau}$. At the $k$–th iteration, The sketched-and-project version of the ADI method described in (5.10)–(5.11) consists in sampling two sketching matrices $\mathbf{S}_{1/2}, \mathbf{S}_1 \sim \mathcal{D}$ and applying the following updates:

$$\begin{aligned}
\mathbf{X}^{(k+1/2)} &= \underset{\mathbf{X}\in\mathbb{R}^{n\times n}}{\arg\min} \left\|\mathbf{X} - \mathbf{X}^{(k)}\right\|_{\mathbf{M_A}} \\
&\qquad \text{subject to} \quad \mathbf{S}_{1/2}^\top(\mathbf{A} - p_k\mathbf{I}_n)\mathbf{X} = \mathbf{S}_{1/2}^\top\mathbf{X}^{(k)}(\mathbf{B} - p_k\mathbf{I}_n) + \mathbf{S}_{1/2}^\top\mathbf{F} \ , \qquad (5.54)\\
\mathbf{X}^{(k+1)} &= \underset{\mathbf{X}\in\mathbb{R}^{n\times n}}{\arg\min} \left\|\mathbf{X} - \mathbf{X}^{(k+1/2)}\right\|_{\mathbf{M_B}} \\
&\qquad \text{subject to} \quad \mathbf{X}(\mathbf{B} - q_k\mathbf{I}_n)\mathbf{S}_1 = (\mathbf{A} - q_k\mathbf{I}_n)\mathbf{X}^{(k+1/2)}\mathbf{S}_1 - \mathbf{F}\mathbf{S}_1 \ . \qquad (5.55)
\end{aligned}$$

Note that in (5.54) we sketch on the left and in (5.55) we sketch on the right.

The closed form solutions to the above half-steps are given by Theorem 1 in [60]. The one to (5.54) is given by

$$\mathbf{X}^{(k+1/2)} = \mathbf{X}^{(k)} - \mathbf{M}_\mathbf{A}^{-1}\mathbf{Z}_\mathbf{A}(\mathbf{A} - p_k\mathbf{I}_n)^{-1}R\left(\mathbf{X}^{(k)}\right) \ , \qquad (5.56)$$

and the solution to (5.55) is given by

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k+1/2)} + R\left(\mathbf{X}^{(k+1/2)}\right)(\mathbf{B} - q_k\mathbf{I}_n)^{-1}\mathbf{Z}_\mathbf{B}\mathbf{M}_\mathbf{B}^{-1} \ , \qquad (5.57)$$

where

$$\begin{aligned}
\mathbf{Z}_\mathbf{A} &\overset{\text{def}}{=} (\mathbf{A} - p_k\mathbf{I}_n)^\top\mathbf{S}_{1/2}(\mathbf{S}_{1/2}^\top(\mathbf{A} - p_k\mathbf{I}_n)\mathbf{M}_\mathbf{A}^{-1}(\mathbf{A} - p_k\mathbf{I}_n)^\top\mathbf{S}_{1/2})^\dagger\mathbf{S}_{1/2}^\top(\mathbf{A} - p_k\mathbf{I}_n) \ , \qquad (5.58)\\
\mathbf{Z}_\mathbf{B} &\overset{\text{def}}{=} (\mathbf{B} - q_k\mathbf{I}_n)\mathbf{S}_1(\mathbf{S}_1^\top(\mathbf{B} - q_k\mathbf{I}_n)^\top\mathbf{M}_\mathbf{B}^{-1}(\mathbf{B} - q_k\mathbf{I}_n)\mathbf{S}_1)^\dagger\mathbf{S}_1^\top(\mathbf{B} - q_k\mathbf{I}_n)^\top \ , \qquad (5.59)
\end{aligned}$$

and where the residual of the Sylvester matrix equation is given by

$$R(\mathbf{X}) \overset{\text{def}}{=} \mathbf{A}\mathbf{X} - \mathbf{X}\mathbf{B} - \mathbf{F} = \mathbf{A}(\mathbf{X} - \mathbf{X}^*) - (\mathbf{X} - \mathbf{X}^*)\mathbf{B} \ . \qquad (5.60)$$

In Algorithm 16, we provide the implementation of the SADI method.

**Remark 5.15.** *Again, matrices $\mathbf{Z_H}$ and $\mathbf{Z_V}$ can depend on the shift parameters if they are not constant.*

**Remark 5.16** (No natural weight matrices)**.** *Note that due to the alternate structure of ADI methods, we face the same issue than the for PR problem: there are no natural weight matrices $\mathbf{M_A}, \mathbf{M_B}$ that lead to useful simplifications.*

**Remark 5.17** (Difficult update of the residual)**.** *As we have seen in Section 5.2.1, the problem of Algorithm 16 is to update the residual. Unlike the efficient update* (4.13) *in Chapter 4 used for RidgeSketch, there is no clear way to compute $R(\mathbf{X}^{(k)})$ on the fly in order to avoid large matrix-matrix multiplications. Indeed, computing the residual $R(\mathbf{X}^{(k)}) = \mathbf{A}\mathbf{X}^{(k)} - \mathbf{X}^{(k)}\mathbf{B} - \mathbf{F} \in \mathbb{R}^{n\times n}$ from scratch costs $\mathcal{O}(n^3)$, which is computationally cumbersome. But in fact, we are only interested in the sketched residual. For instance at the first half-step, we need $\mathbf{R}_\mathbf{S}^{(k)} = \mathbf{S}_{1/2}^\top R\left(\mathbf{X}^{(k)}\right) \in \mathbb{R}^{\tau\times n}$, which requires $\mathcal{O}(\tau n^2)$ operations if subsampling is performed before matrix-matrix multiplications, where $\tau \ll n$.*

[6] In reference to the great persian poet Saadi Shirazi.

**Algorithm 16** Sketched ADI (SADI) method

**Input:** shift parameters $(p_k)_k, (q_k)_k$, weight matrices $\mathbf{M_A}, \mathbf{M_B}$, sketching distribution $\mathcal{D}$
**Initialize:** $\mathbf{X}^{(0)} = 0_{n\times n}$
**for** $k = 0, 1, \ldots, K-1$ **do**
  Sample $\mathbf{S}_{1/2}, \mathbf{S}_1 \sim \mathcal{D}$ ▷ sketching matrices
  $\mathbf{R}_{\mathbf{S}}^{(k)} = \mathbf{S}_{1/2}^\top R\left(\mathbf{X}^{(k)}\right)$ ▷ 1st sketched residual
  $\mathbf{Y}^{(k)} = \texttt{least\_norm\_solution}\left(\mathbf{S}_{1/2}^\top(\mathbf{A} - p_k\mathbf{I}_n)\mathbf{M_A}(\mathbf{A} - p_k\mathbf{I}_n)^\top\mathbf{S}_{1/2},\, \mathbf{R}_{\mathbf{S}}^{(k)}\right)$
  $\mathbf{X}^{(k+1/2)} = \mathbf{X}^{(k)} - \mathbf{M_A}(\mathbf{A} - p_k\mathbf{I}_n)^\top\mathbf{S}_{1/2}\mathbf{Y}^{(k)}$ ▷ 1st half-step
  $\mathbf{R}_{\mathbf{S}}^{(k+1/2)} = \mathbf{S}_1^\top R\left(\mathbf{X}^{(k+1/2)}\right)^\top$ ▷ 2nd sketched residual
  $\mathbf{Y}^{(k+1/2)} = \texttt{least\_norm\_solution}\left(\mathbf{S}_1^\top(\mathbf{B} - q_k\mathbf{I}_n)^\top\mathbf{M_B}(\mathbf{B} - q_k\mathbf{I}_n)\mathbf{S}_1,\, \mathbf{R}_{\mathbf{S}}^{(k+1/2)}\right)^\top$
  $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k+1/2)} + \mathbf{Y}^{(k+1/2)}\mathbf{S}_1^\top(\mathbf{B} - q_k\mathbf{I}_n)^\top\mathbf{M_B}$ ▷ 2nd half-step
**Output:** $\mathbf{X}^{(K)}$

### 5.4.3 SADI with block subsampling sketching

Here we give a practical example of a simple implementation of SADI. We use block rows subsampling for both half-steps, that is the sketch matrices are concatenations of randomly sampled columns of the identity matrix: $\mathbf{S}_{1/2} = (\mathbf{I}_n)_{:C_{1/2}} \in \mathbb{R}^{n\times\tau}$ and $\mathbf{S}_1 = (\mathbf{I}_n)_{:C_1} \in \mathbb{R}^{n\times\tau}$. In what follows, we note $\mathbf{I}_{C_{1/2}:} \overset{\text{def}}{=} \mathbf{S}_{1/2}^\top\mathbf{I}_n = (\mathbf{I}_n)_{C_{1/2}:} \in \mathbb{R}^{\tau\times n}$ and $\mathbf{I}_{:C_1} \overset{\text{def}}{=} \mathbf{I}_n\mathbf{S}_1 = (\mathbf{I}_n)_{:C_1} \in \mathbb{R}^{n\times\tau}$ to simplify notations. Note that the structure of the Sylvester equation (5.7) has a direct impact on the sketching procedure. Indeed, in the first half-step, sketching acts on the rows of $\mathbf{A} + p_k\mathbf{I}_n$ because $\mathbf{A}$ multiplies $\mathbf{X}$ on the left. On the contrary, in the second half-step, sketching acts on the columns of $\mathbf{B}+q_k\mathbf{I}_n$ because $\mathbf{B}$ multiplies $\mathbf{X}$ on the right. The resulting method that we call *Block Sketched ADI (BSADI)* method is given in Algorithm 17.

**Algorithm 17** Block Sketched ADI (BSADI) method

**Input:** shift parameters $(p_k)_k, (q_k)_k$, sketch size $\tau$
**Initialize:** $\mathbf{X}^{(0)} = 0_{n\times n}$
**for** $k = 0, 1, \ldots, K-1$ **do**
  Sample $C_{1/2}, C_1 \subset [n]$ s.t. $|C_{1/2}| = |C_1| = \tau$ ▷ sampled row indices
  $\mathbf{R}_{\mathbf{S}}^{(k)} = \mathbf{A}_{C_{1/2}:}\mathbf{X}^{(k)} - \mathbf{X}_{C_{1/2}:}^{(k)}\mathbf{B} - \mathbf{F}_{C_{1/2}:}$ ▷ 1st sketched residual
  $\tilde{\mathbf{A}} = \mathbf{A}_{C_{1/2}:} - p_k\mathbf{I}_{C_{1/2}:}$ ▷ sketched shifted $\mathbf{A}$
  $\mathbf{X}^{(k+1/2)} = \mathbf{X}^{(k)} + (\tilde{\mathbf{A}})^\top\left(\tilde{\mathbf{A}}(\tilde{\mathbf{A}})^\top\right)^\dagger\mathbf{R}_{\mathbf{S}}^{(k)}$ ▷ 1st half-step
  $\mathbf{R}_{\mathbf{S}}^{(k+1/2)} = \mathbf{A}\mathbf{X}_{:C_1}^{(k+1/2)} - \mathbf{X}(k+1/2)\mathbf{B}_{:C_1} - \mathbf{F}_{:C_1}$ ▷ 2nd sketched residual
  $\tilde{\mathbf{B}} = \mathbf{B}_{:C_1} - q_k\mathbf{I}_{:C_1}$ ▷ sketched shifted $\mathbf{B}$
  $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k+1/2)} - \mathbf{R}_{\mathbf{S}}^{(k+1/2)}\left((\tilde{\mathbf{B}})^\top\tilde{\mathbf{B}}\right)^\dagger(\tilde{\mathbf{B}})^\top$ ▷ 2nd half-step
**Output:** $\mathbf{X}^{(K)}$

## 5.5 Numerical experiments

In this section, we present an experimental setting to compare the performance of classical *Peaceman-Rachford (PR)* and *Alternating-Directions Implicit (ADI)* methods and our new randomized variants: *Sketched PR (SPR)* and *Sketched ADI (SADI)* methods. In all our experiments, the algorithms are run for a number of iterations such that the methods reach a desired precision $\epsilon > 0$. We use two metrics to monitor convergence: the relative residual

$$\frac{\left\|(\mathbf{H}+\mathbf{V})u^K - s\right\|_2^2}{\|(\mathbf{H}+\mathbf{V})u^0 - s\|_2^2} \quad \text{or} \quad \frac{\left\|\mathbf{A}\mathbf{X}^{(K)} - \mathbf{X}^{(K)}\mathbf{B} - \mathbf{F}\right\|_2^2}{\left\|\mathbf{A}\mathbf{X}^{(0)} - \mathbf{X}^{(0)}\mathbf{B} - \mathbf{F}\right\|_2^2} \ , \tag{5.61}$$

and the relative distance to the solution given by

$$\frac{\left\|u^K - u^*\right\|_2^2}{\|u^0 - u^*\|_2^2} \quad \text{or} \quad \frac{\left\|\mathbf{X}^{(K)} - \mathbf{X}^*\right\|_2^2}{\left\|\mathbf{X}^{(0)} - \mathbf{X}^*\right\|_2^2} \ , \tag{5.62}$$

even if in real applications one cannot use the latter metric since it requires the knowledge of the solution. We always start with initial iterates that are zeros, *i.e.,* $u^0 = 0_d$ and $\mathbf{X}^{(0)} = 0_{n\times n}$. All Python codes used to run the experiments presented hereafter are available at: `https://github.com/ngazagna/sadi`.

### 5.5.1 Solving Peaceman-Rachford problem

We start by comparing the performance of the PR method (5.2)–(5.3) with iteration-dependent shift parameters $(p_k)_k, (q_k)_k$ (two strategies are discussed hereafter) or with constant shifts $p = q$ and the Block SPR method described in Algorithm 15 with constant shifts. We simulate symmetric matrices $\mathbf{H}, \mathbf{V} \in \mathbb{R}^{d\times d}$ for which we control the eigenvalues such that we ensure the existence of $u^*$ since $\mathbf{H}+\mathbf{V}$ is also PSD. Our focus is set on the regime where $d \gg 1$ in order to show the usefulness of our randomization and sketching techniques, which are known to be well-suited for large scale settings. Different cases are studied depending on the spectrum of $\mathbf{H}$ and $\mathbf{V}$. When simulating these data matrices, we also tried different condition number without seeing any clear impact on the relative behaviors of the implemented algorithms.

**Simulated matrices.** Let us denote by $\lambda_{\min}(\mathbf{M})$ and $\lambda_{\max}(\mathbf{M})$ the smallest and the largest eigenvalues of a matrix $\mathbf{M}$. Using the same notations from Section 5.3.1, let $0 < a \overset{\text{def}}{=} \lambda_{\min}(\mathbf{H}) < b \overset{\text{def}}{=} \lambda_{\max}(\mathbf{H})$ and $\alpha \overset{\text{def}}{=} \lambda_{\min}(\mathbf{V}) < \beta \overset{\text{def}}{=} \lambda_{\max}(\mathbf{V})$. The simulated data we use in our experiments are designed to explore two different regimes. In the first one, $\sigma(\mathbf{H})$ and $\sigma(\mathbf{V})$ are both included in $\mathbb{R}_+^*$ and very close, *i.e.,* $a \approx \alpha > 0$ and $b \approx \beta$. In the second one, $\sigma(\mathbf{H}) \subset \mathbb{R}_+^*$ and $\sigma(\mathbf{V}) \subset \mathbb{R}_-^*$ such that $\mathbf{H}+\mathbf{V}$ is PSD. In both regimes, the eigenvalues of each matrix are exponentially decreasing. Details of implementation are available in the online code. Shifts parameters are built upon this spectral information and more precisely on bounds of $\sigma(\mathbf{H}) \cup \sigma(\mathbf{V})$. In what follows, let us denote $\tilde{a} \overset{\text{def}}{=} \min\{a, \alpha\}$ and $\tilde{b} \overset{\text{def}}{=} \max\{b, \beta\}$.

**Shift parameters and implemented algorithms.** For both the PR and the BSPR methods and in both regimes, we use constant shifts.

- *Constant shifts* are set to $p = q = \tilde{b}$ (which corresponds to the first (W) shift at $k = 0$). We use this rough shifting technique since it satisfies the theoretical lower bound on the shift parameters given in [18] for the PR method and in Theorem 5.10 for the SPR method[7].

For the PR method in the first regime where both $\mathbf{H}$ and $\mathbf{V}$ are SPD matrices, so $0 < \tilde{a} < \tilde{b}$, we also tried moving shifting settings given in Section 1.3 of [156]:

[7] This theorem was not extended to BSPR, yet all our experiments with constant shifts are stable and converge.

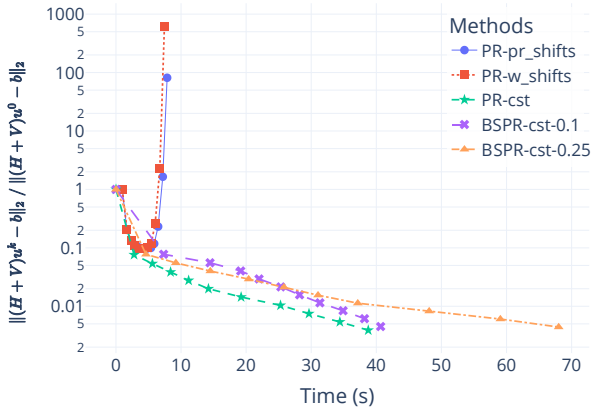



**Figure 5.1:** *Divergence of the PR method with PR (blue) and W (red) shift parameters given in* (5.63) *and* (5.64)*. Dimension* $d = 1000$ *and* $\sigma(\mathbf{H}) \subset [8, 1008]$ *and* $\sigma(\mathbf{V}) \subset [-100, -0.005]$*.*

- *Peaceman-Rachford's (PR) shifts* introduced in the seminal paper [113], such that for all $k\breve{a} \in [K-1]$

$$p_k = q_k = \tilde{b}\left(\frac{\tilde{a}}{\tilde{b}}\right)^{\frac{2k+1}{2K}} \quad . \tag{5.63}$$

- *Wachspress' (W) shifts* introduced in [157], such that for all $k\breve{a} \in [K-1]$

$$p_k = q_k = \tilde{b}\left(\frac{\tilde{a}}{\tilde{b}}\right)^{\frac{k}{K-1}} \quad . \tag{5.64}$$

Yet, these shifts parameters are not valid in the second regime as $\mathbf{V}$ is not SPD. In Figure 5.1, we see that PR and W shifts lead to divergent behaviors.

As explained above, for the constant shifts one only requires an estimate of the largest eigenvalue of $\mathbf{H}$ and $\mathbf{V}$, which can easily be done by applying few steps of the power method. Whereas, for PR and W shifts, one needs to compute both the (positive) smallest and the largest eigenvalues of $\mathbf{H}$ and of $\mathbf{V}$ which is often challenging when $a$ and $\alpha$ are closed to zero. One can compute them through a complete eigenvalue decomposition or by applying the inverse power method. However, computing the smallest eigenvalue of SPD if often way more costly than approximating its largest eigenvalue as we will see in time plots.

**Other parameters fine-tuning.** We optimize the number of iterations $K$ to run with PR and W shift parameters by applying a rough grid search. For instance, from Figure 5.2, we observe that for configuration ($d = 1000$ and close spectra), $K = 10$ is enough to reach a precision $\epsilon = 10^{-4}$. For PR and BSPR with constant shifts, we optimize the number of iterations the same way.
Another important parameter of BSPR is the *sketch fraction* of the data, which equals $\tau/d$ where we recall $\tau \in [d]$ is the sketch size. This is the percentage of the rows sampled at each half-step of BSPR. Obviously, if we set it to 100% we exactly recover PR with constant shifts. In all our plots, we show BSPR for multiple sketch fractions: 10% and 25%.

#### 5.5.1.1 Experiments for H and V with close and positive spectra

The first regime we study sticks to the assumptions made in Theorem 5.10, that is $\mathbf{H}$ and $\mathbf{V}$ are SPD matrices that commute and their eigenvalues are in $\mathbb{R}_+^*$ and very close. We simulate $\mathbf{H}$ and $\mathbf{V}$ by first computing two diagonal matrices containing exponentially decreasing eigenvalues. Then we generated a random rotation matrix and apply it on the right and on the left of both diagonals. That is, $\mathbf{H}$ and $\mathbf{V}$ are ensured to share a common eigenbasis and thus to commute.

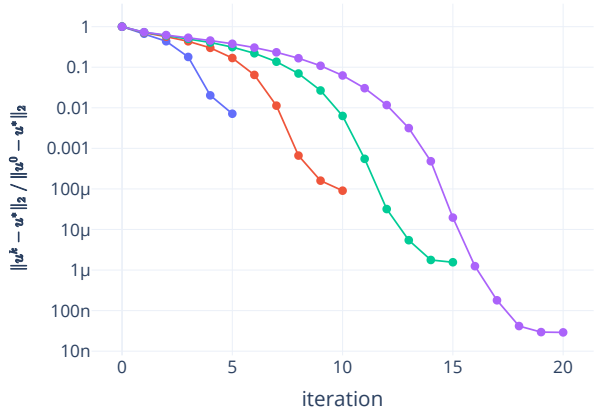



**Figure 5.2:** *Impact on the number of iterates on the convergence of the PR method with PR shift parameters given in* (5.63)*. Dimension* $d = 1000$ *and spectra are such that* $\sigma(\mathbf{H}) \cup \sigma(\mathbf{V}) \subset [0.0001, 1.001]$

In Figure 5.3, we clearly observe the extremely fast convergence of the PR method versus the BSPR method as expected. Indeed, the PR and W shift parameters are optimized to accelerate the convergence by exploiting the spectral bounds $\tilde{a}$ and $\tilde{b}$.
Yet, we can also see that the initialization step computing the spectral bounds and then the moving shifts from (5.63) and (5.64) requires much more time than the one estimating roughly an approximation of $\tilde{b}$ by applying 10 steps of the power method. In the very large scale setting, computing a good approximation of $\tilde{a}$ is intractable which justifies the use of constant shifts $p = q \approx \tilde{b}$. Moreover, for $d = 5000$ one can already notice in the right-hand side plot of Figure 5.3b that BSPR turns out to be faster than the PR method with constant shifts for reaching low precision solutions. This phenomenon justifies the use of randomized methods like BSPR when matrices have millions of rows.

#### 5.5.1.2 Experiments for H with positive and V with negative spectrum

The second regime deals with $\mathbf{H}$ being a positive definite matrix and $\mathbf{V}$ a negative definite one. Both $\mathbf{H}$ and $\mathbf{V}$ are symmetric and commute. We simulate these matrices the same way we do in Section 5.5.1.1. As we explained at the beginning of this section, when $\sigma(\mathbf{V}) \subset \mathbb{R}_-^*$ the moving shift parameters $(p_k)_k$ and $(q_k)_k$ given (5.63)–(5.64) are not valid anymore. Even worse, they lead to divergence of the PR method as showed in Figure 5.1. This is the reason why we only compare here the PR and the BSPR with constant shifts.

Figure 5.4 shows our results for $d = 1000$ and $d = 5000$. As dimension $d$ increases the sketch-and-project version outperforms the classical PR method. Yet, the choice of the sketch size $\tau \in [d]$ is important to leverage randomization and adds a hyperparameter.

### 5.5.2 Solving Sylvester matrix equations

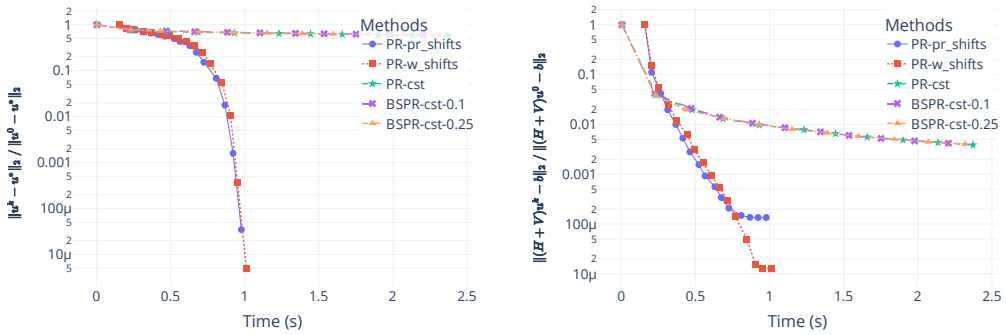
In what follows we only provide relative residual plots since we did not observe any different behavior as compared to plots displaying relative distance to the solution.

#### 5.5.2.1 Solving the Poisson equation with Chebyshev differentiation matrices

Here we apply our method to a real problem already discussed in the introduction in Section 5.1: the discretized Poisson equation. We recall the corresponding Sylvester equation, already given in (5.12):

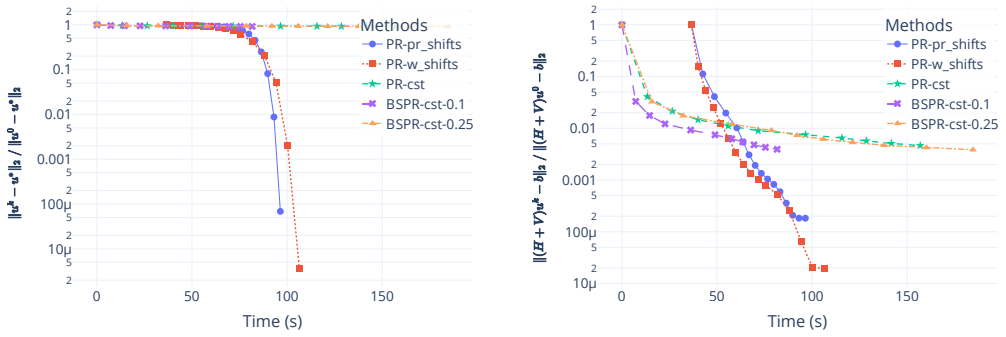
$$\mathbf{D}_2\mathbf{X} + \mathbf{X}\mathbf{D}_2^T = \mathbf{F} \ , \tag{5.65}$$

where $\mathbf{D}_2 \in \mathbb{R}^{n\times n}$ is a second-order differentiation matrix, $\mathbf{F} \in \mathbb{R}^{n\times n}$ is the function $f(x, y) = \cos(10x)\sin(10y)$ applied at the grid points selected in $[-1, 1]$. We also recall that based on previous




*Relative distance to optimum* *Relative residual*

**(a)** $d = 1000$




*Relative distance to optimum* *Relative residual*

**(b)** $d = 5000$

**Figure 5.3:** *Comparison of the PR method with PR (blue), W (red) and constant shift (green) parameters and the BSPR method with constant shifts. Sketch sizes* $\tau = 10\%$ *(purple) and* $25\%$ *(orange) of* $d$*. Spectra are such that* $\sigma(\mathbf{H}) \cup \sigma(\mathbf{V}) \subset [0.0001, 1.001]$.

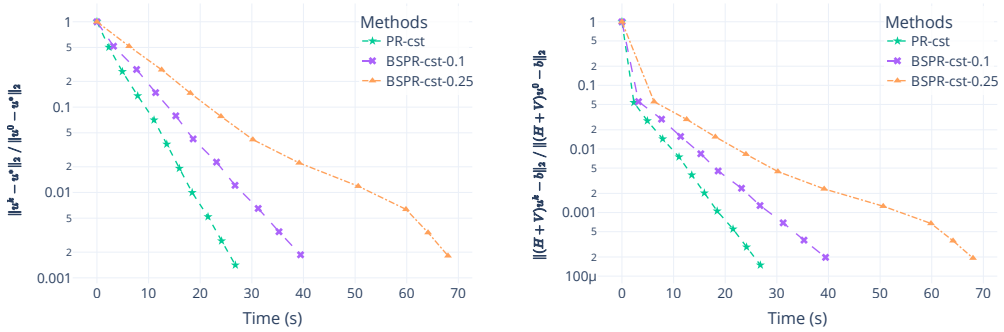
notations, here $\mathbf{A} \overset{\text{def}}{=} \mathbf{D}_2$ and $\mathbf{B} \overset{\text{def}}{=} -\mathbf{D}_2^\top$.

In this experiment, we aim at solving large scale dense Sylvester matrix equations, this is why, we chose $\mathbf{D}_2$ to be a Chebyshev differentiation matrix instead of a tridiagonal finite difference matrix. We built such matrices and the corresponding Chebyshev grid points[8], that are $x_j, y_j \overset{\text{def}}{=} \cos\left(\frac{j\pi}{N}\right)$ for $j \in \{0, \dots, n\}$, using results from chapter 8 of [146]. Theory indicates that $\mathbf{D}_2 \preccurlyeq 0$, and thus spectra of $\mathbf{A}$ and $\mathbf{B}$ are separated: $\sigma(\mathbf{D}_2) \cap \sigma\left(-\mathbf{D}_2^\top\right) = \emptyset$. As explained in Section 5.4.1, this property makes this problem "ADI-friendly" as the more separated spectra are, the faster is the convergence of ADI methods.

Let $a \overset{\text{def}}{=} \lambda_{\min}(\mathbf{A}) < b \overset{\text{def}}{=} \lambda_{\max}(\mathbf{D}_2) < 0$ and $c \overset{\text{def}}{=} \lambda_{\min}(\mathbf{B}) < d \overset{\text{def}}{=} \lambda_{\max}(\mathbf{B})$. We then have that $d = -a > 0$ and $c = -b > 0$. Theory in Section 6 of [161] tells us that when $N$ is large, $c \approx \frac{\pi^2}{4}$ and $d = \mathcal{O}(n^4)$.

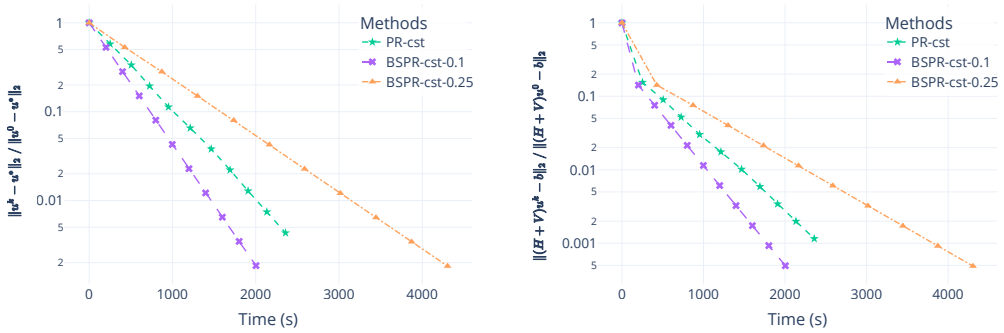
**Filtered Chebyshev differentiation matrix.** Because second-order Chebyshev differentiation matrix of size $n \times n$ has a condition number $\mathcal{O}(n^4)$, we also decided to compare our methods on a *filtered version* of the problem. That is, we reduce the magnitude of the largest eigenvalues of $\mathbf{D}_2$, according to Section 5 of [161]. To filter $\mathbf{D}_2$ we first compute its eigenvalue decomposition, then for all $k \geq \frac{2N}{\pi}$ we replace the $k$–th eigenvalue by $-\frac{k^2\pi^2}{4}$, which actually is the $k$–th eigenvalue of the continuous second derivative operator. In Figure 5.5, we reproduced Figure 1(b) from [161] to ensure that our filtering method was correct.

[8] We adapted code from `https://github.com/nikola-m/another-chebpy/blob/master/chebPy.py`.




*Relative distance to optimum* *Relative residual*

**(a)** $d = 1000$




*Relative distance to optimum* *Relative residual*

**(b)** $d = 5000$

**Figure 5.4:** *Comparison of the PR (green) and the BSPR method with constant shifts. Sketch sizes* $\tau = 10\%$ *(purple) and* $25\%$ *(orange) of* $d$*. Spectra are such that* $\sigma(\mathbf{H}) \subset [8, 1008]$ *and* $\sigma(\mathbf{V}) \subset [-100, -0.005]$*.*

**Compared algorithms and shifting strategies.** For our *BSADI method* presented in Algorithm 17 we use constant shift $p = d$ and $q = a$. Such values are also used for the classic "full-batch" *ADI method* and in all our experiments always led to converging behaviors.

We compute what we call the *Mobius shifts*[9] $(p_k)_{1\leq k\leq K_\epsilon}, (q_k)_{1\leq k\leq K_\epsilon}$ from Appendix A of [43] which are optimal shift parameters designed for real and disjoint spectra such that $a < b < c < d$. The latter shifts depend on the desired tolerance $\epsilon$ (set to $10^{-3}$ in all our experiments) and indicate in the mean time the number of iterations $K_\epsilon$ to run to reach an $\epsilon$–error. Finally, we also use *constant 1-step Mobius shifts* which are constant across iterations and are computed using a modified version of the code in Appendix A of [43] for which we enforce the number of iteration to be 1. The latter are showed to measure the relative efficiency of ADI with constant shift $p = d$ and $q = a$.

The only pitfall of Mobius shifts is that it requires to have separated spectra and precise knowledge of their upper and lower bounds, whereas constant shifts only require a lower (resp. upper) bound of the spectrum of $\mathbf{A}$ (resp. $\mathbf{B}$) which can be easily be estimated by power iteration.

**Results.** Our results on the original Poisson equation are given in Figure 5.7 and the one with filtered Chebyshev matrices are given in Figure 5.6. Both Figure 5.6 and Figure 5.7 clearly show the rapid convergence in few iterations of ADI with Mobius shifts (red). These figures also highlight the importance of the value of constant shift parameters for the classic ADI. When $n = 100$, in both unfiltered and filtered cases, constant shifts $p = d$ and $q = a$ (green) do not lead to the fast convergence given by

[9] Because of the use of a Mobius transformation.

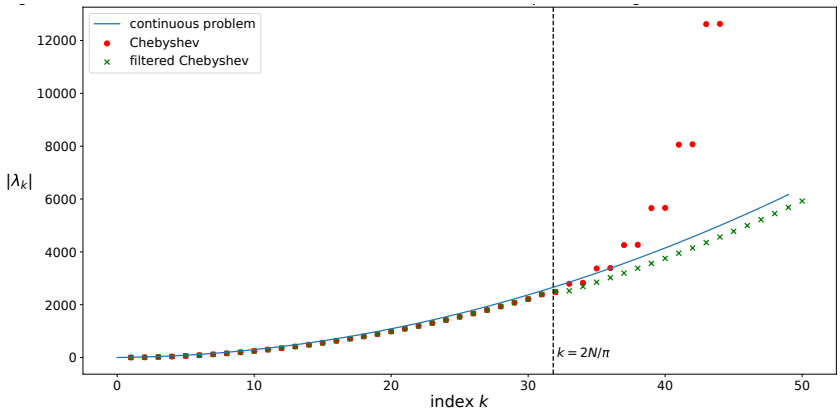
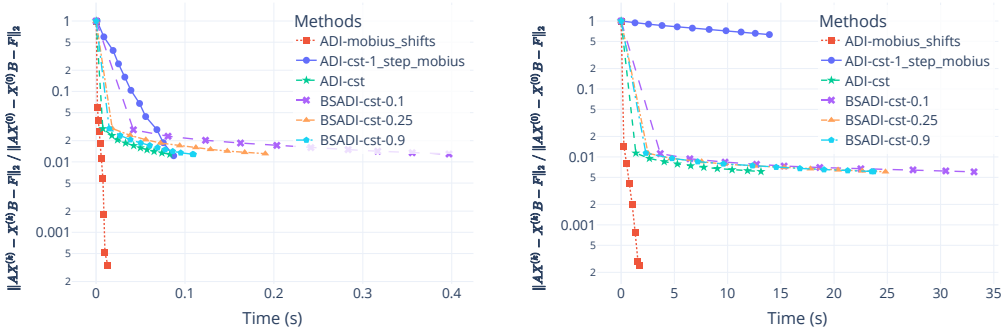



**Figure 5.5:** *Comparison between the absolute value of the eigenvalues of the second-order Chebyshev differentiation matrix, of its filtered version and of its continuous differentiation operator ($n = 50$).*




**(a)** $n = 100$ **(b)** $n = 1000$

**Figure 5.6:** *Comparison of the ADI method with Mobius (red), Mobius 1-step (navy) and constant (green) shifts with the BSADI method with constant shifts applied to the Poisson equation problem. Sketch sizes $\tau = 10\%$ (purple) and $25\%$ (orange) and $90\%$ (turquoise) of $n$.*

constant *Mobius 1-step shifts* (navy). Yet, as dimension $n$ increases, *Mobius 1-step shifts* become very inefficient.

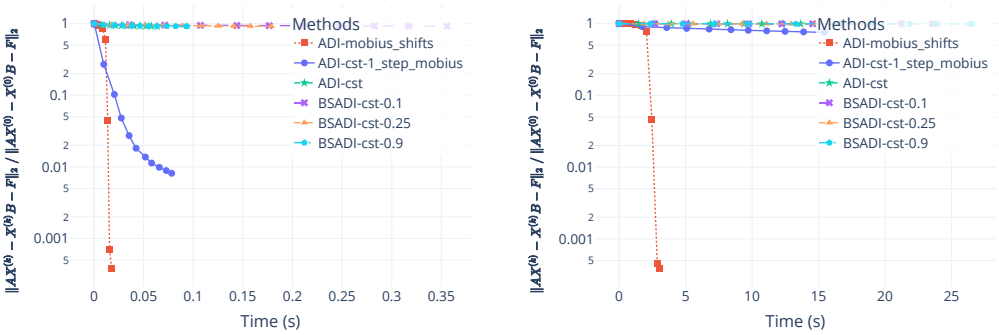
Unfortunately, in either case, even when increasing dimension up to $n = 8000$ we could not observe benefits from using BSADI (either with large or small sketch size) compared to ADI with constant shifts.

#### 5.5.2.2 LANPRO matrices

We also tried to compare the same methods on another problem with sparse real symmetric matrices NOS6 and NOS5 taken from set *LANPRO* of the Harwell-Boeing Collection[10]. We designed this experiments exactly like in Example 6.3 of [15], that is: $\mathbf{A}$ equals NOS6 and of order 675 and $\mathbf{B}$ equals minus NOS5 and of order 468. The right-hand side term $\mathbf{F}$ of the Sylvester equation is randomly generated (more details are provided in the code).

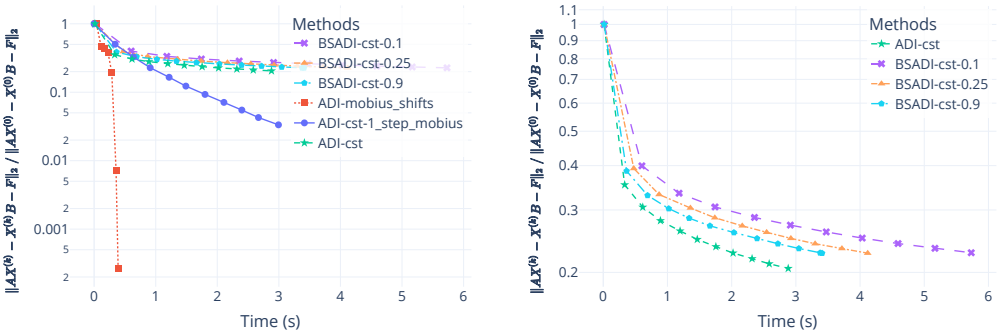
Our results are presented in Figure 5.8 and do not show neither any benefit from sketch-and-projecting the ADI method.

[10] `https://math.nist.gov/MatrixMarket/data/Harwell-Boeing/lanpro/lanpro.html`.




**(a)** $n = 100$ **(b)** $n = 1000$

**Figure 5.7:** *Comparison of the ADI method with Mobius (red), Mobius 1-step (navy) and constant (green) shifts with the BSADI method with constant shifts applied to the Poisson equation problem with filtered Chebyshev second-order differentiation matrices. Sketch sizes* $\tau = 10\%$ *(purple) and* $25\%$ *(orange) and* $90\%$ *(turquoise) of* $n$*.*




**(a)** *All methods* **(b)** *Zoom*

**Figure 5.8:** *Comparison of the ADI method with Mobius (red), Mobius 1-step (navy) and constant (green) shifts with the BSADI method with constant shifts applied to LANPRO matrices. Sketch sizes* $\tau = 10\%$ *(purple) and* $25\%$ *(orange) and* $90\%$ *(turquoise) of* $n$*.*

## 5.6 Conclusion and future work

This chapter aimed at paving the way for randomized sketching methods to solve ADI model problems like Peaceman-Rachford and Sylvester equations. Even if our theoretical results are very limited, since in Theorem 5.10 we only proved the convergence in expectation of our *SPR method*, our numerical experiments in Section 5.5 tend to show that one could benefit from randomization of the *PR method* with constant shift parameters in large scale settings.

Unfortunately, we were not able to highlight benefits of the *BSADI method* compared to classic ADI with constant shifts. We believe that sketched ADI methods with constant shifts could be useful for ill problems, like in Figure 5.1, where spectra of $\mathbf{A}$ and $\mathbf{B}$ are not separated and for which classic ADI theory is not as advanced. This is why, in additional experiments not displayed here[11], we generated matrices $\mathbf{A}$ and $\mathbf{B}$ for which we could control their spectrum (close or interlaced spectra). However, nor could we exhibit a case where BSADI was faster, either in time or epoch, than "full-batch" ADI.

Our intuition is that improvements could be done in our implementation of our BSADI method, which is far from being optimal. For instance, we could try to solve approximately each sketched linear system in each half-step. We also think that even if block row and column subsampling/slicing are quick operations

[11] But available in the provided code at: `https://github.com/ngazagna/sadi`.

in Python, they do not extract and compress enough information out of the matrices. Thus, one could try other type of sketching methods as *Count* or *SubCount sketch* used in Chapter 4.

On the theoretical side, we observed that convergence of ADI methods is very sensitive to the choice of the shift parameters $(p_k)_k$ and $(q_k)_k$ as studied in [14]. In order to make *SPR* and *SADI* methods competitive with classical PR and ADI methods, one should look for a theory that directly analyses the convergence from $u^0$ to $u^K$, that is

$$\mathbb{E}\left[\left\|u^K - u^*\right\|_2^2\right] \leq \rho(p_1, \ldots, p_K, q_1, \ldots, q_K)\ \mathbb{E}\left[\left\|u^0 - u^*\right\|_2^2\right] \quad . \tag{5.66}$$

Then, one could solve, even approximately, a new *minimax problem* like in (5.29) and look for good shift parameters that are robust to randomization of our sketched ADI methods.

## 5.7 Linear algebra tools

In this section we keep track of well-known mathematical tools needed in our analysis.

**Lemma 5.18.** *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *two matrices of order* $n$*. If* $\mathbf{A}$ *and* $\mathbf{B}$ *are symmetric and commute then* $\mathbf{AB}$ *is symmetric too.*

*Proof.* The proof comes from a forward computation

$$(\mathbf{AB})^\top = \mathbf{B}^\top \mathbf{A}^\top \overset{\text{symmetric } \mathbf{A} \text{ and } \mathbf{B}}{=} \mathbf{BA} \overset{\mathbf{A} \text{ and } \mathbf{B} \text{ commute}}{=} \mathbf{AB} \ .$$

□

**Definition 5.19** (Spectral radius)**.** *Let* $\mathbf{A}$ *an* $n \times n$ *matrix which eigenvalues are denoted* $(\lambda_i)_{i=1}^n$*. The* spectral radius *of the matrix* $\mathbf{A}$ *is defined by*

$$\rho(\mathbf{A}) \overset{def}{=} \max_{i=1,\ldots,n} |\lambda_i| \quad . \tag{5.67}$$

**Definition 5.20** (Operator norm)**.** *Let* $\mathbf{A}$ *an* $n \times n$ *matrix. The* operator norm *of the matrix* $\mathbf{A}$ *is defined by*

$$\|\mathbf{A}\| \overset{def}{=} \sup_{x \neq 0} \frac{\|\mathbf{A}x\|}{\|x\|} \quad , \tag{5.68}$$

*which is often called the* largest singular value *of* $\mathbf{A}$*.*

**Lemma 5.21.** *Let* $\mathbf{A}$ *an* $n \times n$ *matrix, then*

$$\|\mathbf{A}\| = \sqrt{\rho(\mathbf{A}^\top \mathbf{A})} \ . \tag{5.69}$$

*Proof.* See proof of Theorem 1.7 in [153]. □

**Lemma 5.22.** *For an arbitrary matrix* $\mathbf{A}$ *of order* $n$*,*

$$\rho(\mathbf{A}) \leq \|\mathbf{A}\| \quad . \tag{5.70}$$

*Moreover, if* $\mathbf{A}$ *is symmetric, then this inequality becomes an equality.*

*Proof.* Let $\lambda$ be an eigenvalue of $\mathbf{A}$ and $v$ an nonzero eigenvector associated with $\lambda$. We then get $\mathbf{A}v = \lambda v$, which implies

$$|\lambda| \,.\, \|v\| = \|\lambda v\| = \|\mathbf{A}v\| \leq \|\mathbf{A}\| \,.\, \|v\| \quad ,$$

by taking the maximum over all the eigenvalues we prove the desired inequality.

Now, let us assume that $\mathbf{A}$ is symmetric, then

$$\|\mathbf{A}\|^2 \overset{\text{Lemma 5.21}}{=} \rho(\mathbf{A}^\top \mathbf{A}) = \rho(\mathbf{A}^2) = \rho(\mathbf{A})^2 \ ,$$

where the last equality comes from the fact that the spectrum of $\mathbf{A}^2$ is the squared spectrum of $\mathbf{A}$. This ends the proof of the equality. □

**Definition 5.23** (Frobenius norm)**.** *Let $\mathbf{A}$ an $m \times n$ matrix. The Frobenius (or Hilbert-Schmidt) norm is defined in at least two ways*

$$\|\mathbf{A}\|_F \overset{def}{=} \sqrt{\mathbf{Tr}\left(\mathbf{A}^\top \mathbf{A}\right)} = \sqrt{\sum_{i=1}^{\min\{m,n\}} s_i^2(\mathbf{A})} \ , \tag{5.71}$$

*where $\mathbf{Tr}\left(.\right)$ denotes the trace function which returns the sum of the diagonal entries of a squared matrix, and the $s_i(\mathbf{A})$ are the singular values of $\mathbf{A}$.*

**Lemma 5.24** (Singular values of symmetric positive semidefinite matrices match eigenvalues)**.** *Let $\mathbf{A}$ be an $n \times n$ symmetric positive semidefinite matrix, noted $\mathbf{A} \succcurlyeq 0$. Then, the singular values of $\mathbf{A}$ exactly match its eigenvalues.*

*Proof.* Let $\mathbf{A}$ be an $n \times n$ symmetric positive semidefinite matrix and let $(\lambda, v)$ be an eigenpair. Hence

$$\mathbf{A}v = \lambda v \implies \mathbf{A}^\top \mathbf{A}v = \mathbf{A}^\top \lambda v \overset{\mathbf{A}^\top = \mathbf{A}}{\implies} \mathbf{A}^\top \mathbf{A}v = \lambda^2 v \ .$$

Therefore, $\lambda^2$ is an eigenvalue of $\mathbf{A}^\top \mathbf{A}$. We recall that the singular values of $\mathbf{A}$ are the square root of the eigenvalues of $\mathbf{A}^\top \mathbf{A}$. Since $\mathbf{A} \succcurlyeq 0$, we deduce that $\lambda \geq 0$ and hence $\sqrt{\lambda^2} = \lambda$. Thus, the singular values of $\mathbf{A}$ are its eigenvalues. □

# Part III

# Conclusion

CHAPTER 6

# Conclusion and Perspectives

In this thesis, we first analysed practical variants of stochastic variance reduced gradient methods for solving the Empirical Risk Minimization problem. We showed for the SAGA algorithm that an optimal mini-batch size could minimize total complexity. Such an analysis was possible through the concept of expected smoothness which measures the smoothness of the stochastic subsampled loss. This quantity allowed us to develop convergence results interpolating between the regime of SAGA with single element sampling and full gradient descent.

Then, using the same theoretical tools and introducing the expected residual, we studied the convergence of variants of the SVRG algorithm. We were also able to set the parameters to optimal values such as the mini-batch and step sizes to minimize the total complexity of our algorithm. Moreover, our analysis focused on more practical variants of SVRG for which our theory ensures convergence for all values of the outer loop.

Finally, we worked on extending sketch-and-project methods to solve structured linear systems. The ridge regression problem appeared to be a natural problem to tackle as its optimality conditions boil down to solving a primal or dual system depending on the number of samples and features. This study gave us an opportunity to prove a fast sublinear convergence rate of a momentum variant of the sketch-and-project method and to experiment with the efficiency of several sketching techniques such as subsampling and SubCount sketch. We also extended the latter method to Alternating-Direction Implicit method. Yet, our suggested methods could barely compete against 60-year old well optimized ADI methods.

After our papers [45] and [134], the understanding of the expected smoothness then went on to be key in understanding a host of stochastic gradient methods, like SGD in the convex [58] and non-convex setting [77] and other variance reduced methods such as in [78].

As discussed in the previous chapter, there are still lots of challenges to overcome in the randomization of ADI methods. We believe that sketching could help, especially for problems with interlaced spectra for which no optimal shifting strategy exists. Our intuition is that improvements could be done in our implementation of our BSADI method. For instance, one could try to solve approximately each sketched linear system in each half-step. We also think that even if block row and column subsampling are quick operations in Python, they do not extract nor compress enough information out of the matrices. Thus, one could try other type of sketching methods as *Count* or *SubCount sketch*.

Variance reduction has already started to be introduced in Federated Learning for instance by using control variates like in SCAFFOLD [75] and L2SGD+ [63] or by adding momentum terms such as in MIME [74]. Following this stream, we think the study we did on interpolating between stochastic variance reduced methods and full batch gradient descent could be applied to the Federated Learning framework. This way we could try to optimize the computational cost by estimating the expected smoothness on the different workers and thus personalizing the mini-batch size to each of them. This could lead to convergence rate improvements and maybe enhance robustness by spotting very irregular losses. Moreover, exploring the aggregation of client's updates from a "sketch-and-project" perspective could lead too new interesting updates involving other compression schemes, like sketching methods we studied in Chapter 4.

# Bibliography


[1] N. Ailon and B. Chazelle. "The Fast Johnson-Lindenstrauss Transform and Approximate Nearest Neighbors." *SIAM J. Comput.* 39 (1) (May 2009), pp. 302–322.

[2] N. Ailon and E. Liberty. "Fast dimension reduction using Rademacher series on dual BCH codes." *Discrete & Computational Geometry* 42 (4) (2009), p. 615.

[3] Z. Allen-Zhu. "Katyusha: The first direct acceleration of stochastic gradient methods." *The Journal of Machine Learning Research* 18 (1) (2017), pp. 8194–8244.

[4] Z. Allen-Zhu and E. Hazan. "Variance Reduction for Faster Non-Convex Optimization." In: *Proceedings of The 33rd International Conference on Machine Learning*. Vol. 48. 2016, pp. 699–707.

[5] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. D. J. Dongarra, J. D. Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Third. Philadelphia, Pennsylvania, USA: SIAM, 1999.

[6] H. Avron, P. Maymounkov, and S. Toledo. "Blendenpik: Supercharging LAPACK's least-squares solver." *SIAM Journal on Scientific Computing* 32 (3) (2010), pp. 1217–1236.

[7] R. Babanezhad, M. O. Ahmed, A. Virani, M. Schmidt, J. Konečnỳ, and S. Sallinen. "Stop Wasting My Gradients: Practical SVRG." *Advances in Neural Information Processing Systems* 28 (2015), pp. 2251–2259.

[8] F. Bach. "Sharp analysis of low-rank kernel matrix approximations." In: *COLT 2013 - The 26th Annual Conference on Learning Theory*. 2013, pp. 185–209.

[9] F. Bach. *Learning Theory from First Principles Draft*. 2021, to appear.

[10] F. Bach and E. Moulines. "Non-strongly-convex smooth stochastic approximation with convergence rate $o(1/n)$." *arXiv preprint arXiv:1306.2119* (2013).

[11] Z.-Z. Bai and W.-T. Wu. "On greedy randomized Kaczmarz method for solving large sparse linear systems." *SIAM Journal on Scientific Computing* 40 (1) (2018), A592–A606.

[12] R. H. Bartels and G. W. Stewart. "Solution of the matrix equation AX+XB=C." *Communications of the ACM* 15 (9) (1972), pp. 820–826.

[13] A. Beck. *First-order methods in optimization*. SIAM, 2017.

[14] P. Benner, P. Kürschner, and J. Saak. "Self-generating and efficient shift parameters in ADI methods for large Lyapunov and Sylvester equations." *Electronic Transactions on Numerical Analysis (ETNA)* 43 (2014), pp. 142–162.

[15] P. Benner, R.-C. Li, and N. Truhar. "On the ADI method for Sylvester equations." *Journal of Computational and Applied Mathematics* 233 (4) (2009), pp. 1035–1045.

[16] D. P. Bertsekas. "Incremental least squares methods and the extended Kalman filter." *SIAM Journal on Optimization* 6 (3) (1996), pp. 807–822.

[17] R. Bhatia and P. Rosenthal. "How and why to solve the operator equation AX-XB=Y." *Bulletin of the London Mathematical Society* 29 (1) (1997), pp. 1–21.

[18] G. Birkhoff, R. S. Varga, and D. Young. "Alternating direction implicit methods." In: *Advances in computers*. Vol. 3. Elsevier, 1962, pp. 189–273.

[19] L. Bottou and O. Bousquet. "The Tradeoffs of Large Scale Learning." In: *Advances in Neural Information Processing Systems*. Ed. by J. Platt, D. Koller, Y. Singer, and S. Roweis. Vol. 20. Curran Associates, Inc., 2008.

[20] L. Bottou, F. E. Curtis, and J. Nocedal. "Optimization Methods for Large-Scale Machine Learning." *Siam Review* 60 (2) (2018), pp. 223–311.

[21] C. Boutsidis and A. Gittens. "Improved matrix algorithms via the subsampled randomized Hadamard transform." *SIAM Journal on Matrix Analysis and Applications* 34 (3) (2013), pp. 1301–1340.

[22] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[23] P. L. T. Brian. "A finite-difference method of high-order accuracy for the solution of three-dimensional transient heat conduction problems." *AIChE Journal* 7 (3) (1961), pp. 367–370.
[24] S. Bubeck. "Convex optimization: Algorithms and complexity." *Foundations and Trends® in Machine Learning* 8 (3-4) (2015), pp. 231–357.
[25] D. Calvetti, B. Lewis, and L. Reichel. "On the solution of large Sylvester-observer equations." *Numerical linear algebra with applications* 8 (6-7) (2001), pp. 435–451.
[26] C.-C. Chang and C.-J. Lin. "LIBSVM: A library for support vector machines." *ACM transactions on intelligent systems and technology (TIST)* 2 (3) (2011), p. 27.
[27] M. Charikar, K. Chen, and M. Farach-Colton. "Finding frequent items in data streams." In: *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP)*. Springer-Verlag London, 2002, pp. 693–703.
[28] K. L. Clarkson and D. P. Woodruff. "Low-rank approximation and regression in input sparsity time." *Journal of the ACM (JACM)* 63 (6) (2017), pp. 1–45.
[29] G. Cormode. "Count-Min Sketch." *Management* (2003), pp. 1–5.
[30] G. Cormode and S. Muthukrishnan. "An improved data stream summary: the count-min sketch and its applications." *Journal of Algorithms* (55) (2005), pp. 29–38.
[31] J. A. De Loera, J. Haddock, and D. Needell. "A sampling Kaczmarz-Motzkin algorithm for linear feasibility." *SIAM Journal on Scientific Computing* 39 (5) (2017), S66–S87.
[32] A. Defazio. "A simple practical accelerated method for finite sums." *Advances in neural information processing systems* 29 (2016), pp. 676–684.
[33] 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*. 2014, pp. 1646–1654.
[34] A. Defazio and L. Bottou. "On the ineffectiveness of variance reduced optimization for deep learning." *arXiv preprint arXiv:1812.04529* (2018).
[35] A. Defazio, J. Domke, et al. "Finito: A faster, permutable incremental gradient method for big data problems." In: *International Conference on Machine Learning*. PMLR. 2014, pp. 1125–1133.
[36] J. Douglas and H. H. Rachford. "On the numerical solution of heat conduction problems in two and three space variables." *Transactions of the American mathematical Society* 82 (2) (1956), pp. 421–439.
[37] P. Drineas, M. W. Mahoney, and S. Muthukrishnan. "Sampling algorithms for l2 regression and applications." In: *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*. 2006, pp. 1127–1136.
[38] K. Du and H. Gao. "A new theoretical estimate for the convergence rate of the maximal weighted residual Kaczmarz algorithm." *Numer. Math. Theory Methods Appl* 12 (2) (2019), pp. 627–639.
[39] D. Dua and C. Graff. *UCI Machine Learning Repository*. 2017.
[40] J. Eckstein. "Splitting methods for monotone operators with applications to parallel optimization." PhD thesis. Massachusetts Institute of Technology, 1989.
[41] J. Eckstein and D. P. Bertsekas. "On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators." *Mathematical Programming* 55 (1) (1992), pp. 293–318.
[42] B. J. Fino and V. R. Algazi. "Unified matrix treatment of the fast Walsh-Hadamard transform." *IEEE Transactions on Computers* 25 (11) (1976), pp. 1142–1146.
[43] D. Fortunato and A. Townsend. "Fast Poisson solvers for spectral methods." *IMA Journal of Numerical Analysis* 40 (3) (2020), pp. 1994–2018.
[44] J. Friedman, T. Hastie, R. Tibshirani, et al. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.
[45] N. Gazagnadou, R. Gower, and J. Salmon. "Optimal mini-batch and step sizes for SAGA." In: *International conference on machine learning*. PMLR. 2019, pp. 2142–2150.
[46] G. Golub, S. Nash, and C. Van Loan. "A Hessenberg-Schur method for the problem AX+XB=C." *IEEE Transactions on Automatic Control* 24 (6) (1979), pp. 909–913.
[47] G. H. Golub, P. C. Hansen, and D. P. O'Leary. "Tikhonov Regularization and Total Least Squares." 21 (1) (1999).
[48] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Third. Johns Hopkins University Press, Baltimore, MD, 1996.
[49] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[50] R. Gower, F. Hanzely, P. Richtarik, and S. Stich. "Accelerated Stochastic Matrix Inversion: General Theory and Speeding up BFGS Rules for Faster Second-Order Optimization." In: *Advances in Neural Information Processing Systems 31*. 2018, pp. 2292–2300.
[51] R. M. Gower. "Convergence theorems for gradient descent." *Lecture notes* (2018).
[52] R. M. Gower. "Sketch and Project: Randomized Iterative Methods for Linear Systems and Inverting Matrices." PhD thesis. University of Edinburgh, 2016.
[53] R. M. Gower and P. Richtárik. "Linearly convergent randomized iterative methods for computing the pseudoinverse." *arXiv preprint arXiv:1612.06255* (2016).
[54] R. M. Gower and P. Richtárik. "Stochastic Dual Ascent for Solving Linear Systems." *arXiv:1512.06890* (2015).
[55] R. M. Gower, P. Richtárik, and F. Bach. "Stochastic Quasi-Gradient Methods: Variance Reduction via Jacobian Sketching." *arXiv preprint arXiv:1805.02632* (2018).
[56] R. M. Gower, P. Richtárik, and F. Bach. "Stochastic quasi-gradient methods: Variance reduction via Jacobian sketching." *Mathematical Programming* 188 (1) (2021), pp. 135–192.
[57] R. M. Gower, M. Schmidt, F. Bach, and P. Richtárik. "Variance-reduced methods for machine learning." *Proceedings of the IEEE* 108 (11) (2020), pp. 1968–1983.
[58] R. M. Gower, N. Loizou, X. Qian, A. Sailanbayev, E. Shulgin, and P. Richtárik. "SGD: General analysis and improved rates." In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5200–5209.
[59] R. M. Gower and P. Richtárik. "Randomized Iterative Methods for Linear Systems." *SIAM Journal on Matrix Analysis and Applications* 36 (4) (2015), pp. 1660–1690.
[60] R. M. Gower and P. Richtárik. "Randomized Quasi-Newton Updates are Linearly Convergent Matrix Inversion Algorithms." *SIAM Journal on Matrix Analysis and Applications* 38 (4) (2017), pp. 1380–1409.
[61] D. Gross and V. Nesme. "Note on sampling without replacing from a finite collection of matrices." *arXiv preprint arXiv:1001.2738* (2010).
[62] J. Haddock and A. Ma. "Greed works: An improved analysis of sampling Kaczmarz-Motzkin." *SIAM Journal on Mathematics of Data Science* 3 (1) (2021), pp. 342–368.
[63] F. Hanzely and P. Richtárik. "Federated learning of a mixture of global and local models." *arXiv preprint arXiv:2002.05516* (2020).
[64] M. R. Hestenes and E. Stiefel. "Methods of Conjugate Gradients for Solving Linear Systems." *Journal of Research of the National Bureau of Standards* 49 (6) (1952).
[65] A. S. Hodel and P. Misra. "Solution of underdetermined Sylvester equations in sensor array signal processing." *Linear algebra and its applications* 249 (1-3) (1996), pp. 1–14.
[66] W. Hoeffding. "Probability inequalities for sums of bounded random variables." *Journal of the American statistical association* 58 (301) (1963), pp. 13–30.
[67] E. Hoffer, I. Hubara, and D. Soudry. "Train longer, generalize better: closing the generalization gap in large batch training of neural networks." *arXiv preprint arXiv:1705.08741* (2017).
[68] T. Hofmann, A. Lucchi, S. Lacoste-Julien, and B. McWilliams. "Variance reduced stochastic gradient descent with neighbors." In: *Advances in Neural Information Processing Systems*. 2015, pp. 2305–2313.
[69] S. Horváth and P. Richtárik. "Nonconvex variance reduced optimization with arbitrary sampling." In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2781–2789.
[70] T. S. Jayram and D. P. Woodruff. "Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with subconstant error." *ACM Transactions on Algorithms (TALG)* 9 (3) (2013), pp. 1–17.
[71] R. Johnson and T. Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." In: *Advances in Neural Information Processing Systems*. 2013, pp. 315–323.
[72] W. Johnson and J. Lindenstrauss. "Extensions of Lipschitz mappings into a Hilbert space." In: *Conference in modern analysis and probability (New Haven, Conn., 1982)*. Vol. 26. Contemporary Mathematics. American Mathematical Society, 1984, pp. 189–206.
[73] M. S. Kaczmarz. "Angenäherte Auflösung von Systemen linearer Gleichungen." *Bulletin International de l'Académie Polonaise des Sciences et des Lettres. Classe des Sciences Mathématiques et Naturelles. Série A, Sciences Mathématiques* 35 (1937), pp. 355–357.

[74] S. P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh. "Mime: Mimicking centralized stochastic algorithms in federated learning." *arXiv preprint arXiv:2008.03606* (2020).
[75] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. "Scaffold: Stochastic controlled averaging for federated learning." In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5132–5143.
[76] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. "On large-batch training for deep learning: Generalization gap and sharp minima." *arXiv preprint arXiv:1609.04836* (2016).
[77] A. Khaled and P. Richtárik. "Better theory for SGD in the nonconvex world." *arXiv preprint arXiv:2002.03329* (2020).
[78] A. Khaled, O. Sebbouh, N. Loizou, R. M. Gower, and P. Richtárik. "Unified analysis of stochastic gradient methods for composite convex and smooth optimization." *arXiv preprint arXiv:2006.11573* (2020).
[79] 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* 10 (2) (2016), pp. 242–255.
[80] J. Konečný and P. Richtárik. "Semi-stochastic gradient descent methods." *Frontiers in Applied Mathematics and Statistics* 3 (2017), p. 9.
[81] D. Kovalev, S. Horváth, and P. Richtárik. "Don't jump through hoops and remove those loops: SVRG and Katyusha are better without the outer loop." In: *Algorithmic Learning Theory*. PMLR. 2020, pp. 451–467.
[82] T. N. Krishnamurti. "Numerical weather prediction." *Annual Review of Fluid Mechanics* 27 (Jan. 1995), pp. 195–224.
[83] A. Kulunchakov and J. Mairal. "Estimate Sequences for Variance-Reduced Stochastic Composite Optimization." In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. 2019, pp. 3541–3550.
[84] A. J. Laub. *Matrix analysis for scientists and engineers*. Vol. 91. Siam, 2005.
[85] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86 (11) (1998), pp. 2278–2324.
[86] Y. T. Lee and A. Sidford. "Efficient Accelerated Coordinate Descent Methods and Faster Algorithms for Solving Linear Systems." *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS* (2013), pp. 147–156.
[87] D. Leventhal and A. S. Lewis. "Randomized Methods for Linear Constraints: Convergence Rates and Conditioning." *Mathematics of Operations Research* 35 (3) (2010), pp. 641–654.
[88] H. Lin, J. Mairal, and Z. Harchaoui. "Catalyst acceleration for first-order convex optimization: from theory to practice." *Journal of Machine Learning Research* 18 (1) (2018), pp. 7854–7907.
[89] P.-L. Lions and B. Mercier. "Splitting algorithms for the sum of two nonlinear operators." *SIAM Journal on Numerical Analysis* 16 (6) (1979), pp. 964–979.
[90] J. Liu and S. J. Wright. "An accelerated randomized Kaczmarz algorithm." *Mathematics of Computation* 85 (297) (2016), pp. 153–178.
[91] N. Loizou and P. Richtárik. "Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods." *Computational Optimization and Applications* 77 (3) (2020), pp. 653–710.
[92] A. Lu and E. L. Wachspress. "Solution of Lyapunov equations by alternating direction implicit iteration." *Computers & Mathematics with Applications* 21 (9) (1991), pp. 43–58.
[93] A. Ma, D. Needell, and A. Ramdas. "Convergence properties of the randomized extended Gauss-Seidel and Kaczmarz methods." *SIAM J. Matrix Anal. A.* 36 (4) (2015), pp. 1590–1604.
[94] M. W. Mahoney. "Randomized Algorithms for Matrices and Data." *Found. Trends Mach. Learn.* 3 (2) (Feb. 2011), pp. 123–224.
[95] J. Mairal. "Incremental majorization-minimization optimization with application to large-scale machine learning." *SIAM Journal on Optimization* 25 (2) (2015), pp. 829–855.
[96] D. Masters and C. Luschi. "Revisiting small batch training for deep neural networks." *arXiv preprint arXiv:1804.07612* (2018).
[97] M. S. Morshed, S. Ahmad, and M. Noor-E-Alam. "Stochastic Steepest Descent Methods for Linear Systems: Greedy Sampling & Momentum." *arXiv:2012.13087* (2020).

[98] M. S. Morshed, M. S. Islam, and M. Noor-E-Alam. "Accelerated sampling Kaczmarz Motzkin algorithm for the linear feasibility problem." *J. Glob. Optim.* 77 (2) (2020), pp. 361–382.
[99] M. S. Morshed and M. Noor-E-Alam. "Sketch & Project Methods for Linear Feasibility Problems: Greedy Sampling & Momentum." *arXiv:2012.02913* (2020).
[100] T. Murata and T. Suzuki. "Doubly Accelerated Stochastic Variance Reduced Dual Averaging Method for Regularized Empirical Risk Minimization." In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. 2017, pp. 608–617.
[101] I. Necoara. "Faster randomized block Kaczmarz algorithms." *SIAM Journal on Matrix Analysis and Applications* 40 (4) (2019), pp. 1425–1452.
[102] D. Needell. "Randomized Kaczmarz solver for noisy linear systems." *BIT Numerical Mathematics* 50 (2) (2010), pp. 395–403.
[103] D. Needell, N. Srebro, and R. Ward. "Stochastic gradient descent, weighted sampling, and the randomized Kaczmarz algorithm." *Mathematical Programming* 155 (1-2) (2016), pp. 549–573.
[104] D. Needell and J. A. Tropp. "Paved with good intentions: analysis of a randomized block Kaczmarz method." *Linear Algebra and its Applications* 441 (2014), pp. 199–221.
[105] D. Needell, R. Zhao, and A. Zouzias. "Randomized Block Kaczmarz Method with Projection for Solving Least Squares." *Linear Algebra and its Applications* 484 (2015), pp. 322–343.
[106] Y. Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. 1st ed. Springer Publishing Company, Incorporated, 2014.
[107] Y. Nesterov and J.-P. Vial. "Confidence level solutions for stochastic programming." In: *Automatica*. Vol. 44. 2008, pp. 1559–1568.
[108] Y. E. Nesterov. "A method for solving the convex programming problem with convergence rate $O(1/k^2)$." In: *Dokl. akad. nauk Sssr*. Vol. 269. 1983, pp. 543–547.
[109] L. M. Nguyen, J. Liu, K. Scheinberg, and M. Takáč. "SARAH: A novel method for machine learning problems using stochastic recursive gradient." In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2613–2621.
[110] L. M. Nguyen, P. H. Nguyen, M. van Dijk, P. Richtárik, K. Scheinberg, and M. Takáč. "SGD and Hogwild! Convergence Without the Bounded Gradients Assumption." *arXiv preprint arXiv:1802.03801* (2018).
[111] A. Nitanda. "Stochastic Proximal Gradient Descent with Acceleration Techniques." In: *Advances in Neural Information Processing Systems 27*. 2014, pp. 1574–1582.
[112] J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
[113] D. W. Peaceman and H. H. Rachford Jr. "The numerical solution of parabolic and elliptic differential equations." *Journal of the Society for industrial and Applied Mathematics* 3 (1) (1955), pp. 28–41.
[114] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.
[115] S. Petra and C. Popa. "Single projection Kaczmarz extended algorithms." *Numerical Algorithms* 73 (3) (2016), pp. 791–806.
[116] M. Pilanci and M. Wainwright. "Randomized Sketches of Convex Programs With Sharp Guarantees." *Information Theory, IEEE Transactions on* 61 (9) (2015), pp. 5096–5115.
[117] M. Pilanci and M. J. Wainwright. "Iterative Hessian sketch : Fast and accurate solution approximation for constrained least-squares." *Journal of Machine Learning Research* 17 (2016), pp. 1–33.
[118] B. T. Polyak. "Some Methods of Speeding up the Convergence of Iteration Methods." *USSR Computational Mathematics and Mathematical Physics* 4 (1964), pp. 1–17.
[119] B. T. Polyak and A. B. Juditsky. "Acceleration of stochastic approximation by averaging." *SIAM journal on control and optimization* 30 (4) (1992), pp. 838–855.
[120] Z. Qu, P. Richtárik, and T. Zhang. "Quartz: Randomized dual coordinate ascent with arbitrary sampling." *Advances in neural information processing systems* 28 (2015), pp. 865–873.
[121] A. Raj and S. U. Stich. "k-SVRG: Variance Reduction for Large Scale Optimization." *arXiv preprint arXiv:1805.00982* (2018).
[122] S. J. Reddi, A. Hefny, S. Sra, B. Póczos, and A. J. Smola. "Stochastic Variance Reduction for Nonconvex Optimization." In: *Proceedings of the 34th International Conference on Machine Learning*. Vol. 48. 2016, pp. 314–323.

[123] P. Richtárik and M. Takáč. "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function." *Mathematical Programming* 144 (1) (2014), pp. 1–38.
[124] P. Richtárik and M. Takáč. "Parallel coordinate descent methods for big data optimization." *Mathematical Programming* 156 (1-2) (2016), pp. 433–484.
[125] P. Richtárik and M. Takác. "Stochastic reformulations of linear systems: algorithms and convergence theory." *SIAM Journal on Matrix Analysis and Applications* 41 (2) (2020), pp. 487–524.
[126] H. Robbins and S. Monro. "A stochastic approximation method." *Annals of Mathematical Statistics* 22 (1951), pp. 400–407.
[127] H. Robbins and D. Siegmund. "A convergence theorem for non negative almost supermartingales and some applications." In: *Herbert Robbins Selected Papers*. 1985, pp. 111–135.
[128] R. T. Rockafellar. "Monotone operators and the proximal point algorithm." *SIAM journal on control and optimization* 14 (5) (1976), pp. 877–898.
[129] 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*. 2012, pp. 2663–2671.
[130] D. Ruppert. *Efficient estimations from a slowly convergent Robbins-Monro process*. Tech. rep. Cornell University Operations Research and Industrial Engineering, 1988.
[131] J. Sabino. "Solution of large-scale Lyapunov equations via the block modified Smith method." PhD thesis. Rice University, 2007.
[132] G. Saunders, A. Gammerman, and V. Vovk. "Ridge regression learning algorithm in dual variables." In: *Proc. 15th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA, 1998, pp. 515–521.
[133] M. Schmidt, N. Le Roux, and F. Bach. "Minimizing finite sums with the stochastic average gradient." *Mathematical Programming* 162 (1-2) (2017), pp. 83–112.
[134] O. Sebbouh, N. Gazagnadou, S. Jelassi, F. Bach, and R. Gower. "Towards closing the gap between the theory and practice of SVRG." *Advances in Neural Information Processing Systems* 32 (2019), pp. 648–658.
[135] O. Sebbouh, R. M. Gower, and A. Defazio. "On the convergence of the stochastic heavy ball method." *arXiv preprint arXiv:2006.07867* (2020).
[136] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
[137] S. Shalev-Shwartz, Y. Singer, N. Srebro, and A. Cotter. "Pegasos: Primal estimated sub-gradient solver for SVM." *Mathematical programming* 127 (1) (2011), pp. 3–30.
[138] S. Shalev-Shwartz and T. Zhang. "Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization." In: *International conference on machine learning*. PMLR. 2014, pp. 64–72.
[139] S. Shalev-Shwartz and T. Zhang. "Stochastic dual coordinate ascent methods for regularized loss minimization." *Journal of Machine Learning Research* 14 (2) (2013).
[140] O. Shamir. "Without-replacement sampling for stochastic gradient methods." *Advances in neural information processing systems* 29 (2016), pp. 46–54.
[141] J. R. Shewchuk et al. *An introduction to the conjugate gradient method without the agonizing pain*. 1994.
[142] V. Simoncini. "Computational methods for linear matrix equations." *SIAM Review* 58 (3) (2016), pp. 377–441.
[143] R. Smith. "Matrix Equation XA+BX=C." *SIAM Journal on Applied Mathematics* 16 (1) (1968), pp. 198–201.
[144] T. Strohmer and R. Vershynin. "A Randomized Kaczmarz Algorithm with Exponential Convergence." *Journal of Fourier Analysis and Applications* 15 (2) (2009), pp. 262–278.
[145] A. B. Taylor, J. M. Hendrickx, and F. Glineur. "Exact worst-case convergence rates of the proximal gradient method for composite convex minimization." *Journal of Optimization Theory and Applications* 178 (2) (2018), pp. 455–476.
[146] L. N. Trefethen. "Finite difference and spectral methods for ordinary and partial differential equations" (1996).
[147] J. A. Tropp. "Improved analysis of the subsampled randomized Hadamard transform." *Advances in Adaptive Data Analysis* 3 (01n02) (2011), pp. 115–126.

[148] J. A. Tropp. "User-Friendly Tail Bounds for Sums of Random Matrices." *Foundations of Computational Mathematics* 12 (4) (2012), pp. 389–434.
[149] J. A. Tropp et al. "An introduction to matrix concentration inequalities." *Foundations and Trends® in Machine Learning* 8 (1-2) (2015), pp. 1–230.
[150] S. Tu, S. Venkataraman, A. C. Wilson, A. Gittens, M. I. Jordan, and B. Recht. "Breaking Locality Accelerates Block Gauss-Seidel." In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. 2017, pp. 3482–3491.
[151] A. Usadi and C. Dawson. "50 Years of ADI Methods: Celebrating the Contributions of Jim Douglas, Don Peaceman and Henry Rachford." *SIAM News* 39 (2) (2006), p. 2006.
[152] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. "The NumPy array: a structure for efficient numerical computation." *Computing in science & engineering* 13 (2) (2011), pp. 22–30.
[153] R. S. Varga. *Matrix iterative analysis*. Prentice-Hall, 1963.
[154] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python." *Nature methods* 17 (3) (2020), pp. 261–272.
[155] V. Vovk. "Kernel ridge regression." In: *Empirical inference*. Springer, 2013, pp. 105–116.
[156] E. Wachspress. *The ADI model problem*. Springer, 2013.
[157] E. L. Wachspress. *CURE: A generalized two-space-dimension multigroup coding for the IBM-704*. Tech. rep. Knolls Atomic Power Lab., Schenectady, NY, 1957.
[158] E. L. Wachspress. "Extended application of alternating direction implicit iteration model problem theory." *Journal of the Society for Industrial and Applied Mathematics* 11 (4) (1963), pp. 994–1016.
[159] E. L. Wachspress. *Iterative solution of elliptic systems: And applications to the neutron diffusion equations of reactor physics*. Prentice-Hall, 1966.
[160] S. Wang, A. Gittens, and M. W. Mahoney. "Sketched ridge regression: Optimization perspective, statistical perspective, and model averaging." In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3608–3616.
[161] J. Weideman and L. N. Trefethen. "The eigenvalues of second-order spectral differentiation matrices." *SIAM Journal on Numerical Analysis* 25 (6) (1988), pp. 1279–1298.
[162] H. D. Wilber. "Computing Numerically with Rational Functions." PhD thesis. Cornell University, 2021.
[163] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert. "A fast randomized algorithm for the approximation of matrices." *Applied and Computational Harmonic Analysis* 25 (3) (2008), pp. 335–366.
[164] S. J. Wright. "Coordinate descent algorithms." *Mathematical Programming* 151 (1) (2015), pp. 3–34.
[165] L. Xiao and T. Zhang. "A proximal stochastic gradient method with progressive variance reduction." *SIAM Journal on Optimization* 24 (4) (2014), pp. 2057–2075.
[166] R. Yuan, A. Lazaric, and R. M. Gower. "Sketched Newton-Raphson." *arXiv preprint arXiv:2006.12120* (2020).
[167] Y. Y. Zeyuan Allen-Zhu. "Improved SVRG for Non-Strongly-Convex or Sum-of-Non-Convex Objectives." *arXiv:1506.01972* (2015).
[168] E. Zolotarev. "Application of elliptic functions to questions of functions deviating least and most from zero." *Zap. Imp. Akad. Nauk. St. Petersburg* 30 (5) (1877), pp. 1–59.

# Part IV

# Introduction en français

CHAPTER 7

# Introduction

## 7.1 Optimization stochastique pour l'apprentissage supervisé

### 7.1.1 Minimisation du risque empirique

Dans de nombreuses applications industrielles, économiques ou scientifiques où une quantité importante de données est disponible, l'apprentissage automatique est utilisé pour accomplir une tâche spécifique pour laquelle il a été *entraîné*. Le problème d'apprentissage correspondant est appelé *supervisé* lorsque la tâche d'apprentissage est effectuée sur un ensemble de données contenant à la fois des entrées/vecteurs de caractéristiques, *e.g.,* des mesures effectuées sur des patients, et des sorties/cibles, *e.g.,* une variable indiquant la progression de leur diabète[1]. En supposant que ce jeu de données contient $n \in \mathbb{N}$ échantillons et que chacun d'eux appartient à un espace de dimension $d \in \mathbb{N}$, nous désignons le jeu de données par $\{(x_1, y_1), ..., (x_n, y_n)\}$ où $x \in \mathbb{R}^d$ et $y \in \mathbb{R}$. Dans cette thèse, le modèle est représenté par un vecteur de poids $w \in \mathbb{R}^d$.

**Minimisation du risque empirique.** Afin d'ajuster le modèle, les praticiens définissent des fonctions de perte / objectif (éventuellement régularisées) $f_i : \mathbb{R}^d \to \mathbb{R}$ basées sur des paires $(x_i, y_i)$ pour $i = 1, \ldots, n$, qui quantifient la qualité de la prédiction. Plus la prédiction est mauvaise, plus l'objectif est important.

C'est ainsi que, la plupart du temps, la résolution du problème de formation se résume à un problème d'optimisation : minimiser l'objectif sur les données disponibles. Nous appelons ce problème d'optimisation résultant : *minimisation du risque empirique (ERM)* :

$$w^* \in \arg\min_{w \in \mathbb{R}^d} \left( f(w) := \frac{1}{n} \sum_{i=1}^{n} f_i(w) \right) , \tag{7.1}$$

où nous supposons qu'il existe un vecteur de poids optimal $w^*$. Nous soulignons que dans les applications d'apprentissage automatique, l'objectif n'est pas d'estimer la valeur minimale $f(w^*)$ mais de trouver un modèle proche de $w^*$ qui peut être utilisé pour la prédiction sur des données non vues. Évidemment, si $f$ est suffisamment régulier, l'approximation de $w^*$ implique l'approximation de $f(w^*)$. Voir [136, 44] pour une introduction plus large à l'apprentissage supervisé et aux notions de théorie de l'apprentissage, de test et de sélection de modèle que nous omettons dans ce manuscrit.

**Résoudre le problème ERM.** Pour certains objectifs, le problème d'optimisation a une solution à forme fermée. Par exemple, considérons l'objectif quadratique $f_i(w) = \frac{1}{2}(x_i^\top w - y_i)^2$. Le problème ERM devient maintenant le célèbre problème des moindres carrés ordinaires (MCO) et peut être reformulé comme suit

$$w^*_{\text{OLS}} \in \arg\min_{w \in \mathbb{R}^d} \frac{1}{2} \|\mathbf{X}w - y\|_2^2 , \tag{7.2}$$

où $\mathbf{X} \overset{\text{def}}{=} [x_1, \ldots, x_n]^\top \in \mathbb{R}^{n \times d}$ désigne la matrice de conception, $y \overset{\text{def}}{=} [y_1, \ldots, y_n] \in \mathbb{R}^n$ le vecteur cible. En regardant la condition d'optimalité de premier ordre, nous obtenons que si $\mathbf{X}^\top \mathbf{X} \in \mathbb{R}^{d \times d}$ est non

[1] Voir l'ensemble de données `diabetes` disponible sur `https://scikit-learn.org/stable/datasets/toy_dataset.html`.

singulier, la solution de (7.2) est

$$w^*_{\mathrm{OLS}} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top y \ , \tag{7.3}$$

**Nécessité des méthodes itératives.** Dans la plupart des cas, on ne peut pas résoudre exactement et directement le problème ERM. Soit parce qu'il est impossible de résoudre directement les conditions d'optimalité, *e.g.,* la régression logistique n'admet pas de solution à forme fermée. Ou simplement parce que les solveurs directs sont trop coûteux, par exemple dans (7.3), former $\mathbf{X}^\top \mathbf{X}$ coûte $\mathcal{O}\left(nd^2\right)$. ce qui peut s'avérer peu pratique dans le régime du big data où $n, d \gg 1$. Ce régime est typiquement rencontré dans de nombreuses applications d'apprentissage automatique traitant d'énormes quantités de données de haute dimension, comme dans les ensembles de données de publicité sur Internet avec des millions d'échantillons et des centaines de milliers de caractéristiques (certains d'entre eux sont librement accessibles sur LIBSVM [26][2]. Dans ce cas, les praticiens appliquent plutôt des méthodes itératives qui produisent une séquence de $\left(w^k\right)_k$ qui se rapproche de la solution au fil des itérations. On pourrait légitimement se demander si résoudre approximativement (7.1) est suffisamment satisfaisant. En fait, en apprentissage statistique, l'erreur d'optimisation n'a pas besoin d'atteindre la précision de la machine puisqu'elle est également équilibrée par les erreurs d'estimation et d'approximation. Voir [19] pour plus de détails sur ce cadre de décomposition des erreurs. Les méthodes itératives canoniques à utiliser sont les méthodes du premier ordre/du gradient.

### 7.1.2 Méthode de la descente de gradient

Afin d'appliquer des algorithmes du premier ordre pour résoudre le problème ERM, une certaine structure du problème d'optimisation est nécessaire. Les hypothèses que nous allons maintenant faire sur le problème garantissent qu'il admet au moins une solution et permettent de mesurer l'efficacité des méthodes par leur vitesse de convergence.

**Hypothèses générales.** Nous supposons la différentiabilité, la $L$–smoothness et la $\mu$–forte convexité (ce qui est typiquement le cas des problèmes régularisés) de $f$, de sorte que (7.1) admet une solution unique $w^*$ :

**Hypothèse 7.1** (Smoothness)**.** *Il existe $L \geq 0$ tel que pour tout $x, y \in \mathbb{R}^d$,*

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2} \|y - x\|_2^2 \ . \tag{7.4}$$

*On dit que $f$ est $L$–smooth.*

**Hypothèse 7.2** (Forte convexité)**.** *Il existe $\mu \geq 0$ tel que pour tout $x, y \in \mathbb{R}^d$,*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2} \|y - x\|_2^2 \ . \tag{7.5}$$

*On dit que $f$ est $\mu$–fortement convexe.*

**Descente de gradient.** L'algorithme de descente de gradient (GD) part d'un point initial $w^0 \in \mathbb{R}^d$ et minimise une fonction différentiable et convexe en procédant à chaque itération comme suit :

$$w^{k+1} = w^k - \gamma \nabla f(w^k) \ , \tag{7.6}$$

où $\gamma > 0$ est une taille de pas constante. L'intuition derrière GD est qu'en chaque itéré $w^k$, l'opposé de la dérivée première indique une direction de diminution locale des valeurs de la fonction. Dans la Figure 7.1, nous montrons les lignes de niveau et les 100 premières itérés $(w^k)$ de GD avec une taille de pas $\gamma = 1/L$ sur un problème de régression logistique $\ell^2$–régularisé avec des données synthétiques ($n = 100000$, $d = 2$, et paramètre de régularisation $\lambda = 1/\sqrt{n}$)[3]. Nous observons que GD est une *méthode de descente* appropriée car elle diminue l'objectif à chaque itération. GD et ses variantes (proximale, projetée, accélérée, etc.) sont appelées des méthodes *full batch*, car elles nécessitent d'accéder à tous les points de notre jeu de données à chaque itération. En effet, cette méthode nécessite le calcul du gradient de l'objectif entier, *i.e.,* $\nabla f(w^k) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(w^k)$, ce qui coûte $n$ calculs de gradients individuels. Nous

[2]Disponible à `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

[3]Le code correspondant est disponible à l'adresse : `https://github.com/ngazagna/plotOpt`.

**Algorithme 18** DESCENTE DE GRADIENT

**Entrée:** pas $\gamma > 0$
**Initialisation:** $w^0 \in \mathbb{R}^d$
**for** $k = 0, 1, \ldots, K-1$ **do**
  $w^{k+1} = w^k - \gamma \nabla f(w^k)$ ▷ $\mathcal{O}(d)$
**Sortie:** $w^K$

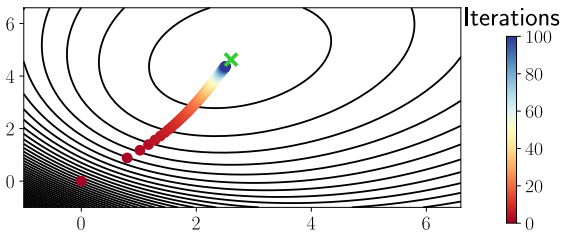



**Figure 7.1:** *Itérés de GD avec $\gamma = 1/L$ affichés sur les lignes de niveau $f$.*

appelons cette quantité la *complexité par itération* ou *coût d'itération*. Pour une présentation exhaustive des méthodes du premier ordre par lots complets, nous nous référons à [13].

Dans l'Algorithme 18, nous fournissons l'implémentation de l'algorithme GD avec le coût du calcul complet du gradient. Notez que dans ce manuscrit, nous présentons souvent des algorithmes itératifs utilisant des boucles for, omettant ainsi le critère d'arrêt qui dépend habituellement de la norme du gradient à l'itération courante $\left\|\nabla f(w^k)\right\|_2$ ou de l'évaluation des pertes $f(w^k)$. Comme la fonction objectif $f$ est la moyenne de $n$ fonctions de perte, son évaluation tend à être très coûteuse lorsque $n \gg 1$. C'est pourquoi les praticiens évitent de calculer $f$ tout au long des itérations afin de ne pas ralentir la convergence.

**Convergence et complexité.** Pour comparer les méthodes d'optimisation, nous pouvons mesurer la vitesse à laquelle elles s'approchent de la solution. Il existe deux façons classiques d'estimer si un itéré est proche d'une solution. Premièrement, en calculant sa *suboptimalité* $f(w^k) - f(w^*)$, qui est par définition une quantité non négative. Deuxièmement, en mesurant sa *distance au minimiseur* $\left\|w^k - w^*\right\|_2^2$ ou à l'ensemble des minimiseurs si son unicité n'est pas satisfaite. Dans le cadre convexe ou non convexe, l'examen de la sous-optimalité est plus pertinent car il peut exister plusieurs solutions du problème ERM (7.1). Dans cette thèse, puisque nous supposons la plupart du temps que la forte convexité est vérifiée, il existe un unique minimiseur et nous nous concentrons sur la distance au minimiseur. Ensuite, nous donnons le résultat classique de la convergence de la descente de gradient sous les Hypothèses 7.1 and 7.2 qui est donné dans de nombreux manuels comme dans le Théorème 3.10 de [24], le Théorème 5.1 de [9], le Théorème 10.29 de [13] dans le cas proximal ou la Section 9.3.1 de [22] pour GD avec recherche linéaire.

**Théorème 7.3** (Convergence des itérés de GD)**.** *Soit le point initial $w^0 \in \mathbb{R}^d$. Si $f$ est $L$–smooth et $\mu$–fortement convexe et si la taille du pas est fixée à $\gamma = 1/L$, alors, les itérés générés par la méthode de descente du gradient convergent conformément à*

$$\left\|w^k - w^*\right\|_2^2 \leq \left(1 - \frac{\mu}{L}\right)^k \left\|w^0 - w^*\right\|_2^2 \quad . \tag{7.7}$$

*Proof.* Nous développons d'abord le terme d'erreur au pas $k \geq 1$ en utilisant la mise à jour par descente de gradient donnée dans (7.6) :

$$\begin{aligned}
\left\|w^k - w^*\right\|_2^2 &= \left\|w^{k-1} - w^* - \gamma \nabla f(w^{k-1})\right\|_2^2 \\
&= \left\|w^{k-1} - w^*\right\|_2^2 - 2\gamma \left\langle \nabla f(w^{k-1}), w^{k-1} - w^* \right\rangle + \gamma^2 \left\|\nabla f(w^{k-1})\right\|_2^2 \\
&\overset{(7.5)}{\leq} (1 - \gamma\mu) \left\|w^{k-1} - w^*\right\|_2^2 - 2\gamma(f(w^{k-1}) - f(w^*)) + \gamma^2 \left\|\nabla f(w^{k-1})\right\|_2^2 \\
&\overset{(7.4)}{\leq} (1 - \gamma\mu) \left\|w^{k-1} - w^*\right\|_2^2 - 2\gamma(1 - \gamma L)(f(w^{k-1} - f(w^*)) \quad .
\end{aligned}$$

En fixant le pas à $\gamma = \frac{1}{L}$, nous obtenons que pour tout $k \geq 1$.

$$\left\|w^k - w^*\right\|_2^2 \leq (1 - \gamma\mu) \left\|w^{k-1} - w^*\right\|_2^2 \quad .$$

□

Le type de convergence donné dans le Théorème 7.3 est connu sous le nom de *convergence linéaire*. Afin d'atteindre une $\epsilon$–solution, c'est-à-dire atteindre un $w^k$ tel que l'erreur relative $\left\|w^k - w^*\right\|_2^2 / \left\|w^0 - w^*\right\|_2^2$ soit plus petite que $\epsilon > 0$, (7.7) implique que l'on doit exécuter au moins

$$\frac{L}{\mu} \log\left(\frac{1}{\epsilon}\right) \tag{7.8}$$

itérations. À partir de ce résultat de *complexité en itérations*, on peut voir que plus le conditionnement $\kappa \overset{\text{def}}{=} L/\mu$ est grand, plus la descente du gradient est lente. En fait, $\kappa$ apparaît dans toutes les analyses de convergence et quantifie la difficulté à résoudre le problème ERM dans (7.1). Cependant, $\mathcal{O}\left(\frac{L}{\mu}\log\left(\frac{1}{\epsilon}\right)\right)$, où la notation $\mathcal{O}$ cache les termes constants, n'est pas la meilleure complexité que les méthodes du premier ordre peuvent atteindre. Un résultat de borne inférieure [106] a montré que la complexité optimale des méthodes de gradient est $\mathcal{O}\left(\sqrt{\frac{L}{\mu}}\log\left(\frac{1}{\epsilon}\right)\right)$, qui est en fait atteinte par la méthode gradient accélérée de Nesterov [108, 24].

**Remarque 7.4.** *Notez que le résultat dans le Théorème 7.3 peut être amélioré. Il est juste donné à titre d'exemple pédagogique. Les Théorèmes et 2.1 et 3.1 dans [145] montrent que le taux de convergence de* (7.7) *peut être remplacé par* $\left(1 - \frac{\mu}{L}\right)^{2k}$.

**Inefficacité en grande dimension.** Cependant, se concentrer sur la complexité d'itération peut être trompeur car cela ne prend pas en compte le coût d'une seule itération. Nous introduisons alors la *complexité totale* : le coût par itération multiplié par la complexité en itérations, qui est égale à $\mathcal{O}\left(n\kappa\log\left(\frac{1}{\epsilon}\right)\right)$ pour GD. Ainsi, en grande dimension, c'est-à-dire lorsque le nombre d'échantillons $n$ est très grand, on observe que l'application de GD pour résoudre le problème ERM devient une tâche difficile et longue. Dans ce régime, les algorithmes stochastiques avec des mises à jour des itérés peu coûteuses sont préférées à la descente de gradient comme nous le verrons dans la section suivante.

### 7.1.3 Algorithmes du gradient stochastique pour les problèmes en grande dimension

L'augmentation considérable du nombre de données $n$ dans les applications récentes de l'apprentissage automatique (comme dans la publicité sur le web et la bioinformatique) exclut l'utilisation des méthodes de gradient *full batch*. En effet, la résolution du problème ERM devient trop longue puisque ces algorithmes doivent parcourir tous les points du jeu de données à chaque itération, ce qui les rend peu compétitifs. En outre, l'objectif des applications d'apprentissage automatique n'est pas seulement de résoudre exactement le problème d'apprentissage, mais de trouver un modèle qui sera performant sur de nouvelles données. En d'autres termes, notre objectif est de résoudre l'ERM de telle sorte que l'erreur d'optimisation atteigne l'erreur statistique, c'est-à-dire l'erreur inhérente due au choix du modèle statistique. Comme décrit dans [19], les méthodes d'optimisation stochastique du premier ordre sont particulièrement bien adaptées aux objectifs qui sont des moyennes de fonctions de perte et conduisent à une convergence plus rapide que les méthodes *full batch* tout en généralisant mieux aux données inconnues. Ces dernières années, les méthodes de gradients stochastiques ont été popularisées par la communauté du Deep Learning avec leur utilisation dans l'entraînement de grands réseaux neuronaux [85].

Ces raisons ont popularisé la méthode de descente de gradient stochastique (SGD), introduite dans [126], pour résoudre de manière incrémentale le problème ERM. Cette méthode ne nécessite qu'une estimation sans biais du gradient total à chaque itération, qui est facilement accessible par le gradient d'une fonction de perte individuelle $f_i$ où $i$ est tiré aléatoirement dans $[n] \overset{\text{def}}{=} \{1, \dots, n\}$ :

$$w^{k+1} = w^k - \gamma_k \nabla f_i(w^k) \ , \tag{7.9}$$

où $\gamma_k > 0$ est la taille du pas à l'itération $k$. L'algorithme SGD est donné dans l'Algorithme 19 et ses 1000 premières itérés avec une taille de pas constante sur le même problème de régression logistique sont affichées dans la Figure 7.2. Une caractéristique importante à souligner, visible dans la Figure 7.2, est que le terme "descente" du gradient stochastique est mal choisi car cette méthode ne satisfait aucune propriété de descente, contrairement à GD pour lequel, au pas $k$

$$f(w^{k+1}) \overset{(7.6)}{=} f(w^k - \gamma\nabla f(w^k)) \overset{(7.4)}{\leq} f(w^k) + \gamma\left(\gamma\frac{L}{2} - 1\right)\left\|\nabla f(w^k)\right\|_2^2 \leq f(w^k)$$

dès que $0 < \gamma \leq 1/2L$.

**Convergence simple de SGD.** SGD s'avère être une alternative aux méthodes *ful-batch* car elle bénéficie d'un faible coût par itération : $\mathcal{O}(1)$ contre $\mathcal{O}(n)$ pour GD. Mais pour comparer ces deux méthodes, nous avons besoin de plus que le coût d'une itération : nous devons comparer leurs complexités totales. Ainsi, donnons un résultat de convergence similaire à celui du Théorème 7.3 pour SGD avec une taille de pas constante.

**Théorème 7.5** (Convergence des itérés de SGD)**.** *Soit $w^0 \in \mathbb{R}^d$ le point initial. Supposons que les gradients stochastiques sont uniformément bornés sur le chemin de SGD* [4] *:*

$$\mathbb{E}_k \left[ \left\| \nabla f_i(w^k) \right\|_2^2 \right] \leq B^2, \quad \forall i \in [n], \forall k \in \mathbb{N}^* \tag{7.10}$$

*où $(w^k)_k$ sont les itérations générées par SGD et $B > 0$. Si $f$ $\mu$–fortement convexe et si la taille du pas est fixée de telle sorte que $0 < \gamma < 1/\mu$, alors, les itérés générés par la méthode du gradient stochastique converge selon*

$$\mathbb{E} \left[ \left\| w^k - w^* \right\|_2^2 \right] \leq (1 - \gamma\mu)^k \left\| w^0 - w^* \right\|_2^2 + \frac{\gamma}{\mu} B^2 \ . \tag{7.11}$$

*Proof.* La preuve est similaire à celle du Théorème 7.3 et est donnée dans [51]. □

**Un compromis de convergence.** Nous retrouvons le terme de gauche qui est le même que dans (7.7) pour GD (avec $\gamma = 1/L$) et diminue exponentiellement avec $k$. Dans le cas stochastique, nous obtenons un second terme de *bruit résiduel* $\gamma B^2/\mu$ que nous ne pouvons pas forcer à converger vers zéro. En effet, (7.11) prouve la convergence de SGD vers un voisinage de la solution et met en évidence un compromis dans le choix de la taille du pas $\gamma$. D'une part, il faudrait fixer $\gamma$ proche de $1/\mu$, qui peut être très grand car souvent $\mu$ avoisine de zéro, pour que le terme de gauche se rapproche rapidement de zéro. Mais d'autre part, $\gamma$ devrait être proche de 0 pour annuler le terme de bruit. Ce compromis est souvent connu comme l'opposition entre *oublier les conditions initiales*, c'est-à-dire s'échapper rapidement de $w^0$ pour approcher $w^*$, et *annuler le bruit*, c'est-à-dire réduire le rayon de la boule autour de la solution que SGD atteint.

**Comparaison de la complexité avec celle de GD.** Des résultats plus avancés [10, 103, 58] montrent la convergence sous-linéaire de SGD avec une taille de pas constante sous une hypothèse fortement convexe sur la perte moyenne $f$ et une hypothèse de $L_i$–smoothness sur les $f_i$. Soit $L_{\max} \overset{\text{def}}{=} \max_{i=1,\dots,n} L_i$. Cela implique que la complexité en itérations de SGD, qui est égale à sa complexité totale puisqu'un seul gradient est calculé à chaque itération, est de

$$\mathcal{O}\left( \left( \frac{L_{\max}}{\mu} + \frac{\sigma^2}{\mu^2 \epsilon} \right) \log\left( \frac{1}{\epsilon} \right) \right) \ , \tag{7.12}$$

où $\sigma^2 \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n \left\| \nabla f_i(w^*) \right\|_2^2$ est le *bruit de gradient*. Nous rappelons que, d'après (7.8), GD bénéficie d'un taux de convergence linéaire conduisant à une complexité totale de $\mathcal{O}\left( n \frac{L}{\mu} \log\left( \frac{1}{\epsilon} \right) \right)$ comme nous l'avons vu dans la section précédente. Par conséquent, cette comparaison suggère d'appliquer la méthode SGD pour obtenir une solution de faible précision, mais pour obtenir des résultats de haute précision lorsque $\epsilon$ est petit, la méthode GD est à privilégier.

**Forcer la convergence de SGD.** Un désavantage évident de la méthode de SGD avec un pas constant est qu'elle ne converge pas exactement vers la solution. Par exemple, si l'on initialise l'Algorithme 19 avec $w^0 = w^*$ la solution du problème ERM (7.1), les itérés suivants s'en éloigneront et oscilleront autour de celle-ci, comme on peut le voir pour les dernières itérations dans la Figure 7.2. Ceci est dû au fait que les gradients individuels $\nabla f_i(w^*)$ ne sont pas nécessairement nuls à l'optimum[5] contrairement au gradient total $\nabla f(w^k)$ qui converge vers 0 lorsque $w^k \to w^*$.

Plusieurs options permettent de corriger ce comportement oscillant et de forcer la convergence. Au lieu de renvoyer la dernière itération, ce qui est la sortie la plus naturelle des méthodes de gradient, on

[4] Il convient de noter que si cette hypothèse est faite en tout point $w$, elle entre en conflit avec l'hypothèse de forte convexité de $f$, voir [110, 20].

[5] Excepté dans ce qu'on appelle les *modèles sur-paramétrés* que nous n'étudions pas dans cette thèse.

**Algorithme 19** GRADIENT STOCHASTIQUE

**Entrée:** suite de pas positifs $(\gamma_k)_k$
**Initialisation:** $w^0 \in \mathbb{R}^d$
**for** $k = 0, 1, \ldots, K-1$ **do**
  Tirer aléatoirement $i \sim [n]$
  $w^{k+1} = w^k - \gamma_k \nabla f_i(w^k)$ ▷ $\mathcal{O}(1)$
**Sortie:** $w^K$

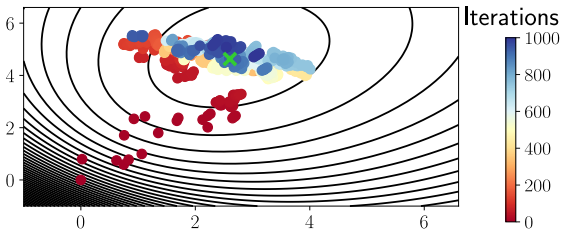



**Figure 7.2:** *Itérés de SGD avec $\gamma = 1/L$ affichés sur les lignes de niveau de $f$.*

peut renvoyer une moyenne des itérés, c'est-à-dire appliquer un moyennage de *Polyak-Ruppert* [119, 130, 136]. Cependant, cette option est moins pratique car elle nécessite de stocker en mémoire une moyenne des itérations précédentes, ce qui est encombrant pour les très grands modèles comme ceux de Deep Learning où le nombre de poids peut être extrêmement grand. Une autre stratégie consiste à utiliser des pas décroissants $\gamma_k$, typiquement de l'ordre de $\mathcal{O}\left(1/\sqrt{k}\right)$ ou $\mathcal{O}(1/k)$, pour annuler le bruit du gradient lors de l'approche de la solution. Cependant, cette séquence de pas est souvent difficile à ajuster et demande beaucoup de temps aux praticiens. Notons que des résultats récents sur la méthode de SGD dans [58] conduisent à un nouveau choix de pas, constants puis décroissants en $\mathcal{O}(1/k)$, qui permet d'atteindre une convergence sous-linéaire tout en annulant le bruit de gradient.

### 7.1.4 Algorithmes du gradient stochastique à variance réduite

Au cours des dernières années, une partie de la communauté de l'optimisation a essayé d'obtenir une convergence linéaire efficace de GD tout en conservant le coût d'itération peu élevé de la SGD. Les auteurs de [129, 133] ont réussi à obtenir "le meilleur des deux mondes" en concevant et en analysant la méthode Stochastic Average Gradient (SAG), un algorithme du *gradient stochastique à variance réduite (VR)* qui réussi à résoudre (7.1) avec une convergence linéaire en exploitant sa structure de somme finie. Ils ont ensuite été suivis par de nombreuses autres méthodes telles que SVRG [71], SAGA [33], la version non biaisée de SAG, SDCA [139], FINITO/MISO [35, 95], SVRG++ [167]. Nous nous référons à l'article de synthèse [57] pour plus de détails sur les méthodes à variance réduite pour résoudre les problèmes d'apprentissage automatique. Notez que pour les méthodes à variance réduite, l'hypothèse de gradients stochastiques bornés (7.10) n'est plus nécessaire. Ceci est préférable car supposer à l'avance une propriété sur les itérés générés par un algorithme pour prouver la convergence de ce dernier est une sorte d'argument circulaire que l'on cherche à éviter.

**Réduction de la variance via des covariables de contrôle.** Dans toutes ces méthodes, on montre que la convergence de la SGD peut être plus rapide si l'on remplace $\nabla f_i(w^k)$ par une meilleure estimation stochastique $g^k \in \mathbb{R}^d$ du gradient total. Cette estimation est généralement choisie pour être sans biais, *e.g.,* $\mathbb{E}\left[g^k\right] = \nabla f(w^k)$, et telle qu'elle a une variance plus petite que $\nabla f_i(w^k)$ en exploitant les informations des gradients stochastiques des étapes précédentes. Réduire la variance de l'estimation du gradient entre les itérations, *i.e.,* $\lim_{w^k \to w^*} \mathrm{Var}(g^k) = 0$, permet ensuite d'utiliser un pas constant, ce qui rend les méthodes du gradient stochastique à variance réduite non seulement plus rapides mais aussi plus pratiques que SGD, qui nécessite des pas décroissants.

Expliquons brièvement comment $\mathrm{Var}(g^k) \overset{\text{def}}{=} \mathbb{E}\left[\left\|g^k - \nabla f(w^k)\right\|_2^2\right]$ peut être réduite. D'abord, soient $n$ *fonctions covariables* $h_i : \mathbb{R}^d \to \mathbb{R}$, pour $i = 1, \ldots, n$. Si $i$ est tiré de façon i.i.d. dans $[n]$, nous pouvons reformuler la fonction objectif comme suit

$$\frac{1}{n}\sum_{i=1}^{n} f_i(w) = \mathbb{E}\left[f_i(w)\right] = \mathbb{E}\left[f_i(w) - h_i(w) + \mathbb{E}\left[h_i(w)\right]\right] \quad . \tag{7.13}$$

Maintenant, à l'itération $k$, au lieu d'appliquer un pas de SGD au problème de l'ERM à gauche de (7.13), appliquons-la à droite de (7.13), que nous appelons la *reformulation stochastique contrôlée* [56] du problème de l'ERM, pour construire une nouvelle estimation du gradient total :

$$g^k \overset{\text{def}}{=} \nabla f_i(w^k) - \nabla h_i(w^k) + \mathbb{E}\left[\nabla h_i(w^k)\right] \quad .$$

**Algorithm 20** SAGA

**Input:** pas $\gamma_k > 0$
**Initialize:** $w^0 \in \mathbb{R}^d$,
$g_j = \nabla f_j(w^0)$ for $j = 1, \dots, n$ ▷ $\mathcal{O}(n)$
$G = \frac{1}{n} \sum_{j=1}^n g_j$
**for** $k = 0, 1, \dots, K-1$ **do**
  Sample $i \sim [n]$
  $g^k = \nabla f_i(w^k) - g_i + G$ ▷ $\mathcal{O}(1)$
  $w^{k+1} = w^k - \gamma g^k$
  $G = G + \frac{1}{n}(\nabla f_i(w^k) - g_i)$
  $g_i = \nabla f_i(w^k)$
**Output:** $w^K$

**Algorithm 21** SVRG

**Input:** pas $\gamma > 0$, taille de boucle interne $m$
**Initialize:** $z^0 \in \mathbb{R}^d$
**for** $k = 0, 1, \dots, K-1$ **do**
  $z = z^k$
  $G = \nabla f(z)$ ▷ $\mathcal{O}(n)$
  $w^0 = z$
  **for** $t = 0, 1, \dots, m-1$ **do**
    Sample $i \sim [n]$
    $g^k = \nabla f_i(w^k) - \nabla f_i(z) + G$ ▷ $\mathcal{O}(1)$
    $w^{k+1} = w^k - \gamma g^k$
  $z^{k+1} = w^m$
**Output:** $w^m$

Par construction, $g^k$ est généralement une estimation non biaisée du gradient, c'est-à-dire que $\mathbb{E}\left[g^k\right] = \nabla f(w^k)$, et sa variance est égale à

$$\mathrm{Var}(g^k) = \mathrm{Var}(\nabla f_i(w^k)) - 2\,\mathrm{Cov}(\nabla f_i(w^k), \nabla h_i(w^k)) + \mathrm{Var}(\nabla h_i(w^k)) \ .$$

Ainsi, la variance de $g^k$ est plus petite que celle du gradient stochastique si $\nabla h_i(w^k)$ et $\nabla f_i(w^k)$ sont suffisamment corrélés positivement. Dans les méthodes VR, nous souhaitons avoir $\nabla h_i(w^k) \approx \nabla f_i(w^k)$ où $\nabla h_i(w)$ est basé sur les calculs de gradient passés. Pour répondre à ces exigences, les fonctions covariables $h_i$ sont souvent choisies comme des approximations linéaires des $f_i$, comme nous le verrons ci-dessous pour SAGA et SVRG. Il est intéressant de noter que même si nous définissons $\nabla h_i(w^k) = \nabla f_i(w^k)$, cela nous retrouvons GD puisque

$$g^k = \nabla f_i(w^k) - \nabla f_i(w^k) + \mathbb{E}\left[\nabla f_i(w^k)\right] = \nabla f(w^k) \ ,$$

qui, certes, a une variance nulle mais que nous voulons éviter en raison de son coût élevé par itération. Enfin, un aspect clé des méthodes VR à garder à l'esprit est qu'elles introduisent une notion de *mémoire*, les informations doivent être stockées à travers les itérations pour être réutilisées comme covariables de contrôle.

**Exemple de SAGA et SVRG.** Détaillons maintenant deux célèbres méthodes de RV étudiées en profondeur dans les Chapitres 2 et 3. Tout d'abord, SAGA [33] est un algorithme qui garde en mémoire un tableau $n \times d$ des gradients individuels précédemment calculés $\nabla f_j$ pour chaque échantillon $j \in [n]$. Ensuite, à l'étape $k$, la mise à jour de SAGA utilise la moyenne des gradients stockés pour corriger le gradient stochastique fraîchement calculé $\nabla f_i(w^k)$ :

$$g^k = \nabla f_i(w^k) - \nabla f_i(z_i^k) + \frac{1}{n} \sum_{j=1}^n \nabla f_j(z_j^k) \ , \tag{7.14}$$

où $z_j^k \in \mathbb{R}^d$ représente le dernier point auquel $\nabla f_j$ a été calculé. Ici, $h_i(w) = w^\top \nabla f_i(z_i^k)$ et donc nous avons $\nabla h_i(w^k) = \nabla f_i(z_i^k)$. Notez qu'en pratique, ces points $z_j^k$ ne sont jamais stockés, seules les $n$ valeurs des gradients correspondants $\nabla f_j(z_j^k) \in \mathbb{R}^d$ sont gardées en mémoire. En effet, c'est exactement comme cela que l'algorithme SAGA est écrit dans l'Algorithme 20 : ces gradients sont stockés dans les variables $g_j$, formant un tableau $d \times n$, et leur moyenne $G$ est calculée à la volée. Le principal inconvénient de SAGA est ce coût mémoire supplémentaire de $\mathcal{O}(nd)$ qui rend cette méthode peu pratique dans le cas de problèmes à très grande échelle[6] [34]. Ce dernier point est important car il peut constituer un obstacle à l'application des méthodes à réduction de variance aux grands réseaux de neurones, pour lesquels $d \gg 1$, entraînés sur des millions d'échantillons, *i.e.,* $n \gg 1$ aussi.

[6] Pour les modèles linéaires généraux (GLM), une astuce de stockage permet de réduire ce coût mémoire à $\mathcal{O}(n)$.

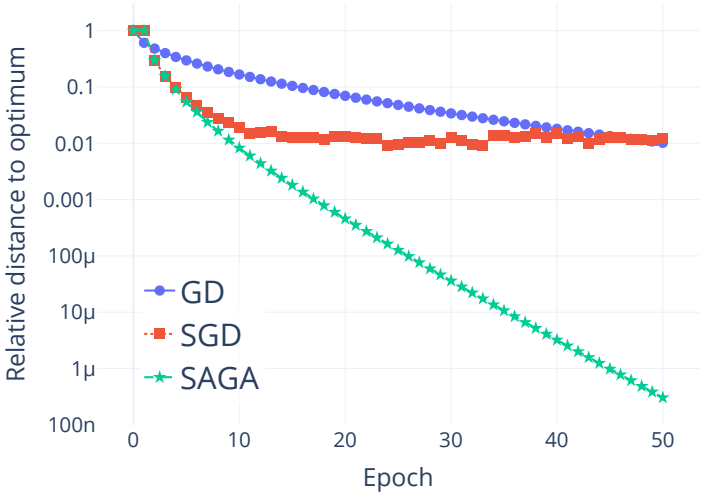



**Figure 7.3:** *Comparaison entre GD, SGD et SAGA avec des tailles de pas théoriques (respectivement $1/L$ pour GD et $1/L_{\max}$ pour les autres) pour un problème synthétique de régression logistique régularisée avec la norme $\ell^2$ ($n = 1000$, $d = 200$, $\lambda = 1/\sqrt{n}$).*

En ce qui concerne SVRG [71], comme le montre l'Algorithme 21, il a un coût mémoire de $\mathcal{O}(d)$ puisqu'il stocke un *point de référence* $z \in R^d$ et le gradient total en ce point $\nabla f(z) \in \mathbb{R}^d$, également appelé *gradient de référence*. Toutes les $m$ itérations, où $m$ est appelé la *taille de la boucle interne*, le point de référence $z$ est réinitialisé à l'itéré courant $w^k$ et le gradient de référence est calculé pour ce nouvel point. L'estimation du gradient pour SVRG est égale à

$$g^k = \nabla f_i(w^k) - \nabla f_i(z) + \nabla f(z) \tag{7.15}$$

Ici, les fonctions covariables sont égales à $h_i(w) = w^\top \nabla f_i(z)$ telles que $\nabla h_i(w^k) = \nabla f_i(z)$. Le point faible de SVRG est que tous les $m$ pas internes, il calcule un gradient total $\nabla f(z)$ ce qui peut être très coûteux dès lors que $m$ est petit ou $n \gg 1$. Notez que dans (7.15) deux gradients stochastiques sont calculés à chaque étape alors qu'un seul est requis dans la mise à jour SAGA dans (7.14). Comme exposé dans l'Algorithme 21, SVRG est classiquement décrit avec une boucle interne et une boucle externe même s'il n'est pas implémenté de cette façon dans la pratique[7]. Il existe également une version *randomisée* de SVRG : Loopless SVRG (L-SVRG) [68, 81] qui remplace la boucle externe par une procédure de tirage aléatoire et met à jour le point de référence $z$ avec une probabilité $p$ (équivalente à $1/m$).

**Complexité des méthodes du gradient stochastique à variance réduite.** En supposant que $f$ est $\mu$–fortement convexe et que les $f_i$ sont convexes et $L_i$–smooth et en rappelant que $L_{\max} = \max_{i=1,\dots,n} L_i$. SAGA, SAG, SVRG et de nombreuses autres méthodes du gradient stochastique à variance réduite bénéficient d'une *convergence linéaire rapide* qui est équivalente à une complexité totale de

$$\mathcal{O}\left(\left(n + \frac{L_{\max}}{\mu}\right) \log\left(\frac{1}{\epsilon}\right)\right) \quad . \tag{7.16}$$

En regard de la complexité de GD présentée dans (7.8), cela signifie que les méthodes VR sont plus rapides que GD dès que

$$L \geq \mu + \frac{L_{\max}}{n} \quad .$$

Le Lemme 2.9 montre que $L \geq L_{\max}/n$, de sorte que cette condition est souvent vérifiée dans les applications d'apprentissage automatique où $\mu$ est négligeable par rapport à $L_{\max}/n$. La Figure 7.3 illustre la convergence linéaire de SAGA par rapport à SGD, qui oscille autour de la solution, et GD, qui est beaucoup plus lent en raison de son coût par itération. Notez que pendant les premières époques (1 époque étant égale à $n$ calculs de gradients stochastiques), SGD et SAGA ont le même comportement car nous n'avons pas initialisé les gradients $(g_j)_{j=1,\dots,n}$ stockés en mémoire. Tout au long des époques,

[7] La boucle interne peut être facilement remplacée par une clause `if` avec un opérateur modulo $m$ appliqué au nombre d'itérations indiquant quand mettre à jour le point et le gradient de référence.

à mesure que les $g_j$ sont mis à jour, la réduction de la variance accélère la convergence de SAGA. De nombreuses variantes (proximale, projetée, en ligne, etc.) de chacune de ces méthodes ont été dérivées, notamment des versions accélérées au sens de Nesterov comme Acc-SDCA [138], Point-SAGA [32], Katyusha [3] et Catalyst [88]. Toutes ces méthodes citées bénéficient d'une convergence linéaire plus rapide avec une complexité $\tilde{\mathcal{O}}\left(\left(n+\sqrt{n\frac{L_{\max}}{\mu}}\right)\log\left(\frac{1}{\epsilon}\right)\right)$ où la notation $\tilde{\mathcal{O}}$ cache les constantes et les termes logarithmiques.

### 7.1.5 Extrapolation entre les méthodes de gradients stochastiques et déterministes

Les méthodes de gradient stochastique ont prouvé leur efficacité dans les applications d'apprentissage automatique telles que dans la Figure 7.3, lorsqu'on chercher à atteindre des solutions de faible précision. Grâce aux récentes avancées technologiques, notamment la parallélisation des calculs, un autre moyen très populaire d'accélérer SGD et ses variantes a été popularisé : il s'agit du *mini-batching* [49, 20].

**Mini-batching.** Cette technique consiste à construire une estimation du gradient total en calculant les gradients stochastiques sur un *mini-batch* de points $B \subseteq [n]$ plutôt que sur un seul :

$$g^k = \nabla f_B(w) = \frac{1}{|B|}\sum_{i\in B}\nabla f_i(w) \ , \tag{7.17}$$

où $f_B(w) \overset{\text{def}}{=} \frac{1}{|B|}\sum_{i\in B} f_i(w)$ désigne la *fonction sous-échantillonnée* pour le mini-batch $B$. À chaque itération, ce mini-batch $B$, généralement de cardinal fixe $b \in [n]$, peut être tiré selon différentes procédures telles que : *échantillonnage par partition*, c'est-à-dire que les données sont séparées en blocs de taille $b$ qui sont échantillonnés de manière aléatoire ou cyclique, *échantillonnage sans remplacement* ou *échantillonnage avec remplacement/b–nice sampling* [8] [56] qui est la plus répandue, et *importance sampling* où les échantillons les plus significatifs sont échantillonnés plus souvent [120, 165]. Dans [16], les auteurs utilisaient déjà des blocs de données pour résoudre de manière incrémentale des problèmes de moindres carrés et mettaient l'accent sur leur utilisation dans l'entraînement de réseaux neuronaux exploitant la technique de *backpropagation/rétropropagation*. Comme dans [137], le calcul des gradients sur un mini-batch peuvent être parallélisés pour accélérer les accélérer, mais même en utilisation en série cette technique peut accélérer l'optimisation. Les questions concernant l'impact du mini-batching sur les performances en généralisation, en particulier dans des contextes d'apprentissage profond [76, 67, 96], ne sont pas abordées ici et vont au-delà de la portée de cette thèse.

D'après (7.17), on remarque que si $b = 1$, on retrouve la méthode SGD (ou une de ses variantes VR) et à l'autre extrême si $b = n$, on retrouve GD. Comme nous l'avons vu, la taille du pas et le taux de convergence de GD dépendent de $L$ dans (7.8) alors que $L_{\max}$ joue ce rôle pour les méthodes de gradient stochastique à variance réduite dans (7.16). Le mini-batching conduit souvent à une convergence plus rapide car $g^k$ estime mieux le gradient total $\nabla f(w^k)$ lorsque $b$ se rapproche de $n$. Aussi, on peut faire de plus grands pas, comme nous le verrons dans le paragraphe suivant, et ainsi accélérer encore plus la convergence.

**Smoothness et taille du pas.** Il est important de noter la dépendance en $L$ de la convergence de GD dans (7.8) alors que $L_{\max}$ joue ce rôle pour les méthodes stochastiques à variance réduite dans (7.16). Par définition de $L_i$–smoothness de $f_i$ (7.4), nous avons que

$$f_i(y) \leq f_i(x) + \langle \nabla f_i(x), y-x\rangle + \frac{L_i}{2}\|y-x\|_2^2, \quad \forall i = 1,\dots,n \ . \tag{7.18}$$

[8] Une confusion peut facilement survenir lorsque l'on parle de mini-batching *avec ou sans remplacement* car il peut soit faire référence à une politique d'échantillonnage locale à l'étape $k$, soit à la politique d'échantillonnage globale sur différentes itérations. Dans la définition de [124], *b–nice sampling* fait référence à une règle locale où à chaque itération $b$ indices sont choisis au hasard sans remplacement parmi $[n]$. Alors que dans [140], *without-replacement sampling* se réfère à l'action d'échantillonner à chaque itération les points qui n'ont pas été échantillonnés dans les itérations précédentes. Ce type de règle d'échantillonnage global introduit plus de difficulté dans l'analyse car les fonctions traitées à chaque itération ne sont pas statistiquement indépendantes.

En faisant la moyenne de ces $n$ inégalités, on obtient

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\bar{L}}{2} \|y - x\|_2^2 \ , \tag{7.19}$$

où $\bar{L} \overset{\text{def}}{=} \frac{1}{n} \sum_{i=1}^n L_i \leq L_{\max}$. Cette dernière inégalité (7.19) montre que

$$L \leq \bar{L} \leq L_{\max} \ , \tag{7.20}$$

puisque $L$ est supposé être la *plus petite constante de Lipschitz* satisfaisant (7.4). Nous pouvons interpréter (7.20) en disant que la perte objective moyenne $f$ que nous cherchons à minimiser dans (7.1) est "plus smooth" que la perte individuelle la plus irrégulière parmi les $f_i$. On remarquera que le pas constant $\gamma$ est proportionnel à l'inverse de ces constantes de smoothness. Ceci implique que GD, avec $\gamma = 1/L$, peut effectuer des pas plus grands que les méthodes stochastiques à variance réduite, pour lesquelles $\gamma \propto 1/L_{\max}$. Soit $B \subseteq [n]$ un mini-batch d'échantillons, nous rappelons que la fonction sous-échantillonnée sur $B$ est définie par $f_B(w) = \frac{1}{|B|} \sum_{i \in B} f_i(w)$. Intuitivement, $f_B$ a plus de chances d'être plus smooth que $f_i$ pour $i \in [n]$ si $b$ est assez grand. Cela explique pourquoi, pour les versions avec mini-batch des méthodes de gradient stochastique à variance réduite, on peut faire des pas plus grands que lorsqu'on échantillonne un seul point à chaque itération. Pourtant, comme nous l'avons vu dans la Figure 7.3, un mini-batch $b$ et des pas $\gamma$ plus grands ne signifient pas toujours une minimisation plus rapide, car en grande dimension GD converge beaucoup plus lentement que SAGA. Une question naturelle se pose donc :

*Comment le taux de convergence se comporte-t-il dans ce cadre intermédiaire de mini-batching ?*

**Expected smoothness.** Pour étudier l'effet des techniques classiques d'échantillonnage et en particulier du mini-batching, des travaux récents [56] ont introduit le concept clé d'*expected smoothness*. Au lieu de supposer que chaque perte individuelle $f_i$ est $L_i$–smooth ou que $f$ est $L$–smooth comme dans (7.4) et (7.18), supposons une forme plus générale de smoothness qui mesure la régularité de la fonction sous-échantillonnée $f_B$. Dans le cas d'un mini-batching sans remplacement, cette hypothèse peut être simplifiée comme suit.

**Hypothèse 7.6** (Constante d'expected smoothness)**.** *Soit $B \subseteq [n]$ un sous-ensemble aléatoire de $b$ points échantillonnés sans remplacement. Il existe $\mathcal{L}(b) > 0$ tel que*

$$\mathbb{E}\left[\|\nabla f_B(w) - \nabla f_B(w^*)\|_2^2\right] \leq 2\mathcal{L}(b)\left(f(w) - f(w^*)\right) \ . \tag{7.21}$$

Nous remarquons que cette propriété de $\mathcal{L}$ ne définit pas exactement la régularité dans le sens d'avoir des dérivées de premier ordre lipschitziennes comme dans (7.4). Elle est plutôt liée à une propriété des fonctions convexes et $L$–smooth minimisées en $w^*$. Sous ces hypothèses, l'équation (2.1.7) dans [106] nous donne que la variance du gradient est contrôlée par la sous-optimalité comme suit

$$\mathbb{E}\left[\|\nabla f(w) - \nabla f(w^*)\|_2^2\right] \leq 2L\left(f(w) - f(w^*)\right) \ .$$

Dans sa formulation générale, $\mathcal{L}$ capture l'effet de la stratégie d'échantillonnage tout entière, qui englobe les probabilités d'échantillonnage et la taille du mini-batch. Nos analyses dans les Chapitres 2 et 3 tirent parti des résultats de [56], où les auteurs ont prouvé que la constante d'expected smoothness $\mathcal{L}$ régit la complexité d'itération et la taille de pas d'une grande classe de méthodes stochastiques, y compris SAGA. L'expected smoothness est l'outil clé qui nous permet d'estimer de manière précise la complexité totale des versions mini-batch des méthodes stochastiques à variance réduite sous des stratégies d'échantillonnage très différentes.

## 7.2 La méthode de sketch-and-project pour la résolution de systèmes linéaires structurés

Les systèmes linéaires sont omniprésents dans de nombreux domaines quantitatifs tels que l'industrie, l'économétrie ou la physique et les résoudre efficacement est une tâche courante. Ils sont généralement définis comme la résolution en $w \in \mathbb{R}^m$ d'un système linéaire.

$$\mathbf{A}w = b \ , \tag{7.22}$$

où $\mathbf{A} \in \mathbb{R}^{m \times m}$ et $b \in \mathbb{R}^m$[9]. L'étude des systèmes linéaires est au cœur de l'algèbre linéaire, et de nombreux problèmes d'analyse numérique, d'informatique, d'optimisation, d'apprentissage automatique, etc. se résument à leur résolution. De plus, la résolution de systèmes linéaires est parfois une sous-routine de programmes ou d'algorithmes plus larges, comme dans les méthodes de Newton en optimisation ou dans la méthode de Peaceman-Rachford [113] étudiée dans le Chapitre 5.

La vitesse de résolution de (7.22) dépend fortement de la *structure* (sparsité, type de matrice $A$, format de stockage de $A$) du problème. Par exemple, les célèbres fonctions Python telles que `numpy.linalg.solve`[10] [152] pour les problèmes denses ou `scipy.linalg.solve`[11] [154] pour les matrices creuses (généralement au format CSC, CSR ou COO[12]) exécute le solveur de système linéaire le plus approprié (dans ces cas, les routines LAPACK [5]) selon que $\mathbf{A}$ est défini positif, défini semi-positif, symétrique, etc. Dans certaines applications, la matrice $A$ peut également être inaccessible et seule sa représentation en tant qu'opérateur linéaire peut être utilisée via son produit matrice-vecteur $A : w \mapsto Aw$.

La dimension du problème, ici $m$, influe également sur le choix du solveur. Dans les petites dimensions, les solveurs directs (reposant souvent sur les factorisations de Cholesky, QR ou LU) [48] sont préférés même s'ils présentent un coût de $\mathcal{O}(m^3)$. Cependant, à grande échelle, il se peut que l'on ne puisse pas charger la matrice entière $A$ en RAM et ce coût cubique les rend terriblement lentes.

**Méthodes itératives randomisées.** Afin de faire évoluer la résolution de systèmes linéaires pour des problèmes en très grande dimension, de récentes avancées ont prouvé l'utilité de la *randomisation*. En ce sens, des parallèles peuvent être établis entre la randomisation de GD conduisant à SGD et celle des méthodes itératives déterministes classiques, *e.g.,* Jacobi, Gauss-Seidel et SOR [48], en *solveurs itératifs randomisés*, *e.g.,* méthodes de Kaczmarz randomisé (RK) et de Gauss-Seidel/Coordinate Descent randomisé (RCD). Contrairement aux solveurs directs, qui doivent effectuer tout leur processus avant de fournir la solution à (7.22), les solveurs itératifs randomisés sont plus flexibles puisqu'ils construisent séquentiellement des itérations $(w^k)_k$ qui se rapprochent de plus en plus de la solution (si elle est unique). Ces méthodes s'arrêtent lorsqu'un critère d'arrêt est satisfait (généralement si le résidu est inférieur à un seuil donné), et renvoient la dernière itération comme solution approximative du système. Des travaux relativement récents sur les méthodes RK [73, 144, 102, 103] et RCD [87, 123, 93] ont prouvé leur efficacité, tant sur le plan théorique que numérique, face aux solveurs déterministes itératifs classiques ou directs. Ces résultats ont confirmé la nécessité d'une randomisation de l'algorithme classique à l'échelle, car le traitement de quantités massives de données est aujourd'hui une pratique courante. Par soucis d'exhaustivité, nous mentionnons la méthode classique du gradient conjugué (CG) [64, 141], datant des années 1950, qui est toujours considérée comme un standard difficile à battre puisqu'elle converge typiquement en $n$ itérations (voir le chapitre 5 de [112]).

### 7.2.1 La méthode de sketch-and-project

Cette section présente une méthode appelée *sketch-and-project* [59, 60, 53, 52] qui a été initialement conçue pour résoudre des systèmes linéaires classiques comme (7.22). Dans ce qui suit, nous présentons cette méthode et expliquons comment elle a été utilisée et étendue à d'autres problèmes linéaires en tirant parti de leur structure inhérente dans les Chapitres 4 et 5. Sketch-and-project est un algorithme itératif randomisé qui unifie une grande variété de méthodes, notamment RK [144, 102, 103] et RCD [59], ainsi que toutes leurs variantes avec échantillonnage par importance ou par blocs. Cette méthode va au-delà des simples techniques d'échantillonnage de lignes ou de colonnes et permet de nombreuses techniques de sketching plus générales, présentées ci-après.

Pour résoudre approximativement un système linéaire, nous pouvons utiliser une *matrice de sketching* $\mathbf{S} \in \mathbb{R}^{m \times \tau}$ pour réduire le nombre de lignes du système linéaire (7.22) à $\tau$ lignes comme suit

$$\mathbf{S}^\top \mathbf{A} w = \mathbf{S}^\top b \ . \tag{7.23}$$

**Sketching à un coup.** Comme dans [37, 94], si la matrice de sketching est correctement choisie et surtout si le nombre de lignes échantillonnées $\tau$ est suffisamment grand, alors la solution du système linéaire *sketché* (7.23) est proche de la solution $w^*$ du système initial (7.22) avec une forte probabilité. Nous appelons une telle technique de sketching *à un coup*, par opposition aux méthodes de *sketching*

[9] Dans cette section nous ne considérerons que les systèmes carrés d'ordre $m$.

[10] See `https://numpy.org/doc/stable/reference/generated/numpy.linalg.solve.html`.

[11] See `https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.solve.html`.

[12] Les formats de matrices creuses de Scipy sont indiqués dans `https://docs.scipy.org/doc/scipy/reference/sparse.html`.

**Algorithm 22** L'Algorithme Sketch-and-Project

**Input:** distribution $\mathcal{D}$ sur les matrices $m \times \tau$

**Initialize:** $w^0 = 0_m$, $r^0 = \mathbf{A}w^0 - b = -b \in \mathbb{R}^m$

**for** $k = 0, 1, \ldots, K-1$ **do**

Tirer aléatoirement $\mathbf{S} \sim \mathcal{D}$

$w^{k+1} = w^k - \mathbf{A}^\top \mathbf{S} \left(\mathbf{S}^\top \mathbf{A}\mathbf{A}^\top \mathbf{S}\right)^\dagger \mathbf{S}^\top (\mathbf{A}w^k - b)$

**Output:** $w^K$

*itératives* telles que les méthodes de sketch-and-project ou de Iteration Hessian Sketching (IHS) [117], qui impliquent un processus de sketching à chaque étape. Cependant, malgré les garanties statistiques données par l'analyse du sketching à un coup, il peut être difficile d'estimer la *taille du sketching* $\tau$ et si cette quantité est trop grande, *i.e.,* proche de la taille du système original $m$, le calcul de la solution approchée sera toujours très lent en raison de l'absence de réduction de dimension. De plus, le système sketché (7.23) peut admettre plusieurs solutions même s'il en existe une unique pour le système original (7.22) et enfin, avec une faible probabilité, la solution du système sketché peut être éloignée de $w^*$ ce qui peut être préjudiciable.

**Sketch-and-project.** Afin de résoudre ces problèmes, les méthodes de sketching itératives exploitent pas à pas les nouvelles informations compressées du système et produisent des itérés $(w^k)_k$ qui s'approchent de $w^*$. Plus précisément, dans les méthodes de sketch-and-project, nous appliquons une étape de projection itératif. À l'étape $k$, une nouvelle matrice de sketching $\mathbf{S}$ est échantillonnée à partir de $\mathcal{D}$, une distribution prédéfinie sur les matrices de taille $m \times \tau$, et l'itération actuelle $w^k$ est projetée sur l'espace solution du système sketché $\mathbf{S}^\top \mathbf{A}w = \mathbf{S}^\top b$, soit

$$w^{k+1} = \underset{w \in \mathbb{R}^m}{\arg\min} \left\| w - w^k \right\|_2^2 \quad \text{subject to} \quad \mathbf{S}^\top \mathbf{A}w = \mathbf{S}^\top b \ . \tag{7.24}$$

Dans cette section, nous avons décidé d'utiliser la projection euclidienne, même si des normes pondérées correctement choisies en fonction des caractéristiques de $A$ peuvent accélérer le taux de convergence [59]. Le problème dans (7.24) est connu comme le *point de vue "sketching"* de sketch-and-project puisque $w^{k+1}$ est défini comme le point le plus proche de $w^k$ qui résout le système sketché. Sa solution sous forme fermée est donnée par

$$w^{k+1} = w^k - \mathbf{A}^\top \mathbf{S} \left(\mathbf{S}^\top \mathbf{A}\mathbf{A}^\top \mathbf{S}\right)^\dagger \mathbf{S}^\top (\mathbf{A}w^k - b) \ , \tag{7.25}$$

où $^\dagger$ désigne le pseudo-inverse de Moore–Penrose.

Une implémentation naïve de cette méthode est donnée dans l'Algorithme 22. Des discussions sur la façon d'améliorer l'implémentation pour accélérer la convergence sont disponibles dans le Chapitre 4. Dans l'Algorithme 22, on peut voir que le seul "paramètre" est la distribution $\mathcal{D}$ à partir de laquelle la matrice de sketching $(\mathbf{S})_k$ est échantillonnée. Cela permet d'utiliser une vaste palette de techniques de compression, non seulement les échantillonnages par lignes/colonnes (blocs) et par importance, mais aussi les transformations rapides de Johnson-Lindenstrauss (JL) [2] ou toute technique de sketching mélangeant les informations lignes/colonnes de manière linéaire comme le *CountSketch* [27].

**Application aux modèles de régression.** Comme nous l'avons vu dans la Section 7.1, certains problèmes ERM apparaissant en apprentissage automatique peuvent être facilement écrits comme la résolution d'un système linéaire ou approchés par une formulation des moindres carrés. C'est par exemple le cas des problèmes de moindre carré ordinaire (7.2) et de régression ridge [132, 47]. Comme présenté dans (7.2), la solution du problème des moindres carrés $w^*_{\text{OLS}}$ est donnée par ses conditions d'optimalité, c'est-à-dire en résolvant en $w$ le système suivant

$$\mathbf{X}^\top \mathbf{X}w = \mathbf{X}^\top y \ . \tag{7.26}$$

Dans le Chapitre 4, nous nous concentrons sur l'application de la méthode de sketch-and-project au problème de la régression ridge qui présente des conditions d'optimalité similaires mais admet une solution

unique. Cela a conduit à une nouvelle méthode appelée *RidgeSketch*. Ce projet a été l'occasion de publier un package Python open source éponyme disponible à l'adresse suivante :

`https://github.com/facebookresearch/RidgeSketch`.

Ce package est entièrement documenté, testé, avec une couverture étendue et est fourni avec deux notebooks de didacticiels pour encourager son utilisation future par la communauté des chercheurs.

### 7.2.2 Interprétation de sketch-and-project comme une méthode SGD

Dans notre analyse de *RidgeSketch* dans le Chapitre 4, nous avons réussi à fournir des garanties de convergence pour une version avec momentum de (7.25) :

$$w^{k+1} = w^k - \mathbf{A}^\top \mathbf{S} \left(\mathbf{S}^\top \mathbf{A}\mathbf{A}^\top \mathbf{S}\right)^\dagger \mathbf{S}^\top (\mathbf{A}w^k - b) + \beta_k (w^k - w^{k-1}) \ , \tag{7.27}$$

avec des paramètres de momentum dépendant de l'itération courante $\beta_k \geq 0$. Cette idée a été introduite dans [125], où les auteurs ont développé des *reformulations stochastiques* de systèmes linéaires et ont dérivé que la méthode de sketch-and-project pouvait être considérée comme une méthode de SGD appliquée à cette nouvelle formulation de l'objectif. Le même raisonnement a également été utilisé récemment pour donner une interprétation SGD en ligne de la méthode *Newton-Raphson* [166] et pour établir une nouvelle théorie de convergence globale pour cette dernière. Elle a également été utilisée pour prouver la convergence sous-linéaire de l'itération moyenne de de sketch-and-project avec un momentum constant [91]. Ce nouveau point de vue nous a permis de fournir une nouvelle preuve (et plus simple que celle de [91] et traitant de la convergence du dernier itéré) de la convergence sous-linéaire de la version momentum de *RidgeSketch* dans le Chapitre 4 en exploitant des résultats récents [135] sur la convergence de la méthode Stochastic Heavy Ball (SHB).

Soit $\|x\|_{\mathbf{W}}$ la norme pondérée définie par une définie positive symétrique $\mathbf{W} \in \mathbb{R}^{m\times m}$ telle que $\|x\|_{\mathbf{W}} = \sqrt{\langle \mathbf{W}x, x\rangle}$, pour tout $x \in \mathbb{R}^m$. Soit $\mathbf{S} \sim \mathcal{D}$ et soit $\mathbf{H}_{\mathbf{S}} \overset{\text{def}}{=} \mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{A}^\top \mathbf{S})^\dagger \mathbf{S}^\top \in \mathbb{R}^{m\times m}$. Pour établir ce point de vue SGD, définissons d'abord la fonction stochastique

$$f_{\mathbf{S}}(w) \overset{\text{def}}{=} \frac{1}{2} \|\mathbf{A}w - b\|^2_{\mathbf{H}_{\mathbf{S}}} \ ,$$

qui est une version pondérée du résidu. Le système linéaire peut alors être reformulé comme le problème de minimisation stochastique suivant

$$\min_{w\in\mathbb{R}^d} f(w) \overset{\text{def}}{=} \frac{1}{2} \|\mathbf{A}w - b\|^2_{\mathbb{E}[\mathbf{H}_{\mathbf{S}}]} = \mathbb{E}\left[\frac{1}{2} \|\mathbf{A}w - b\|^2_{\mathbf{H}_{\mathbf{S}}}\right] =: \mathbb{E}\left[f_{\mathbf{S}}(w)\right] \ , \tag{7.28}$$

Résoudre (7.22) est maintenant équivalent à résoudre (7.28) tant que $\text{Range}\,(\mathbf{A}) \perp \text{Null}\,(\mathbb{E}\,[\mathbf{H}_{\mathbf{S}}])$, pour lequel on peut appliquer la SGD

$$w^{k+1} = w^k - \gamma_k \nabla f_{\mathbf{S}}(w^k) \ , \tag{7.29}$$

où $\mathbf{S} \sim \mathcal{D}$ est échantillonné de façon i.i.d. à chaque itération, $\gamma_k > 0$ est la taille du pas et le gradient est donné par

$$\nabla f_{\mathbf{S}}(w^k) = \mathbf{A}^\top \mathbf{H}_{\mathbf{S}} \left(\mathbf{A}w^k - b\right) = \mathbf{A}^\top \mathbf{S}(\mathbf{S}^\top \mathbf{A}\mathbf{A}^\top \mathbf{S})^\dagger \mathbf{S}^\top \left(\mathbf{A}w^k - b\right) \ . \tag{7.30}$$

Si l'on fixe la taille de pas $\gamma_k$ à 1, on retrouve précisément dans (7.29) la mise à jour de sketch-and-project (7.25).

Dans le Chapitre 4, nous avons pu proposer une nouvelle preuve de convergence de sketch-and-project avec un momentum dépendant de l'itération courante en exploitant cette reformulation stochastique des systèmes linéaires. Techniquement parlant, nous exploitons des propriétés de la fonction stochastique $f_{\mathbf{S}}$ telles que sa convexité, le caractère non-biaisé de son gradient, *e.g.,* $\mathbb{E}\left[\nabla f_{\mathbf{S}}(w^k)\right] = \nabla f(w^k)$ et son rapport avec le résidu $\|\mathbf{A}w - b\|_2^2$ (voir [125]).

### 7.2.3 Randomisation des méthodes ADI par sketch-and-project

Comme nous l'avons vu, la méthode de sketch-and-project peut être appliquée à des problèmes nécessitant la résolution d'un système linéaire comme une méthode itérative stochastique et peut même être interprété comme un forme de SGD. De plus, elle peut également être utilisée pour randomiser des

**Algorithm 23** Méthode de Peaceman-Rachford

**Input:** paramètres de shift $(p_k)_k, (q_k)_k \in \mathbb{R}$

**Initialize:** $u^0 \in \mathbb{R}^m$

**for** $k = 0, 1, \ldots, K-1$ **do**

$u^{k+1/2} \leftarrow$ solution en $u$ de $(\mathbf{H} + p_k \mathbf{I}_m)u = s + (p_k \mathbf{I}_m - \mathbf{V})u^k$

$u^{k+1} \leftarrow$ solution en $u$ de $(\mathbf{V} + q_k \mathbf{I}_m)u = s + (q_k \mathbf{I}_m - \mathbf{H})u^{k+1/2}$

**Output:** $u^K$

sous-routines d'algorithmes nécessitant l'inversion de systèmes linéaires, comme les mises à jour de quasi-Newton dans [60]. Dans le Chapitre 5, nous nous concentrerons sur les équations linéaires formées par la somme de deux opérateurs, *i.e.,* $\mathbf{\Sigma} \stackrel{\text{def}}{=} \mathbf{H} + \mathbf{V}$. Notre travail visait à randomiser les méthodes *Alternating-Direction Implicit (ADI)* résolvant de tels problèmes, en espérant que cela conduirait à une convergence plus rapide pour des problèmes de grande échelle ou sur des problèmes ayant des propriétés spectrales complexes.

Les méthodes ADI sont généralement utilisées pour résoudre les équations matricielles de Sylvester et de Lyapunov, mais pour illustrer notre propos, nous nous concentrons ici sur un cas plus simple : la méthode *Peaceman-Rachford (PR)* [113]. La méthode PR a été initialement introduite en 1955 pour résoudre l'équation de la chaleur en 2 dimensions, une Équation aux Dérivées Partielles (EDP) parabolique, sur un carré :

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{\partial T}{\partial t} \ , \tag{7.31}$$

avec une condition aux limites de Dirichlet

$$\begin{aligned} T(\pm 1, y, t) &= 0, \quad \forall y \in [-1, 1], \ \forall t > 0 \\ T(x, \pm 1, t) &= 0, \quad \forall x \in [-1, 1], \ \forall t > 0 \ , \end{aligned}$$

et une condition initiale $T(x, y, 0) = 1$ pour tous les $x, y \in [-1, 1]$, où $T$ est la température, $x$ et $y$ sont les coordonnées des points du carré et $t$ représente le temps.

Plus généralement, cette méthode PR est appliquée pour résoudre en $u \in \mathbb{R}^m$ des systèmes linéaires de la forme

$$(\mathbf{H} + \mathbf{V})u = s \ , \tag{7.32}$$

où $\mathbf{H}, \mathbf{V} \in \mathbb{R}^{m \times m}$ sont des matrices symétriques. Dans l'exemple (7.31), les opérateurs $\mathbf{H}$ (resp. $\mathbf{V}$) représentent des opérateurs de différenciation discrète du second ordre par rapport à l'axe $x$ (resp. $y$) et le membre de droite $s$ est égal au vecteur zéro. La méthode PR est particulièrement utile pour les systèmes linéaires tels que la matrice du membre de gauche vaut $\mathbf{\Sigma} = \mathbf{H} + \mathbf{V}$ et est trop difficile à inverser directement [89, 41]. Cette méthode fait partie de la famille des *méthodes de splitting des opérateurs* [40] qui reposent sur une hypothèse fondamentale : même si $(\mathbf{\Sigma} + r\mathbf{I}_m)$ n'est pas facilement inversible, pour $r \in \mathbb{R}$, supposons que $(\mathbf{H} + p\mathbf{I}_m)$ et $(\mathbf{V} + q\mathbf{I}_m)$ sont relativement faciles à inverser, pour certains $p, q \in \mathbb{R}$. Ensuite, en combinant astucieusement les étapes *forward*, qui consistent à appliquer $(\mathbf{H}+p\mathbf{I}_m)$ ou $(\mathbf{V}+q\mathbf{I}_m)$, et les étapes *backward*, qui consistent à appliquer $(\mathbf{H}+p\mathbf{I}_m)^{-1}$ ou $(\mathbf{V}+q\mathbf{I}_m)^{-1}$, on s'approchera d'une application directe de $(\mathbf{\Sigma} + r\mathbf{I}_m)^{-1}$. Par conséquent, $\mathbf{H}$ et $\mathbf{V}$ ne sont pas simplement additionnés et nous ne pouvons pas non plus appliquer l'*algorithme du point proximal* [128].

Ensuite, nous donnons directement les mises à jour de la méthode PR, mais une explication détaillée de l'intuition sous-jacente basée sur le splitting des variables est fournie dans l'introduction du Chapitre 5. La méthode PR résout (7.32) en initialisant un vecteur $u^0 \in \mathbb{R}^m$ (souvent initialisé à zéro) et effectue deux demi-étapes comme décrit dans l'Algorithme 23. Dans le Chapitre 5, nous essayons de remplacer ces deux inversions de systèmes linéaires dans les mises à jour de la méthode PR (et plus généralement de la méthode ADI) en espérant que la randomisation accélérera la convergence en grande dimensions ou lorsque l'information spectrale de $\mathbf{H}$ et $\mathbf{V}$ est inaccessible.

## 7.3 Contributions

Cette thèse est divisée en deux parties : la première partie étudie les méthodes de gradient stochastique à variance réduite, largement utilisées pour résoudre le problème de minimisation du risque empirique, et fournit des résultats théoriques pour des paramètres ou des variantes utiles en pratique (comme le mini-batching). La deuxième partie se concentre sur les extensions de l'algorithme *sketch-and-project* pour s'attaquer à des problèmes de grandes dimensions en fournissant à la fois des preuves théoriques de convergence mais aussi de nombreuses preuves numériques d'efficacité.

Dans ce manuscrit, l'accent est mis sur l'aspect pratique des méthodes proposées. C'est pourquoi des codes open source en Julia ou Python ainsi que des expériences numériques approfondies sont fournis pour permettre la reproductibilité de ce travail. Notez que chaque chapitre peut être lu indépendamment.

La structure de la Partie I est la suivante :

- L'objectif du Chapitre 2 est de fournir une base théorique permettant de saisir les avantages du mini-batching pour SAGA. Une telle analyse est possible grâce à l'utilisation d'un outil récemment introduit appelé l'*expected smoothness* qui mesure la régularité de la fonction (stochastique) sous-échantillonnée avec une taille de mini-batch $b$. Cette quantité nous permet de développer des résultats de convergence interpolant entre le régime de SAGA avec le tirage d'un seul point par pas et la descente de gradient utilisant l'ensemble des points. En particulier, nous fournissons des approximations de cette constante d'expected smoothness, ce qui permet de fixer des tailles de pas beaucoup plus grandes et d'obtenir un taux de convergence beaucoup plus rapide. De plus, nous montrons que la complexité totale de l'algorithme SAGA diminue linéairement avec la taille du mini-batch jusqu'à une valeur prédéfinie : la taille optimale du mini-batch. La suite de codes Julia développée pour ce projet est disponible à l'adresse suivante : `https://github.com/gowerrobert/StochOpt.jl`.
- Dans le Chapitre 3 nous fournissons une analyse plus générale de SVRG que ce qui avait été fait auparavant en utilisant un échantillonnage arbitraire, ce qui nous permet d'analyser virtuellement toutes les formes de mini-batching à travers un seul théorème. De plus, notre analyse se concentre sur des variantes plus pratiques de SVRG, notamment une nouvelle variante de *loopless SVRG* [68, 81, 83] et une variante de *k-SVRG* [121] où $m = n$ et où $n$ est le nombre de points de données. Puisque notre analyse reflète la pratique, nous sommes en mesure de définir les paramètres tels que la taille du mini-batch et du pas en utilisant notre théorie ce qui produit des algorithmes plus efficaces en pratique. Des expériences numériques ont également été réalisées avec la suite de code Julia susmentionnée : `StochOpt.jl`.

La Partie II est divisée comme suit :

- Dans le Chapitre 4, nous proposons de nouvelles variantes de la méthode de sketch-and-project pour résoudre les problèmes de régression ridge en grande dimension. Premièrement, nous proposons une nouvelle alternative de momentum et fournissons un théorème montrant qu'elle peut accélérer la convergence de la méthode sketch-and-project, grâce à un taux de convergence *sous-linéaire*. Deuxièmement, nous envisageons de combiner la méthode de sketch-and-project avec de nouvelles méthodes de sketching telles que le Count sketching, le SubCount sketching (une nouvelle méthode que nous proposons) ainsi que les transformées de Hadamard sous-échantillonnées. Nous montrons expérimentalement que lorsqu'elle est combinée à la méthode de sketch-and-project, le (Sub)Count sketching est très efficace sur les données sparses et Subsample sketching est efficace sur les données denses. Elle est même compétitive par rapport à la méthode du gradient conjugué. Le package open source correspondant développé en Python est disponible à l'adresse suivante : `https://github.com/facebookresearch/RidgeSketch`.
- Dans le Chapitre 5, nous développons des versions sketch-and-project de la méthode Alternating-Direction Implicit (ADI). Un exemple particulier de ces méthodes est la méthode *Peaceman-Rachford (PR)*, conçue pour résoudre les systèmes linéaires quadratiques de la forme $(\mathbf{H}+\mathbf{V})u = s$ où nous ne pouvons accéder qu'aux solveurs des systèmes décalés $(\mathbf{H}+p\mathbf{I}_m)u = s$ et $(\mathbf{V}+q\mathbf{I}_m)u = s$, avec $p, q \in \mathbb{R}$. Ces équations apparaissent en analyse numérique, par exemple lors de la résolution d'EDP, et peuvent facilement être limitées par le coût des inversions de système pour les problèmes de grande dimension. Elles sont également couramment utilisées pour résoudre les équations de matrices de Sylvester de la forme $\mathbf{AX} - \mathbf{XB} = \mathbf{F}$. Nous proposons donc les méthodes *Sketched PR (SPR)* et *Sketched ADI (SADI)* pour essayer de faire évoluer ces algorithmes par la randomisation des deux demi-étapes à l'aide de sketch-and-project. SPR réussit à surpasser la méthode PR classique avec des shifts constants $p, q$ sur des problèmes en grande dimension, mais SADI ne conduit pas encore à une amélioration de la convergence (en temps ou en époques) par rapport à ADI. Les codes correspondants développés en Python sont disponibles à l'adresse suivante : `https://github.com/ngazagna/sadi`.

## 7.4 Publications

Cette thèse est basée sur les articles suivants :

- **N. Gazagnadou**, R. Gower, and J. Salmon. "Optimal mini-batch and step sizes for SAGA". ICML, 2019
- O. Sebbouh, **N. Gazagnadou**, S. Jelassi, F. Bach, and R. Gower. "Towards closing the gap between the theory and practice of SVRG". NeurIPS, 2019
- **N. Gazagnadou**, M. Ibrahim, and R. M. Gower. "RidgeSketch: A Fast sketching based solver for large scale ridge regression". arXiv preprint arXiv:2105.05565, 2021 (Soumis à SIMAX)
- **N. Gazagnadou**. "Sketched ADI: a randomized iterative method for solving large scale Sylvester matrix equations". Preprint à venir

ECOLE
DOCTORALE
DE MATHEMATIQUES
HADAMARD

Titre : Régularité moyenne pour les méthodes stochastiques à variance réduite et méthodes sketch-and-project pour systèmes linéaires structurés

Mots clés : Minimisation du risque empirique, optimisation convexe, méthode stochastique itérative, sketch-and-project, sketching

Résumé : L'augmentation considérable du volume de données ainsi que de la taille des échantillons complexifie la phase d'optimisation des algorithmes d'apprentissage, nécessitant la minimisation d'une fonction de perte. La descente de gradient stochastique (SGD) et ses variantes à réduction de variance (SAGA, SVRG, MISO) sont largement utilisées pour résoudre ces problèmes. En pratique, ces méthodes sont accélérées en calculant des gradients stochastiques sur un "mini-batch" : un petit groupe d'échantillons tiré aléatoirement. En effet, les récentes améliorations technologiques permettant la parallélisation de ces calculs ont généralisé l'utilisation des mini-batchs.
Dans cette thèse, nous nous intéressons à l'étude d'algorithmes du gradient stochastique à variance réduite en essayant d'en trouver les hyperparamètres optimaux : taille du pas et du mini-batch. Cette étude nous permet de donner des résultats de convergence interpolant entre celui des méthodes stochastiques tirant un seul échantillon par itération et la descente de gradient dite "full-batch" utilisant l'ensemble des échantillons disponibles à chaque itération. Notre analyse se base sur la constante de régularité moyenne, outil fondamental de notre analyse, qui permet de mesurer la régularité de la fonction aléatoire dont le gradient est calculé.
Nous étudions un autre type d'algorithmes d'optimisation : les méthodes "sketch-and-project". Ces dernières peuvent être utilisées lorsque le problème d'apprentissage est équivalent à la résolution d'un système linéaire. C'est par exemple le cas des moindres carrés ou de la régression ridge. Nous analysons ici des variantes de cette méthode qui utilisent différentes stratégies de momentum et d'accélération. L'efficacité de ces méthodes dépend de la stratégie de "sketching" utilisée pour compresser l'information du système à résoudre, et ce, à chaque itération. Enfin, nous montrons que ces méthodes peuvent aussi être étendues à d'autres problèmes d'analyse numérique. En effet, l'extension des méthodes de sketch-and-project aux méthodes de direction alternée implicite (ADI) permet de les appliquer en grande dimension lorsque les solveurs classiques s'avèrent trop lents.

Titre : Expected smoothness for stochastic variance reduced methods and sketch-and-project methods for structured linear systems

Keywords : Empirical risk minimization, convex optimizationn, randomized iterative methods, sketch-and-project, sketching

Abstract : The considerable increase in the number of data samples and features complicates the learning phase requiring the minimization of a loss function. Stochastic gradient descent (SGD) and variance reduction variants (SAGA, SVRG, MISO) are widely used to solve this problem. In practice, these methods are accelerated by computing these stochastic gradients on a "mini-batch" : a small group of samples randomly drawn. Indeed, recent technological improvements allowing the parallelization of these calculations have generalized the use of mini-batches.
In this thesis, we are interested in the study of variants of stochastic gradient algorithms with reduced variance by trying to find the optimal hyperparameters : step and mini-batch size. Our study allows us to give convergence results interpolating between stochastic methods drawing a single sample per iteration and the so-called "full-batch" gradient descent using all samples at each iteration. Our analysis is based on the expected smoothness constant which allows to capture the regularity of the random function whose gradient is calculated.
We study another class of optimization algorithms : the "sketch-and-project" methods. These methods can also be applied as soon as the learning problem boils down to solving a linear system. This is the case of ridge regression. We analyse here variants of this method that use different strategies of momentum and acceleration. These methods also depend on the sketching strategy used to compress the information of the system to be solved at each iteration. Finally, we show that these methods can also be extended to numerical analysis problems. Indeed, the extension of sketch-and-project methods to Alternating-Direction Implicit (ADI) methods allows to apply them to large scale problems, when the so-called "direct" solvers are too slow.

Institut Polytechnique de Paris
91120 Palaiseau, France