PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure de Paris

First-Order Optimization and Applications

Soutenue par
Benjamin Paul-Dubois-Taine
Le 6 Novembre 2024

École doctorale n°386
Sciences Mathématiques de Paris Centre

Spécialité
Informatique

Composition du jury :

Martin Jaggi Professeur, EPFL	*Rapporteur*
Sebastian Pokutta Professeur, Technische Universität Berlin	*Rapporteur*
Antonin Chambolle Directeur de recherche, Paris Dauphine	*Président du jury*
Jelena Diakonikolas Professeur, University of Wisconsin-Madison	*Examinatrice*
Alexandre d'Aspremont Directeur de Recherche, CNRS	*Directeur de thèse*

1794
ENS | PSL

Résumé

Les méthodes de gradient conditionnel, également connues sous le nom de méthodes de Frank-Wolfe, sont des algorithmes itératifs de premier ordre pour résoudre des problèmes d'optimisation convexe. Contrairement aux méthodes de descente de gradient, qui reposent sur des projections, les méthodes de gradient conditionnel s'appuient sur une minimisation linéaire sur l'ensemble des contraintes. Cette propriété est particulièrement avantageuse dans de nombreuses applications d'apprentissage automatique et de traitement du signal, où la projection sur l'ensemble des contraintes est souvent soit infaisable, soit trop coûteuse.

Dans la première partie de cette thèse, nous développons un algorithme stochastique proximal accéléré et exploitons des résultats récents sur le lissage de fonctions de support pour concevoir une variante de la méthode de Frank-Wolfe atteignant un taux de convergence accéléré similaire à celui de la descente de gradient accélérée, lorsque des unités de calcul sont disponibles en parallèle pour calculer l'oracle de minimisation linéaire.

Dans la deuxième partie de cette thèse, nous étudions des problèmes d'optimisation non convexes séparables avec des contraintes affines. Nous montrons qu'il est possible d'approcher la valeur duale optimale en exécutant Frank-Wolfe sur l'épigraphe de la relaxation convexe du problème, et que la solution obtenue peut être réduite en utilisant des algorithmes calculant des représentations (approximées) de Carathéodory. Nous montrons que la solution du problème primal ainsi construite atteint la borne classique sur le saut de dualité obtenue par le théorème de Shapley-Folkman, à un terme additif arbitrairement petit près.

Enfin, la dernière partie de cette thèse se concentre sur le problème de déroulement de phase en norme ℓ_1, qui vise à reconstruire une image à partir de son observation modulo 2π. Nous proposons d'utiliser une approche de moindres carrés itérativement repondérés, couplée à un algorithme de gradient conjugué préconditionné. Nous montrons que la méthode obtenue bénéficie d'un taux de convergence sous-linéaire. Nous démontrons à travers de nombreuses expériences que l'implémentation sur processeur graphique (GPU) de cette méthode surpasse les méthodes de déroulement de phase de l'état de l'art.

Abstract

Conditional gradient methods, also known as Frank-Wolfe methods, are first-order iterative algorithms to solve convex optimization problems. As opposed to gradient descent methods, which rely on projections, conditional gradient methods rely on linear minimizations on the feasible set. This property is particularly advantageous in numerous machine learning and signal processing applications, where projecting onto the feasible set is often either intractable or prohibitively expensive.

In the first part of this thesis, we develop a stochastic accelerated proximal algorithm and leverage recent results on smoothed support functions to design a variant of the Frank-Wolfe method achieving accelerated rates similar to those of accelerated gradient descent, when computing units are available in parallel to compute the linear minimization oracle.

In the second part of this thesis, we study nonconvex separable optimization problems with affine constraints. We show that one can approximate the optimal dual value by running Frank-Wolfe on the epigraph of the convex relaxation of the problem, and that the obtained solution can be trimmed down using algorithms computing (approximate) Carathédory representations. We show that this yields a solution to the primal problem achieving the classical duality gap bound obtained by the Shapley-Folkman theorem up to an arbitrarily small additive term.

Finally, the last part of this thesis focuses on the ℓ_1-norm phase unwrapping problem, which aims at recovering an image from its observation modulo 2π. We use the iteratively reweighted least squares approach coupled with a preconditioned conjugate gradient algorithm. We show that the resulting method enjoys a sublinear convergence rate. We demonstrate through extensive experiments that the resulting GPU implementation outperforms state of the art phase unwrapping methods.

Remerciements

Je tiens d'abord à remercier mon superviseur de thèse Alexandre d'Aspremont. Sa culture scientifique et son perpétuel enthousiasme m'ont beaucoup aidé à persévérer dans mes projets de recherche. Je le remercie aussi d'avoir porté à mon attention le théorème de Shapley-Folkman et ses différentes applications, qui ont donné fruit au projet le plus enrichissant de ma thèse.

I warmly thank the jury for agreeing to be part of my Ph.D. committee and for attending the defense. In particular, I thank Martin Jaggi and Sebastian Pokutta for a detailed feedback on my manuscript. I also thank Antonin Chambolle for accepting to be president of the jury, and for our subsequent interactions, which have led to some very interesting research directions. Finally, I thank Jelena Diakonikolas for her insightful questions that followed the defense.

Le premier projet de ma thèse a été réalisé sous la supervision de Francis Bach, Quentin Berthet et Adrien Taylor. Je les remercie pour leur guidance pendant cette année pleine de découvertes. Je tiens particulièrement à remercier Adrien pour la patience dont il a fait preuve pendant nos longues réunions, et pour les nombreuses conversations, scientifiques ou autres, que nous avons pu avoir depuis.

My first research experience was conducted with Tim Hoheisel at McGill University. I warmly thank him for his kind but rigorous introduction to optimization, for the many advice he has given me since my Montreal days, and for the endless conversations we have had about basketball. I would also like to thank Mark Schmidt and Sharan Vaswani for supervising me while I was a masters student in Vancouver, and for teaching me the rigor required to pursue academic research.

Ma thèse n'aurait pas été la même sans l'excellente ambiance qui règne dans les équipes Sierra et Willow de l'Inria. Je remercie tout particulièrement Céline, Louis, Marguerite, Oumayma, Quentin et Yann pour nos innombrables déjeuners ensemble et nos sorties nocturnes au Ground Control ou à la Buttes-aux-Cailles. Je remercie aussi Antoine, Bruce, Bertille, Corbi, Etienne, Fabian, Frederike, Sarah, Théo, Wilson et tous les autres pour nos petites interactions du quotidien qui ont rendu ces 3 années si enrichissantes. L'esprit de camaraderie chez Willow-Sierra est rendu possible par la bienveillance des directeurs d'équipe, et pour cela je remercie Francis Bach et Justin Carpentier.

Un centre de recherches n'est pas uniquement fait de chercheurs, et je tiens en premier lieu à remercier Lucile et Anaïs du service communication pour leur constante bonne humeur, et leur aide lorsque nous autres doctorants essayions d'organiser des évènements à l'Inria. Je remercie aussi Sandra pour avoir toujours eu réponse à mes questions sur le fonctionnement de l'Inria, et Lise-Marie du CNRS pour sa réactivite et son accompagnement lorsque je me perdais dans les méandres des démarches administratives de l'ENS.

Je suis profondément reconnaissant envers ma famille, et tout particulièrement mes parents et mes frères, qui m'a toujours encouragé dans la poursuite de mon parcours académique. Même si nous sommes souvent éloignés géographiquement, leur soutien et leurs conseils me sont extrêmement précieux.

Je remercie enfin les amis qui m'entourent, et qui souvent ont fait l'effort de s'intéresser à mes travaux de thèse. Je pense à mes amis d'enfance, de collège et de lycée, à mes amis de prépa et de Montréal, et enfin aux belles rencontres que j'ai faites depuis mon retour à Paris, notamment mon groupe adoptif des "ponts". Pour finir, je remercie tout particulièrement Perrine, qui a été d'un incroyable support émotionnel lors de la deuxième partie de cette thèse.

Contents

Résumé i

Abstract ii

Remerciements iii

1 Introduction 1
- 1.1 Introduction to optimization . . . 1
  - 1.1.1 General notation . . . 2
  - 1.1.2 Regularity assumptions . . . 2
- 1.2 Projected Gradient Descent . . . 3
  - 1.2.1 Convergence rates . . . 3
  - 1.2.2 Measuring complexity and lower bounds . . . 4
  - 1.2.3 Acceleration . . . 5
- 1.3 The Frank-Wolfe algorithm . . . 6
  - 1.3.1 Convergence rate . . . 7
  - 1.3.2 Properties of Frank-Wolfe . . . 9
  - 1.3.3 The strongly convex case . . . 10
- 1.4 Variants of Frank-Wolfe . . . 11
  - 1.4.1 Away steps . . . 12
  - 1.4.2 Pairwise steps . . . 13
  - 1.4.3 Fully corrective . . . 14
  - 1.4.4 Linear convergence on polytopes . . . 14
- 1.5 Further enhancements of Frank-Wolfe methods . . . 15
  - 1.5.1 Approximate LMO . . . 15
  - 1.5.2 Stochastic setting . . . 15
  - 1.5.3 Nonconvex objectives . . . 16
  - 1.5.4 Stronger linear oracle . . . 16
  - 1.5.5 Restart schemes . . . 16
  - 1.5.6 Gradient sliding and lazy updates . . . 16
  - 1.5.7 Generalized conditional gradient method . . . 17
- 1.6 Applications of Frank-Wolfe . . . 17
- 1.7 Thesis outline and contributions . . . 19

2 Acceleration of Frank-Wolfe under Parallelization 20
- 2.1 Introduction . . . 20
  - 2.1.1 Related Work . . . 21
- 2.2 Notation and definitions . . . 22
- 2.3 Fast Stochastic Composite Minimization . . . 23
- 2.4 Accelerating Frank-Wolfe with parallelization . . . 25
- 2.5 Experiments . . . 28

Appendix 2.A Proofs for Stochastic Composite Minimization . . . 31

2.A.1 Proof of Proposition 2.3.1 . . . 31
2.A.2 Proof of Theorem 2.3.1 . . . 35
Appendix 2.B Proofs for Accelerated Frank-Wolfe . . . 36
2.B.1 Proof of Theorem 2.4.1 . . . 36
2.B.2 Proof of Theorem 2.4.2 . . . 39
2.B.3 Dependence on the norms . . . 39
Appendix 2.C Experimental Details . . . 42

3 Nonconvex optimization via Frank-Wolfe & Carathéodory 44
3.1 Introduction . . . 45
3.1.1 Related work . . . 45
3.1.2 Contributions . . . 45
3.1.3 Chapter organization . . . 46
3.2 Preliminaries . . . 46
3.2.1 Dual and convex relaxation . . . 47
3.2.2 Carathéodory . . . 47
3.2.3 Shapley Folkman and duality gap estimation . . . 52
3.3 Solving nonconvex separable problems . . . 54
3.3.1 Obtaining primal points . . . 54
3.3.2 Trimming the solution . . . 57
3.3.3 Final reconstruction . . . 59
3.4 The special case of mixed integer linear programs . . . 65
3.5 Numerical results . . . 66
3.5.1 Computing the conjugate . . . 67
3.5.2 Linear minimization over the domain . . . 68
3.5.3 Results . . . 68
Appendix 3.A Additional proofs . . . 70
3.A.1 Proof of Theorem 3.2.3 . . . 70
3.A.2 Proof of Theorem 3.2.7 . . . 70
3.A.3 Proof of Theorem 3.2.8 . . . 71
3.A.4 Proof of Theorem 3.2.9 . . . 71
3.A.5 Proof of Lemma 3.3.3 . . . 72
3.A.6 Proof of Theorem 3.3.3 . . . 73
3.A.7 Proof of Theorem 3.3.6 . . . 75
3.A.8 Proof of Theorem 3.3.7 . . . 76
Appendix 3.B Auxiliary lemma . . . 78
Appendix 3.C Technical lemmas . . . 79
Appendix 3.D Experimental details . . . 81

4 Iteratively Reweighted Least Squares for Phase Unwrapping 83
4.1 Introduction . . . 83
4.1.1 Related Work . . . 84
4.1.2 Contributions . . . 85
4.2 Notations and definitions . . . 86
4.3 Model . . . 87
4.4 Algorithm . . . 88
4.4.1 Iteratively reweighted least squares . . . 88
4.4.2 Conjugate gradient . . . 93
4.4.3 Choosing the preconditioner . . . 94
4.5 Experiments . . . 99
4.5.1 Real Data Acquisition and Synthetic Data Generation . . . 100

4.5.2 Weight computation . . . 101
4.5.3 Number of CG iterations and stopping criterion . . . 101
4.5.4 Code details . . . 102
4.5.5 Results . . . 104
Appendix 4.A Matrix visualization . . . 107
Appendix 4.B Proof of Proposition 4.4.3 . . . 107
Appendix 4.C Additional experiments . . . 109
4.C.1 Simulated images . . . 109
4.C.2 Real images with Goldstein filtering . . . 114

Conclusion and perspectives **118**

Résumé en français **134**

Chapter

1 Introduction

Chapter content

1.1 Introduction to optimization . . . 1
1.1.1 General notation . . . 2
1.1.2 Regularity assumptions . . . 2
1.2 Projected Gradient Descent . . . 3
1.2.1 Convergence rates . . . 3
1.2.2 Measuring complexity and lower bounds . . . 4
1.2.3 Acceleration . . . 5
1.3 The Frank-Wolfe algorithm . . . 6
1.3.1 Convergence rate . . . 7
1.3.2 Properties of Frank-Wolfe . . . 9
1.3.3 The strongly convex case . . . 10
1.4 Variants of Frank-Wolfe . . . 11
1.4.1 Away steps . . . 12
1.4.2 Pairwise steps . . . 13
1.4.3 Fully corrective . . . 14
1.4.4 Linear convergence on polytopes . . . 14
1.5 Further enhancements of Frank-Wolfe methods . . . 15
1.5.1 Approximate LMO . . . 15
1.5.2 Stochastic setting . . . 15
1.5.3 Nonconvex objectives . . . 16
1.5.4 Stronger linear oracle . . . 16
1.5.5 Restart schemes . . . 16
1.5.6 Gradient sliding and lazy updates . . . 16
1.5.7 Generalized conditional gradient method . . . 17
1.6 Applications of Frank-Wolfe . . . 17
1.7 Thesis outline and contributions . . . 19

1.1 Introduction to optimization

The need for 'optimal' solutions is, in one way or another, at the heart of most modern problems in engineering, economics and computational sciences. Mathematically speaking, an optimization problem can be written as

$$\begin{aligned} &\text{minimize} \quad f(x) \\ &\text{such that} \quad x \in \mathcal{C}, \end{aligned} \tag{1.1}$$

where f is commonly referred to as the *objective function* and $\mathcal{C}$ as the *feasible set*. Most algorithms to solve problems of the type (1.1) fall within the category of *iterative* methods; they start from some guess of the solution x_0, and iteratively refine that guess as they gain more information on the problem, leading to a sequence of guesses $x_0, x_1, x_2, \dots$. What differentiates iterative methods is precisely what type of information they have access to, and how they refine their guess based on this information. In this thesis, we are mostly interested in *first-order* methods. Those methods update their guess based on the gradient ∇f of the objective function, as opposed to, for example, *second-order* methods, which rely on information from the hessian $\nabla^2 f$. From now on, we assume that the objective function f is differentiable. We also assume that problem (1.1) has a solution, i.e. there exists $x^* \in \mathcal{C}$ such that $x^* \in \arg\min_{x \in \mathcal{C}} f(x)$.

In the rest of this section, we introduce some classical notions like convexity and smoothness of the objective function, some of the main mathematical tools used to study first-order methods.

1.1.1 General notation

We consider the finite-dimensional vector space $\mathbb{R}^d$ equipped with the standard Euclidean inner product $\langle \cdot, \cdot \rangle : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, and a norm $\|\cdot\| : \mathbb{R}^d \to \mathbb{R}$. We do not assume that the norm is induced by the inner product. We also consider the corresponding dual norm $\|\cdot\|_* : \mathbb{R}^d \to \mathbb{R}$ induced by the choice of $\|\cdot\|$ using $\|z\|_* = \sup_{\|x\| \leq 1} \langle z, x \rangle$. We denote the Euclidean norm $\|\cdot\|_2$, defined as $\|x\|_2 = \sqrt{\langle x, x \rangle}$. Given a set $\mathcal{C} \subset \mathbb{R}^d$, we denote its diameter with respect to (w.r.t.) the norm $\|\cdot\|$ as $\operatorname{diam}(\mathcal{C}, \|\cdot\|) = \sup_{x,y \in \mathcal{C}} \|x - y\|$.

Note that all results derived in this section also apply to the space of matrices $\mathbb{R}^{p \times q}$ equipped with the Euclidean inner product over matrices.

1.1.2 Regularity assumptions

We first introduce the notion of convexity of a set.

Definition 1.1.1. *A set $\mathcal{C} \subset \mathbb{R}^d$ is convex if*

$$\forall x, y \in \mathcal{C}, \forall \lambda \in [0, 1], \ \lambda x + (1 - \lambda) y \in \mathcal{C}.$$

Definition 1.1.2. *A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if*

$$\forall x, y \in \mathbb{R}^d, \forall \lambda \in [0, 1], \ f(\lambda x + (1 - \lambda) y) \leq \lambda f(x) + (1 - \lambda) f(y).$$

We say that f is strictly convex if the inequality is strict whenever $x \neq y$ and $\lambda \in (0, 1)$.

The above definition is a zeroth-order definition of convexity. If the function f is differentiable, one can also get a first-order characterization of convexity.

Proposition 1.1.1. *A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if and only if*

$$\forall x, y \in \mathbb{R}^d, \ f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle.$$

Proof. See for example [BV04, Section 3.1.3]. □

In view of the above result, a function is convex if at any point the function lies above its first-order approximation. An important consequence of convexity is that all local minima of f are global minima. This is a major advantage for first-order iterative methods as it guarantees that those methods cannot get 'stuck' in local minima. The notion of convexity can also be strengthened to incorporate a quadratic term in the lower bound.

Definition 1.1.3. *A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is said to be μ-strongly convex w.r.t. $\|\cdot\|$ if*

$$\forall x, y \in \mathbb{R}^d, \ f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle + \frac{\mu}{2} \|x - y\|^2 .$$

In order to study first-order methods, a common assumption is that of *smoothness* of the function f. Mathematically speaking, smoothness means that the gradient of f is Lipschitz (see for example [Nes03]). Equivalently here, we define smoothness as the existence of a quadratic upper bound on the function f. We do so because it is the most useful version when proving convergence of first-order methods.

Definition 1.1.4. *A differentiable function $f : \mathbb{R}^d \to \mathbb{R}$ is L-smooth w.r.t. $\|\cdot\|$ if*

$$\forall x, y \in \mathbb{R}^d, \ f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2} \|x - y\|^2 . \tag{1.2}$$

1.2 Projected Gradient Descent

In the unconstrained case, namely when $\mathcal{C} = \mathbb{R}^d$, a natural way to refine the current guess x_k is by taking a step in the direction of the negative gradient,

$$x_{k+1} = x_k - \eta_k \nabla f(x_k), \quad k = 0, 1, \ldots \tag{1.3}$$

where $\eta_k > 0$ controls how big the step in the descent direction should be, and is commonly referred to as the *stepsize* (the term *learning rate* is also often used in the machine learning literature). When the feasible set $\mathcal{C}$ is not the full space, update (1.3) might yield a point outside of the feasible set. One way to deal with this is to *project* the point onto $\mathcal{C}$.

Definition 1.2.1. *Given $\tilde{x} \in \mathbb{R}^d$ and a closed set $\mathcal{C} \subset \mathbb{R}^d$, the projection of $\tilde{x}$ onto $\mathcal{C}$ w.r.t. $\|\cdot\|$, denoted as $\Pi_{\mathcal{C}}(\tilde{x})$, is defined as*

$$\Pi_{\mathcal{C}}(\tilde{x}) = \arg\min_{x \in \mathcal{C}} \ \frac{1}{2} \|x - \tilde{x}\|^2 . \tag{1.4}$$

For ease of exposition, we assume throughout this section (Section 1.2) that we are working with the Euclidean norm $\|\cdot\|_2$.

The projection is always well-defined, meaning that the minimum is always attained in $\mathcal{C}$. The projection is also unique whenever $\mathcal{C}$ is convex [BV04]. The update rule (1.3) can then be made feasible by adding a projection step. This yields the projected gradient descent (PGD) algorithm, which we present in Algorithm 1. The influence of this method on the field of numerical mathematics cannot be overstated, as it is still to this day one of the main building blocks for many optimization algorithms. The algorithm was first introduced by Louis-Augustin Cauchy, who published it in his *Compte Rendu à l'Académie des Sciences* in 1847 (see [C+47] for the original paper, and also [Lem12] for a modern discussion of Cauchy's original work).

Algorithm 1 Projected Gradient Descent

Input: $x_0 \in \mathcal{C}$.
for $k = 0, 1, \ldots$ **do**

$$x_{k+1} = \Pi_{\mathcal{C}}(x_k - \eta_k \nabla f(x_k)) \tag{1.5}$$

end for

1.2.1 Convergence rates

We now give the convergence rate of the method. Note that projected gradient methods for general norms can also be studied, and the practicability of the resulting algorithms as well as the convergence rates heavily depend on the choice of the norm (see for example [Dev13]).

Theorem 1.2.1. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and L-smooth w.r.t. $\|\cdot\|_2$ and $\mathcal{C} \subset \mathbb{R}^d$ be closed convex. The iterates of Algorithm 1 with input $x_0 \in \mathcal{C}$ and constant stepsize $\eta_k = \frac{1}{L}$ satisfy*

$$f(x_k) - f(x^*) \leq \frac{L \|x_0 - x^*\|_2^2}{2k}.$$

Proof. See for example [Bec17, Chapter 10]. □

The above result can be strengthened when the objective is strongly convex.

Theorem 1.2.2. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be μ-strongly convex and L-smooth w.r.t. $\|\cdot\|_2$ and $\mathcal{C} \subset \mathbb{R}^d$ be closed convex. The iterates of Algorithm 1 with input $x_0 \in \mathcal{C}$ and constant stepsize $\eta_k = \frac{1}{L}$ satisfy*

$$\|x_k - x^*\|_2^2 \leq \left(1 - \frac{\mu}{L}\right)^k \|x_0 - x^*\|_2^2.$$

Proof. See for example [Bec17, Chapter 10]. □

Remark 1.2.1. *When the smoothness constant is unknown, rates similar to those of Theorem 1.2.1 and Theorem 1.2.2 can be achieved by replacing the constant stepsize setting with a so-called backtracking linesearch, a technique which requires no knowledge of L. For more details, see [BT09, Bec17].*

1.2.2 Measuring complexity and lower bounds

While the projected gradient descent algorithm is a natural method, one might wonder whether faster gradient methods may be designed. To answer this, one needs to define what it means for an algorithm to be fast. For projected gradient methods, the complexity is computed as the number of times the method needs to compute the gradient as well as the number of projections required in order to achieve a given precision level $\epsilon > 0$. As an example, the complexity of Algorithm 1 on smooth convex objectives is $O(1/\epsilon)$ both in terms of gradient evaluations and projections. Indeed, to obtain $f(x_k) - f(x^*) \leq \epsilon$, Theorem 1.2.1 tells us that the number of iterations k, and thus the number of gradients and projections, should sastisfy

$$\frac{L \|x_0 - x^*\|_2^2}{2k} \leq \epsilon \iff k \geq \frac{L \|x_0 - x^*\|_2^2}{2\epsilon} = O\left(\frac{1}{\epsilon}\right).$$

Similarly, the complexity of Algorithm 1 on smooth strongly convex objectives is $O\left(\frac{L}{\mu} \log \frac{1}{\epsilon}\right)$, as one can use Theorem 1.2.2 and the classic inequality $(1 + z) \leq e^z$ for any $z \in \mathbb{R}$ to show that k should satisfy

$$\exp\left(-k\frac{\mu}{L}\right) \|x_0 - x^*\|_2^2 \leq \epsilon \iff k \geq \frac{L}{\mu} \log\left(\frac{\|x_0 - x^*\|_2^2}{\epsilon}\right) = O\left(\frac{L}{\mu} \log \frac{1}{\epsilon}\right).$$

The next two theorems give lower bounds on the performance of first-order methods in the unconstrained case. Note that finding lower bounds of algorithms is a large subfield of optimization research, and a thorough presentation is outside the scope of this introduction.

Theorem 1.2.3 ([Nes03])**.** *For any $L > 0$ and $x_0 \in \mathbb{R}^d$ and any k such that $1 \leq k \leq \frac{d-1}{2}$, there exists a function f, convex and L-smooth convex w.r.t. $\|\cdot\|_2$, such that for any algorithm whose output is of the form*

$$x_k \in x_0 + span\{\nabla f(x_0), \ldots, \nabla f(x_{k-1})\},$$

the following lower bound holds,

$$f(x_k) - f(x^*) \geq \frac{3L \|x_0 - x^*\|_2^2}{32(k+1)^2}.$$

A similar lower bound for strongly convex functions can also be derived.

Theorem 1.2.4 ([Nes03]). *For any $L > \mu > 0$, for any $x_0 \in \mathbb{R}^d$, there exists function f, L-smooth and μ-strongly convex w.r.t. $\|\cdot\|_2$, such that for any algorithm whose output is of the form*

$$x_k \in x_0 + span\{\nabla f(x_0), \dots, \nabla f(x_{k-1})\},$$

the following lower bound holds,

$$\|x_k - x^*\|_2^2 \geq \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^{2k} \|x_0 - x^*\|_2^2,$$

where $\kappa = \frac{L}{\mu}$.

Switching to the ϵ notation, those two results tell us that gradient methods cannot achieve better than complexity $O(1/\sqrt{\epsilon})$ and $O(\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$ on convex and strongly convex objectives, respectively. It turns out that a slight modification to projected gradient descent yields an algorithm achieving those rates, through a mechanism often referred to as *acceleration*.

1.2.3 Acceleration

In order to accelerate the projected gradient descent method and match the lower bounds of the previous section, it is necessary to add an extrapolation step to the iterates of Algorithm 1. This is often commonly referred to as *Nesterov acceleration*, *momentum* or simply *acceleration*, and was first introduced in [Nes83] for convex functions. We describe the method in Algorithm 2.

Algorithm 2 Accelerated Projected Gradient Descent

Input: $x_0 \in \mathcal{C}$.
Set $y_0 = x_0$.
for $k = 0, 1, \dots$ **do**

$$x_{k+1} = \Pi_{\mathcal{C}}\left(y_k - \eta_k \nabla f(y_k)\right) \tag{1.6}$$
$$y_{k+1} = x_{k+1} + \beta_{k+1}(x_{k+1} - x_k) \tag{1.7}$$

end for

The next theorem gives a convergence rate which matches (up to constants) the lower bounds of the previous section under appropriate choices of the stepsize and the momentum parameter.

Theorem 1.2.5. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be convex and L-smooth w.r.t. $\|\cdot\|_2$ and $\mathcal{C} \subset \mathbb{R}^d$ be closed convex. Consider Algorithm 2 with input $x_0 \in \mathcal{C}$, constant stepsize $\eta_k = 1/L$ and momentum parameter β_{k+1} defined as*

$$\begin{aligned} t_{k+1} &= \frac{1+\sqrt{1+4t_k^2}}{2}, \quad k = 0, 1, \dots, \\ \beta_{k+1} &= \frac{t_k - 1}{t_{k+1}} \end{aligned}$$

with $t_0 = 0$. Then, the iterates satisfy

$$f(x_k) - f(x^*) \leq \frac{2L\|x_0 - x^*\|_2^2}{(k+1)^2}.$$

Suppose now that f is also μ-strongly convex w.r.t. $\|\cdot\|_2$, set $\kappa = L/\mu$ and the momentum parameter $\beta_k = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$. Then, the iterates satisfy

$$f(x_k) - f(x^*) \leq \left(1 - \frac{1}{\sqrt{\kappa}}\right)^k \left(f(x_0) - f(x^*) + \frac{\mu}{2}\|x_0 - x^*\|_2^2\right).$$

Proof. See for example [BT09, Bec17]. □

1.3 The Frank-Wolfe algorithm

In some practical cases, the projection operator (1.4) can be intractable, or too costly to compute. The Frank-Wolfe algorithm is one way to avoid projections by replacing the quadratic oracle (1.4) by a linear one. It was first introduced by Marguerite Frank and Philip Wolfe [FW56]. The Frank-Wolfe method is also often called conditional gradient method. However, for ease of exposition, we refer to Frank-Wolfe as the original algorithm from [FW56], and refer to conditional gradient methods (or Frank-Wolfe variants) as all first-order methods relying on linear minimization oracles (see Section 1.4 for more details).

Let us first motivate the Frank-Wolfe algorithm by rewriting the projected gradient descent update in the Euclidean setting. As we saw in Section 1.2, given $x_k \in \mathcal{C}$, the gradient descent update sets $x_{k+1} = x^+$ where x^+ is given by

$$\begin{aligned}
& x^+ = \Pi_{\mathcal{C}}\left(x_k - \eta_k \nabla f(x_k)\right) \\
\iff & x^+ \in \arg\min_{x\in\mathcal{C}} \ \frac{1}{2}\|x - (x_k - \eta_k \nabla f(x_k))\|_2^2 \\
\iff & x^+ \in \arg\min_{x\in\mathcal{C}} \ \frac{\eta_k^2}{2}\|\nabla f(x_k)\|_2^2 + \eta_k\langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2}\|x - x_k\|_2^2 \\
\iff & x^+ \in \arg\min_{x\in\mathcal{C}} \ f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2\eta_k}\|x - x_k\|_2^2,
\end{aligned}$$

where in the last equivalence we removed and added terms which do not depend on x. The last equivalence tells us that at each iteration k, PGD minimizes a quadratic approximation of f at x_k. This way of writing the PGD update is very common and allows for intuitive generalizations to any norms [Dev13]. The motivation behind conditional gradient method is to remove the quadratic term and simply compute the minimization of the linear approximation of f at x_k,

$$x^+ \in \arg\min_{x\in\mathcal{C}} \ f(x_k) + \langle \nabla f(x_k), x - x_k\rangle, \tag{1.8}$$

or, removing unnecessary terms,

$$x^+ \in \arg\min_{x\in\mathcal{C}} \ \langle \nabla f(x_k), x\rangle. \tag{1.9}$$

The above computation is commonly referred to as the *linear minimization oracle*.

Definition 1.3.1. *Given a convex compact set $\mathcal{C} \subset \mathbb{R}^d$, the linear minimization oracle (or LMO) at $y \in \mathbb{R}^d$ is defined as*

$$LMO(y) \in \arg\min_{x\in\mathcal{C}} \ \langle y, x\rangle. \tag{1.10}$$

Given x_k and the result of the LMO computation step x^+, the Frank-Wolfe method sets the next iterate as a convex combination of x^+ and x_k. We summarize the method in Algorithm 3.

Algorithm 3 Frank-Wolfe

Input: $x_0 \in \mathcal{C}$
for $k = 0, 1, \dots$ **do**
  $s_k = \mathrm{LMO}(\nabla f(x_k))$
  (Variant 1): $\eta_k = \frac{2}{k+2}$
  (Variant 2): $\eta_k \in \arg\min_{\eta \in [0,1]} f(x_k + \eta(s_k - x_k))$
  (Variant 3): $\eta_k = \min\left\{1, \frac{\langle \nabla f(x_k), x_k - s_k \rangle}{C_f}\right\}$
  $x_{k+1} = (1 - \eta_k)x_k + \eta_k s_k$
end for

The projection operator (1.4) is a quadratic minimization over the feasible set, while the LMO (1.10) is a linear minimization over the feasible set. In many practical applications, the linear minimization is cheaper than the projection; these include, for example, ℓ_p-norm balls for $p \in]1, 2[\cup]2, +\infty[$, the nuclear norm ball, many polytopes of interest, and the Total Variation norm ball. For more on the computational benefits of linear minimizations over projections, we refer the reader to [HK12, HJN15, CP21].

1.3.1 Convergence rate

In contrast with projected gradient methods, whose analysis is often based on the smoothness assumption, conditional gradient methods are commonly studied using the notion of *curvature* of the objective function [Cla10, Jag13, LJJ13]. Intuitively, the curvature bounds the deviation of the function f from its first order approximation on a specific line segment.

Definition 1.3.2. *The curvature C_f of the function f on the set $\mathcal{C}$ is defined as*

$$C_f = \sup_{\substack{x,v \in \mathcal{C},\ \eta \in [0,1] \\ y = x + \eta(v-x)}} \left\{ \frac{2}{\eta^2} \left(f(y) - f(x) - \langle \nabla f(x), y - x \rangle \right) \right\}.$$

The bounded curvature assumption is weaker than the smoothness assumption, as if f is L-smooth w.r.t. $\|\cdot\|$, one always has (see [Jag13] for a proof)

$$C_f \leq L \mathrm{diam}(\mathcal{C}, \|\cdot\|)^2. \tag{1.11}$$

We now give the main convergence results for Algorithm 3. The proof is a very classic result in convex optimization (see for example [Cla10, Jag13]), and we provide it for completeness.

Theorem 1.3.1. *Let $\mathcal{C}$ be compact and convex and $f : \mathbb{R}^d \to \mathbb{R}$ be convex with curvature C_f. The iterates of Algorithm 3 satisfy*

$$f(x_k) - f(x^*) \leq \frac{2C_f}{k+1}$$

Proof. Let $\alpha_t = \frac{2}{t+2}$. We claim that for any of the variants, we have

$$f(x_{t+1}) \leq f(x_t) - \alpha_t \left(f(x_t) - f(x^*) \right) + \frac{\alpha_t^2 C_f}{2}. \tag{1.12}$$

Indeed, from the definition of C_f we have

$$\begin{aligned} f(x_{t+1}) = f(x_t + \eta_t(s_t - x_t)) &\leq f(x_t) + \langle \nabla f(x_t), x_{t+1} - x_t \rangle + \frac{\eta_t^2 C_f}{2} \\ &= f(x_t) + \eta_t \langle \nabla f(x_t), s_t - x_t \rangle + \frac{\eta_t^2 C_f}{2} \\ &= f(x_t) + \eta_t \min_{s \in \mathcal{C}} \langle \nabla f(x_t), s - x_t \rangle + \frac{\eta_t^2 C_f}{2} \\ &\leq f(x_t) + \eta_t \langle \nabla f(x_t), x^* - x_t \rangle + \frac{\eta_t^2 C_f}{2} \\ &\leq f(x_t) - \eta_t (f(x_t) - f(x^*)) + \frac{\eta_t^2 C_f}{2}, \end{aligned}$$

where the last inequality comes from first-order characterization of convexity. For Variant 1, $\eta_t = \alpha_t$ so (1.12) is verified. For Variant 2, η_t is set so as to minimize $f(x_t + \eta_t(s_t - x_t)$ and therefore is better than any other updates, so (1.12) is also verified. For Variant 3, η_t is set so as to minimize the quadratic upper bound in the last inequality, and therefore is better than the choice $\frac{2}{t+2}$, and so (1.12) is also verified.

Let $\delta_t = a_t(f(x_t) - f(x^*))$ where $a_t > 0$ will be determined later in the proof. From (1.12), we get

$$\delta_{t+1} - \delta_t \leq (a_{t+1}(1 - \alpha_t) - a_t)\,(f(x_t) - f(x^*)) + a_{t+1}\frac{\alpha_t^2 C_f}{2}.$$

Choosing $a_t = \frac{t(t+1)}{2}$, we get

$$a_{t+1}(1 - \alpha_t) - a_t = \frac{(t+1)(t+2)}{2}(1 - \frac{2}{t+2}) - \frac{t(t+1)}{2} = 0,$$

and

$$a_{t+1}\frac{\alpha_t^2}{2} = \frac{(t+1)(t+2)}{2} \times \frac{4}{2(t+2)^2} \leq 1.$$

Putting everything together, we get

$$\delta_{t+1} - \delta_t \leq C_f.$$

By summing from 0 to $k-1$, we get

$$\sum_{t=0}^{k-1} \delta_{t+1} - \delta_t = \delta_k - \delta_0 \leq k C_f.$$

Since $\delta_k = \frac{k(k+1)}{2}(f(x_k) - f(x^*))$ and $\delta_0 = 0$, we get

$$f(x_k) - f(x^*) \leq \frac{2C_f}{k+1}.$$

□

Remark 1.3.1. *Among all the stepsize strategies described in Algorithm 3, Variant 1 is the simplest to implement and does not require any knowledge about the underlying problem. On the other hand, Variant 2 requires solving a one-dimensional optimization problem, whose difficulty depends on the application at hand. Finally, Variant 3 requires knowledge of the curvature*

constant, which might be inaccessible in some applications. Note that (1.11), coupled with Theorem 1.3.1, tells us that if f is L-smooth w.r.t. $\|\cdot\|$, the iterates of Algorithm 3 with either variant satisfy

$$f(x_k) - f(x^*) \leq \frac{2L\,diam(\mathcal{C}, \|\cdot\|)^2}{k+1}, \tag{1.13}$$

which is comparable to the rate of the projected gradient descent algorithm derived in Theorem 1.2.1. In case of smoothness, Variant 3 is often replaced with $\eta_k = \min\left\{1, \frac{\langle \nabla f(x_k), x_k - s_k\rangle}{L\|s_k - x_k\|^2}\right\}$, as L might be easier to obtain than C_f. This stepsize strategy also yields the convergence rate (1.13) [PANJ18].

1.3.2 Properties of Frank-Wolfe

We now discuss a few properties of the Frank-Wolfe algorithm which are not shared by projected gradient descent methods and which, in addition to avoiding projections, make it appealing in various settings.

1.3.2.1 Norm independence

While projected descent methods heavily depend on the choice of the norm for both the algorithm implementation and the resulting convergence rates, this is not the case for Frank-Wolfe. Indeed, the LMO is norm-independent, and so is the curvature C_f. Moreover, the rate derived in (1.13) holds for any norm. In particular, one can choose the norm so as to obtain the tightest rate, i.e. the smallest value of $L\text{diam}(\mathcal{C}, \|\cdot\|)^2$.

1.3.2.2 Affine invariance

Another important property of the Frank-Wolfe algorithm, which is once again not shared by projected gradient methods, is that of invariance under affine transformation of the problem. Let A be a nonsingular matrix and consider the affine translated problem

$$\begin{aligned} &\text{minimize} \quad \hat{f}(y) \\ &\text{such that} \quad y \in \hat{\mathcal{C}}, \end{aligned} \tag{1.14}$$

where $\hat{f}(y) = f(Ay)$ and $\hat{\mathcal{C}} = A^{-1}\mathcal{C}$. For $x_0 \in \mathcal{C}$, one can check that running Frank-Wolfe on the translated problem (1.14) with input $y_0 = A^{-1}x_0$ is equivalent to running Frank-Wolfe on the original problem (1.1) with starting point x_0. Moreover, the convergence rate derived from Theorem 1.3.1 is unchanged, as the curvature is also affine invariant. Indeed,

$$\begin{aligned} C_{\hat{f}} &= \sup_{\substack{x,v \in A^{-1}\mathcal{C},\ \eta\in[0,1] \\ y = x + \eta(v-x)}} \left\{ \frac{2}{\eta^2}\left(\hat{f}(y) - \hat{f}(x) - \langle \nabla \hat{f}(x), y - x\rangle\right)\right\} \\ &= \sup_{\substack{x,v \in A^{-1}\mathcal{C},\ \eta\in[0,1] \\ y = x + \eta(v-x)}} \left\{ \frac{2}{\eta^2}\left(f(Ay) - f(Ax) - \langle A^\top \nabla f(Ax), y - x\rangle\right)\right\} \\ &= \sup_{\substack{x,v \in A^{-1}\mathcal{C},\ \eta\in[0,1] \\ y = x + \eta(v-x)}} \left\{ \frac{2}{\eta^2}\left(f(Ay) - f(Ax) - \langle \nabla f(Ax), A(y - x)\rangle\right)\right\} \\ &= C_f. \end{aligned}$$

In contrast, the smoothness constant L and the strong convexity parameter μ of a given function are not affine invariant. In particular, the usual rates derived for projected gradient descent in Section 1.2 are not affine invariant.

1.3.2.3 Extreme points and sparsity

An extreme point $p \in \mathcal{C}$ is a point which cannot be written as $p = \lambda x + (1-\lambda)y$ for $x, y \in \mathcal{C}$, $0 < \lambda < 1$ and $x \neq y$. Such points are of particular importance for Frank-Wolfe methods, as there always is an extreme point which is a solution to the linear minimization oracle. If the Frank-Wolfe algorithm is initialized properly (namely, at an extreme point of $\mathcal{C}$), then the early iterates of the algorithm are *sparse* convex combinations of extreme points of $\mathcal{C}$, also called *atoms*. Indeed, each iteration of Frank-Wolfe adds at most one atom to the convex combination. This is a useful property, for example for low-rank matrix completion [SSGS11, FGM17], or for computing approximate convex representation of a given point [CP23].

1.3.2.4 Stopping criterion and duality gap convergence

Theorem 1.3.1 gives a rate on the convergence of Frank-Wolfe, but in most applications one does not have access to either the minimum $f(x^*)$ or the curvature C_f, and it may thus seem difficult to decide when to actually stop the algorithm. For projected gradient method, one usually stops after a fixed number of iterations, or when the iterations stop making significant progress. Yet this can be unsatisfying. For the Frank-Wolfe algorithm on the other hand, the *Frank-Wolfe gap* is given by

$$g_k = \langle \nabla f(x_k), x_k - s_k \rangle. \tag{1.15}$$

The gap is an upper bound on the suboptimality at iteration k. In fact, the Frank-Wolfe gap is equal to the primal-dual gap at $(x_k, \nabla f(x_k))$. Indeed, Fenchel duality [Roc15] tells us that the dual of problem (1.1) is

$$\max_{y \in \mathbb{R}^d} \; -f^*(y) - \sigma_{\mathcal{C}}(-y),$$

where f^* is the convex conjugate of f and $\sigma_{\mathcal{C}}(y) = \sup_{x \in \mathcal{C}} \langle y, x \rangle$ is the *support function* of $\mathcal{C}$. The primal-dual gap at $(x_k, \nabla f(x_k))$ is then

$$\begin{aligned} f(x_k) - \Big(- f^*(\nabla f(x_k)) - \sigma_{\mathcal{C}}(-\nabla f(x_k)) \Big) &= \langle \nabla f(x_k), x_k \rangle + \sigma_{\mathcal{C}}(-\nabla f(x_k)) \\ &= \langle \nabla f(x_k), x_k \rangle + \sup_x \langle -\nabla f(x_k), x \rangle \\ &= \langle \nabla f(x_k), x_k \rangle - \langle \nabla f(x_k), s_k \rangle \\ &= g_k. \end{aligned}$$

where the first equality is the Fenchel-Young identity [Roc15], and the rest follows from the definition of $\sigma_{\mathcal{C}}$ and of s_k. Therefore, since g_k can be easily computed at each iteration, it can be used as a stopping criterion. In fact, just like the suboptimality $f(x_k) - f(x^*)$, the minimum Frank-Wolfe gap up to iteration k can be shown to converge to zero at a rate $O(1/k)$ [Jag13, PNAJ20].

1.3.3 The strongly convex case

We have seen that the Frank-Wolfe algorithm reaches a sublinear rate of $O(1/\epsilon)$ for convex, smooth objective functions. Comparing with the rates derived in Section 1.2, it is natural to wonder whether the Frank-Wolfe algorithm achieves a linear rate of convergence (namely, a rate of the form $O(\log \frac{1}{\epsilon})$) when the objective function is strongly convex. This answer is negative, as [CC68] provide an example of a positive definite quadratic over a polytope for which Frank-Wolfe converges at rate $\Omega(\frac{1}{t^{1+\mu}})$ for $\mu > 0$ arbitrarily small. In other words, the rate derived in Theorem 1.3.1 is optimal (up to constants) even when the objective function is strongly convex.

Under some additional structural assumption, the rate of Frank-Wolfe can however be improved. We review two classical results here.

Optimum in the interior. When f is L-smooth and μ-strongly convex and x^* is in the interior of the feasible set, [GM86] prove that Frank-Wolfe enjoys a linear rate of convergence which depends on both L and μ, and thus is not affine invariant. [LJJ13] fix this by defining an affine invariant notion of strong convexity, from which they derive an affine invariant linear convergence rate. In particular, the rate depends on the distance between the optimizer and the boundary of the feasible set, and in that sense the Frank-Wolfe is adaptive to this quantity. However this also means that the rate can become arbitrarily bad if that distance is small.

Strongly convex sets. To obtain linear rates of convergence, an additional quadratic structure can be placed on the set instead of the objective function. This is the motivation behind the notion of strongly convex set.

Definition 1.3.3. *A set $\mathcal{C} \subset \mathbb{R}^d$ is called σ-strongly convex w.r.t. $\|\cdot\|$ if*

$$\forall x, y \in \mathcal{C}, \forall \lambda \in [0,1],\ B_{\frac{\sigma}{2}\lambda(1-\lambda)\|x-y\|^2}(\lambda x + (1-\lambda)y) \subset \mathcal{C},$$

where $B_\delta(\tilde{x})$ denotes the ball of radius δ centered at $\tilde{x}$.

Examples of strongly convex sets include ℓ_p-balls and ℓ_p-Schatten balls for $p \in]1,2]$. Under strong convexity of the feasible set, the Frank-Wolfe algorithm enjoys a linear convergence rate when $\|\nabla f(x)\| > \delta$ for all $x \in \mathcal{C}$ [LP66, DR70, Dun79]. In particular, it holds that

$$f(x_k) - f(x^*) \leq (1-\rho)^k (f(x_0) - f(x^*)),$$

for $\rho = \min\left(\frac{1}{2}, \frac{\sigma\delta}{8L}\right)$. This rate is not affine invariant, as neither L, σ, nor δ are affine invariant properties of the problem. The notion of strongly convex set can be generalized to the one of uniformly convex set which, for example, include all ℓ_p-balls for $p > 1$. This results in an interesting interpolation between linear rates for strongly convex sets and improved sublinear rates for general uniformly convex sets [KdP21].

The assumption on the lower bound on the gradient is quite strong, and in particular does not hold when the unconstrained optimizer is in the feasible set. Removing this assumption, the Frank-Wolfe algorithm still exhibits a $O(1/k^2)$ convergence rate when both the feasible set and the objective function are strongly convex [GH15].

1.4 Variants of Frank-Wolfe

In the last section, we saw that under additional assumptions, Frank-Wolfe can achieve rates faster than $O(1/k)$. Those assumptions are however quite restrictive, and in particular, do not deal with the case when the unconstrained minimizer is on the boundary of the feasible set. In this section, we study variants of Frank-Wolfe achieving fast rates under weaker assumptions. By variants of Frank-Wolfe, we mean an iterative method which, at each iteration, computes one linear minimization over the feasible set, and for which the next iterate is a convex combination of the previous iterates. Those methods are often referred to as *conditional gradient methods*, or *Frank-Wolfe variants*. We describe this general framework in Algorithm 4. Note that the linear function definition, namely the choice of p_k, as well as the way x_k is computed, is omitted, and left for the actual algorithm definition. In particular, the Frank-Wolfe method (Algorithm 3) fits in the framework with $p_k = \nabla f(x_{k-1})$ and $x_k = (1-\eta_k)x_{k-1} + \eta_k s_k$.

The first result we give is a negative one. It states that in the smooth convex case, conditional gradient methods cannot achieve rates better than $O(1/\epsilon)$. This removes the hope for accelerated rates of the form $O(1/\sqrt{\epsilon})$, which were achievable when using projections (see Section 1.2.3).

Algorithm 4 General conditional gradient method

Input: $x_0 \in \mathcal{C}$
for $k = 1, 2, \ldots$ **do**
  Choose some $p_k \in \mathbb{R}^d$
  $s_k = \text{LMO}(p_k)$
  Output $x_k \in \text{conv}(x_0, s_1, \ldots, s_k)$
end for

Theorem 1.4.1 ([Lan13]). *Let $\epsilon > 0$. For any $L > 0$, there exists a convex, L-smooth function f such that the number of iterations of any methods of the form Algorithm 4 to reach $f(x_k) - f(x^*) < \epsilon$ cannot be smaller than*

$$\left\lceil \min\left\{ \frac{d}{2}, \frac{L diam(\mathcal{C}, \|\cdot\|_2)^2}{4\epsilon} \right\} \right\rceil - 1. \tag{1.16}$$

Note that a similar lower bound was derived in [Jag13]. In view of the result derived in Theorem 1.3.1, the Frank-Wolfe algorithm achieves the optimal convergence rate for a conditional gradient method when the underlying dimension is sufficiently large and/or when moderate precision is required, i.e. $d \geq O\left(\frac{L\text{diam}(\mathcal{C}, \|\cdot\|_2)^2}{\epsilon}\right)$.

Although Theorem 1.4.1 states that purely accelerated rates are out of reach in the general smooth convex case, acceleration is still possible under various assumptions. For example, when the feasible set is a polytope and some constraint qualification holds, Frank-Wolfe converges asymptotically at rate $O(1/k^2)$ [Bac21]. Accelerated rates on ℓ_p-norm balls can also be achieved by adding momentum to the Frank-Wolfe iterates [LCGL20].

We now turn our focus to the strongly convex case, for which we saw that the Frank-Wolfe algorithm cannot achieve linear convergence under classical assumptions (see Section 1.3.3). The variants we explore in this section were precisely developed to overcome the limitations of the Frank-Wolfe algorithm, and reach linear convergence when the feasible set is a polytope. For the remainder of this section we assume that $\mathcal{C} = \text{conv}(\mathcal{A})$, where $\mathcal{A} \subset \mathbb{R}^d$ is a finite set of vectors, also called *atoms*.

Before we move on to describing Frank-Wolfe variants, we emphasize that the problem achieving the lower bound of Theorem 1.4.1 is a strongly convex smooth function over a polytope. In that sense, it is important to note that although linear convergence rates of the form $O(\log \frac{1}{\epsilon})$ over polytopes can be attained on strongly convex objectives, those cannot beat the lower bound (1.16), and hence one can only hope that those rates improve on a sublinear rate $O(1/\epsilon)$ when the underlying problem satisfies $d \leq \frac{L\text{diam}(\mathcal{C}, \|\cdot\|_2)^2}{2\epsilon}$.

1.4.1 Away steps

When the optimum x^* lies on the boundary of the feasible set, the late iterates of the Frank-Wolfe algorithm exhibit a *zig-zag* phenomenon between the extreme points defining the face containing the optimum [Wol70, GM86, LJJ15]. This explains the slow rate $O(1/k)$. In order to fix this, the idea to move *away* from some atoms in the active set was first introduced in [Wol70]. At each iteration, the method computes the classical Frank-Wolfe direction, and compares it with the away direction, namely the direction $d_k^A = x_k - v_k$ which maximizes the potential descent $-\langle \nabla f(x_k), x_k - v \rangle$ among all atoms v in the active set. The algorithm takes a step in the direction yielding the best potential descent. We summarize the method in Algorithm 5.

Note that computing v_k is simply a search over the active set $\mathcal{S}_k$, and is typically much simpler than computing the LMO. In contrast with the Frank-Wolfe algorithm, the away-steps Frank-Wolfe algorithm requires keeping track of the active set $\mathcal{S}_k$ and of the coefficients of the

Algorithm 5 Away-steps Frank-Wolfe

Input: $x_0 \in \mathcal{A}$
Set $\mathcal{S}_0 = \{x_0\}$, $\alpha^0_{x_0} = 1$ and $\alpha^0_v = 0$ for $v \in \mathcal{A} \setminus \{x_0\}$.
for $k = 0, 1, \ldots$ **do**
  Compute $s_k = \mathrm{LMO}(\nabla f(x_k))$
  Set $d^{FW}_k = s_k - x_k$
  Compute $v_k \in \arg\max_{v \in \mathcal{S}_k} \langle \nabla f(x_k), v \rangle$
  Set $d^A_k = x_k - v_k$
  if $-\langle \nabla f(x_k), d^{FW}_k \rangle \geq -\langle \nabla f(x_k), d^A_k \rangle$ **then**
    Set $d_k = d^{FW}_k$ and $\eta_{\max} = 1$
  else
    Set $d_k = d^A_k$ and $\eta_{\max} = \alpha^k_{v_k} / (1 - \alpha^k_{v_k})$
  end if
  Find $\eta_k \in \arg\min_{\eta \in [0, \eta_{\max}]} f(x_k + \eta d_k)$
  $x_{k+1} = x_k + \eta_k d_k$
  if $d_k = d^{FW}_k$ **then**
    $\alpha^{k+1}_{s_k} = (1 - \eta_k)\alpha^k_{s_k} + \eta_k$ and $\alpha^{k+1}_v = (1 - \eta_k)\alpha^k_v$ for $v \in \mathcal{A} \setminus \{s_k\}$
  else
    $\alpha^{k+1}_{v_k} = (1 + \eta_k)\alpha^k_{v_k} - \eta_k$ and $\alpha^{k+1}_v = (1 + \eta_k)\alpha^k_v$ for $v \in \mathcal{A} \setminus \{v_k\}$
  end if
  $\mathcal{S}_{k+1} = \left\{ v \in \mathcal{A} \mid \alpha^{k+1}_v > 0 \right\}$
end for

convex combination representation of the current iterate. This entails a slight storage overhead, but is computationally cheap [LJJ15].

1.4.2 Pairwise steps

The away-steps Frank-Wolfe tends to produce sparser iterates than the original Frank-Wolfe algorithm. The pairwise Frank-Wolfe algorithm allows to obtain even sparser iterates. In contrast with the away steps Frank-Wolfe, the pairwise Frank-Wolfe algorithm only considers one descent direction at each iteration. It looks to swap mass between the away atom (v_k in Algorithm 5) and the Frank-Wolfe atom (s_k, the result of the LMO call). We describe the full method in Algorithm 6. It was originally introduced in [MDM74], and revisited in [LJJ15].

As we will see in Section 1.4.4, the theoretical rate of pairwise Frank-Wolfe is not competitive with other conditional gradient methods. However, in addition to producing sparser iterates in general, it also exhibits good practical performance [PNAJ20].

Algorithm 6 Pairwise Frank-Wolfe

Input: $x_0 \in \mathcal{A}$
Set $\mathcal{S}_0 = \{x_0\}$, $\alpha^0_{x_0} = 1$ and $\alpha^0_v = 0$ for $v \in \mathcal{A} \setminus \{x_0\}$.
for $k = 0, 1, \ldots$ **do**
  Compute $s_k = \mathrm{LMO}(\nabla f(x_k))$
  Compute $v_k \in \arg\max_{v \in \mathcal{S}_k} \langle \nabla f(x_k), v \rangle$
  Set $d_k = s_k - v_k$ and $\eta_{\max} = \alpha^k_{v_k}$
  Find $\eta_k \in \arg\min_{\eta \in [0, \eta_{\max}]} f(x_k + \eta d_k)$
  $x_{k+1} = x_k + \eta_k d_k$
  $\alpha^{k+1}_{s_k} = \alpha^k_{s_k} + \eta_k$, $\alpha^{k+1}_{v_k} = \alpha^k_{v_k} - \eta_k$ and $\alpha^{k+1}_v = \alpha^k_v$ for $v \in \mathcal{A} \setminus \{s_k, v_k\}$
  $\mathcal{S}_{k+1} = \left\{ v \in \mathcal{A} \mid \alpha^{k+1}_v > 0 \right\}$
end for

1.4.3 Fully corrective

As we have seen by now, the iterates of conditional gradient methods are convex combination of atoms in the active set. The fully-corrective Frank-Wolfe variant, also known as extended Frank-Wolfe [Hol74, Cla10, BRZ21], is the algorithm which simply picks the convex combination yielding the best function value. We describe it in Algorithm 7. Although it is an intuitive and simple algorithm, the step $x_k \in \arg\min_{y \in \operatorname{conv}(\mathcal{S}_k)} f(y)$ might be computationally expensive (sometimes as expensive as minimizing f over $\mathcal{C}$), which limits the use of the method in some practical applications. However, in situations where linear minimization on $\mathcal{C}$ is expensive, but the minimization over the convex hull of the active set is cheap, Algorithm 7 is an ideal candidate (we will encounter such a case in Chapter 3, see also [CP23]).

Algorithm 7 Fully-corrective Frank-Wolfe

Input: $x_0 \in \mathcal{C}$
Set $\mathcal{S}_0 = \{x_0\}$
for $k = 0, 1, \dots$ **do**
  $s_k = \operatorname{LMO}(\nabla f(x_k))$
  $\mathcal{S}_{k+1} = \mathcal{S}_k \cup \{s_k\}$
  $x_{k+1} \in \arg\min_{y \in \operatorname{conv}(\mathcal{S}_{k+1})} f(y)$
end for

1.4.4 Linear convergence on polytopes

The first linear convergence rate of a conditional gradient method for smooth, strongly convex functions on polytopes was derived in [LJJ15]. It is based on a conditioning of the feasible set known as the *pyramidal width*. Other condition numbers have been proposed, such as the *vertex-facet distance* [BS17], or the *restricted width* [PRS16]. Those turn out to be essentially equivalent, and can be unified under the notion of *facial distance* introduced in [PR19]. Those constants are notoriously hard to apprehend, let alone compute in practical settings. They however allow for linear convergence rates to be derived for conditional gradient methods. For example, [LJJ15] prove that when f is L-smooth and μ-strongly convex, it holds that

$$f(x_k) - f(x^*) \leq (1-\rho)^{\gamma(k)} \left(f(x_0) - f(x^*)\right),$$

where

$$\gamma(k) = \begin{cases} \frac{k}{2} & \text{for away-steps Frank-Wolfe (Algorithm 5),} \\ \frac{k}{3|\mathcal{A}|!+1} & \text{for pairwise Frank-Wolfe (Algorithm 6),} \\ k & \text{for fully-corrective Frank-Wolfe (Algorithm 7),} \end{cases}$$

and ρ is defined as

$$\rho = \min\left(\frac{1}{2}, \frac{\mu}{4L}\left(\frac{\operatorname{PWidth}(\mathcal{A})}{\operatorname{diam}(\mathcal{A})}\right)^2\right),$$

where $\operatorname{PWidth}(\mathcal{A})$ is the pyramidal width defined in [LJJ15]. The poor theoretical rate of the pairwise Frank-Wolfe algorithm is due to so-called *swap steps*, namely iterations when all the mass moves from v_k to s_k. In that case one cannot guarantee sufficient decrease in function value. While the number of swap steps can be lower bounded, this bound is loose, and this is reflected in the convergence rate.

Note that affine invariant versions of the above rates can be obtained by defining smoothness and strong convexity notions depending on both the objective functions and the feasible

set [LJJ15]. Finally, linear convergence rates over polytopes can also be obtained for functions of the form $f(x) = g(Ax) + \langle b, x\rangle$, where g is a smooth strongly convex function, but A might be a singular matrix [LJJ15, BS17]. Note that such functions are not necessarily strongly convex.

We also point out that linear convergence rates for Frank-Wolfe variants can be derived [DAST08, KY11, ÑFSA14] under the more restrictive *Robinson condition* [Rob82].

1.5 Further enhancements of Frank-Wolfe methods

1.5.1 Approximate LMO

In some practical applications, the computation of the LMO might be intractable or too expensive, but computing an approximate linear minimization over the feasible set might be efficient. One can then study the impact of an approximate linear minimization on the convergence of conditional gradient methods. The approximate LMO can either be *additive* or *multiplicative*.

Definition 1.5.1. *For $\delta > 0$, the additive δ-approximate LMO at $y \in \mathbb{R}^d$ outputs a vector $v \in \mathcal{C}$ such that*

$$\langle y, v\rangle \leq \delta + \min_{x\in\mathcal{C}} \langle y, x\rangle.$$

Definition 1.5.2. *For $\gamma \in [0, 1]$, the multiplicative γ-approximate LMO at $y \in \mathbb{R}^d$ outputs a vector $v \in \mathcal{C}$ such that*

$$\langle -y, v\rangle \geq \gamma \max_{x\in\mathcal{C}} \langle -y, x\rangle.$$

Rates similar to those derived so far in this section can be obtained when the LMO is approximate. In particular, the Frank-Wolfe algorithm still converges with a sublinear rate $O(1/k)$ for additive approximate LMO [DH78, Dun79, Jag13] and for multiplicative approximate LMO [LJJSP13]. A linear convergence rate of the Frank-Wolfe algorithm with multiplicative approximate LMO on strongly convex sets can also be found in [Ker20]. Finally, [PNAJ20] provides various rates of convergence for nonconvex, convex and strongly convex objectives under multiplicative approximate LMO for variants of the Frank-Wolfe algorithm.

1.5.2 Stochastic setting

In many practical settings, computing the exact gradient of f is much more expensive that accessing stochastic gradients. This gives rise to *stochastic* algorithms, which use a stochastic gradient instead of an exact one at each iteration. In particular, the stochastic version of the projected gradient descent algorithm (commonly referred to as SGD) has complexity $O(1/\epsilon^2)$ on smooth convex objectives when implemented with a decreasing stepsize strategy. This rate is matched for conditional gradient methods, as [AR21, HZW+22] develop momentum-based variants of Frank-Wolfe requiring $O(1/\epsilon^2)$ stochastic gradients and $O(1/\epsilon^2)$ calls to the LMO. The number of calls to the LMO can be further improved to $O(1/\epsilon)$ with increasing batch sizes [LZ16]. We also refer to the works [RSPS16, CSP20] for the nonconvex case, and to [GIZ17, BCC+22] for the strongly convex case.

Perhaps the most classical example of the use of stochastic gradient algorithms is the *empirical risk minimization* in supervised learning, where $f(x) = \frac{1}{n}\sum_{i=1}^n f_i(x)$ and n is very large (in the range of the thousands, or even millions). In that case, the complexity of evaluating one gradient may prohibit the use of deterministic algorithms, while stochastic methods are computationally cheap, as ∇f_{i_k} is an unbiased estimator of the gradient of f at iteration k, for $i_k \in \{1, \ldots, n\}$ drawn uniformly at random. Moreover, the finite-sum structure can be exploited to obtain faster rates than for purely stochastic methods. In particular, *variance-reduced* version of SGD have shown great theoretical and practical performance [SLRB17, JZ13, DBLJ14,

NLST17]. Conditional gradient methods can also be specifically tailored for the finite-sum setting [HL16, NDT+20]. In particular, for the smooth convex case, rates of the form $\tilde{O}(n + \sqrt{n}/\epsilon)$ in both number of gradients and LMO calls have been derived in [BDG23].

1.5.3 Nonconvex objectives

When the objective function f is not convex, [LJ16] prove a $O(1/\sqrt{k})$ convergence rate for the Frank-Wolfe algorithm under exact line search (Variant 2) or under knowledge of the curvature C_f. This is a rate similar to the one achieved with projected gradient methods [Nes03, CGT12, GL16]. When considering approximate multiplicative LMO, [PNAJ20] prove a similar $O(1/\sqrt{k})$ for adaptive Frank-Wolfe methods, i.e. methods which do not require an exact line search or knowledge of problem-dependent constants.

1.5.4 Stronger linear oracle

To overcome the lower bound of Theorem 1.4.1, [Lan13] suggest an *enhanced linear minimization oracle*, which performs the classical linear minimization with an additional constraint on the norm of the solution. That is, for a given $y \in \mathbb{R}^d$, this oracle computes

$$\arg\min_{s \in \mathcal{C}} \langle y, s \rangle \quad \text{such that} \quad \|s - x_0\| \leq R,$$

for some given $x_0 \in \mathbb{R}^d, R > 0$. Using this oracle, they design a linearly convergent method for smooth strongly convex problems. For specific choices of norms, the complexity of the oracle can be similar to the complexity of the original LMO. However, since the choice of the norm is directly linked to the condition number of the function, the resulting convergence rates might be negatively impacted (but are still linear nonetheless).

In another line of work, [GH16] propose to use the more efficient *local linear minimization oracle*, which in the case of polytopes can be implemented using a single call to the original LMO. The derived algorithm enjoys linear convergence in the smooth strongly convex case, and can be adapted to the non-strongly convex and the stochastic cases.

1.5.5 Restart schemes

Restart schemes periodically reset some of the parameters of optimization methods. These parameters can be, for example, the initial point and/or the stepsize. Such schemes are a popular and powerful way to enhance the performance of first-order methods in some specific settings, and have been successfully applied to projected gradient methods [GB14, OC15, Rd17]. In the context of conditional gradient methods, [KdP22] design a restarted variant of the away-steps Frank-Wolfe algorithm tailored for functions satisfying *Hölderian error bounds*. Such a bound quantifies the behavior of f in a local proximity of the optimizer. The resulting restarted algorithm smoothly interpolates between sublinear and linear rates depending on the parameters of the error bounds. In particular, linear rates over polytopes are achieved under assumptions weaker than strong convexity. The results are also extended to functions satisfying *Hölder smoothness*, a strict generalization of the classical smoothness assumption studied in this section.

1.5.6 Gradient sliding and lazy updates

The result stated in Theorem 1.4.1 gives a lower bound $O(1/\epsilon)$ on the number of LMO calls to solve smooth convex problems up to ϵ-accuracy, but it does not provide a lower bound on the number of gradient evaluations. While the Frank-Wolfe algorithm has complexity $O(1/\epsilon)$ in gradient evaluations, this can be improved using the so-called *sliding* technique, which is essentially the accelerated projected gradient algorithm (Algorithm 2), except that the projection operator

is solved with increased precision using a conditional gradient method. This allows to reach complexity $O(1/\sqrt{\epsilon})$ in gradient evaluations in the convex case [LZ16]. The sliding mechanism can also be applied to the composite setting [CL15] and the nonconvex setting [QLX18].

The number of calls to the LMO can be reduced by using the so-called *lazifying* approach, where the LMO is replaced by a *weak separation oracle* [BPZ19, LPZZ17]. Instead of performing a classic linear minimization, this oracle instead approximately solves a separation problem on the feasible set. In the worst case, the result of this new oracle can be obtained by calling the classical LMO, but often times one can reuse outputs from previous iterations which still satisfy the separation requirements. Even when that is not the case and a linear minimization step is required, in many applications, an approximate solution to the LMO is sufficient to meet the separation requirement. One can thus save on computing time by early stopping the linear minimization steps. The resulting algorithms come with similar theoretical guarantees as classical Frank-Wolfe methods and have shown great performance in practical applications compared to their non-lazy counterparts.

1.5.7 Generalized conditional gradient method

Problem (1.1) can be rewritten as

$$\min_{x\in\mathbb{R}^d} \; f(x) + I_{\mathcal{C}}(x), \tag{1.17}$$

where $I_{\mathcal{C}}$ is the *indicator function* of the feasible set $\mathcal{C}$, defined as

$$I_{\mathcal{C}}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C}, \\ \infty & \text{if } x \notin \mathcal{C}. \end{cases}$$

A direct generalization of (1.17) is then

$$\min_{x\in\mathbb{R}^d} \; f(x) + g(x), \tag{1.18}$$

where g is a proper, closed, convex function with compact domain. The Frank-Wolfe algorithm (Algorithm 3) can be extended to problem (1.18) by replacing the classical LMO call by its generalized counterpart

$$s_k \in \arg\min_{x\in\mathbb{R}^d} \langle \nabla f(x_k), x\rangle + g(x). \tag{1.19}$$

The resulting algorithm is referred to as the *generalized conditional gradient method* [BLM09], and enjoys similar convergence rates as those derived in Section 1.3 [Bac15, Bec17]. There are close ties between this method and other popular first-order algorithm such has *iterative shrinkage* [BLM09] and *mirror descent* [Bac15]. The applicability of generalized conditional gradient methods depends on the complexity of solving (1.19). For example, it has been successfully used for inducing sparsity on image deblurring tasks [BL08].

1.6 Applications of Frank-Wolfe

As we have seen in this section, conditional gradient methods are projection-free, first-order optimization methods. In addition, they are affine invariant and do not depend on the choice of the underlying norm. Their iterates are sparse convex combinations of extreme points of the feasible set, and solution quality can be easily established at each iteration. All those properties have led to a wide use of those methods in a variety of applications. For example, conditional gradient methods have been successfully applied to submodular optimization problems [Fuj80, CJK14, B$^+$13, MHK20], training of support vector machines [LJJSP13, ÑFSA14, OAL$^+$16], computing

approximate convex representations (i.e. approximate Carathéodory) [CP23], matrix completion [SSGS11, HDP+12, AZHHL17, FGM17, GSK18], coresets [KY05, Cla10, KY11, RCS19], neural networks [SCHU17, SRD18, XSZ+20, PSZ20], computer vision [JTFF14, BLB+14, BLG+15, SBLL16, PSLS17, MAB+17, SRD18, Naz23], optimal transport [FCTR16, VCTF19, PC19, LSPC19], collision detection [MLP+22] and adversarial attacks [CZYG20, TR22].

We conclude this section by illustrating the applicability of conditional gradient methods on rank-constrained matrix problems which, given a rank constraint m, read

$$\begin{aligned} \underset{X\in\mathbb{R}^{p\times q}}{\text{minimize}} \quad & f(X) \\ \text{such that} \quad & \text{rank}(X) \leq m, \end{aligned} \tag{1.20}$$

where $f : \mathbb{R}^{p\times q} \to \mathbb{R}$ is convex. The most classical example of problem (1.20) is low-rank matrix completion, where $f(X)$ computes the distance (not necessarily Euclidean) between X and some target matrix Y.

The rank is a highly non-convex function of X, and thus the feasible set of problem (1.20) is non-convex. In that case, it is common to replace the rank of X by its trace norm (also called nuclear norm), which can be seen as the best convex approximation of the rank under appropriate assumptions [Faz02]. Recall that the trace norm of a matrix, which we denote $\|\cdot\|_{tr}$, is the sum of its singular values. The resulting optimization problem reads

$$\begin{aligned} \underset{X\in\mathbb{R}^{p\times q}}{\text{minimize}} \quad & f(X) \\ \text{such that} \quad & \|X\|_{tr} \leq \delta. \end{aligned} \tag{1.21}$$

Let $\mathcal{C}_\delta = \{X \in \mathbb{R}^{p\times q} \mid \|X\|_{tr} \leq \delta\}$. One then has (see [CP21])

$$\mathcal{C}_\delta = \text{conv}\left\{uv^\top \mid u \in \mathbb{R}^p, v \in \mathbb{R}^q,\ \|u\|_2 = \|v\|_2 = \sqrt{\delta}\right\}$$

Projecting a given matrix $Y \in \mathbb{R}^{p\times q}$ onto $\mathcal{C}_\delta$ requires computing the full singular value decomposition (SVD) of Y and performing an additional projection on the simplex. This typically results in complexity $O(pq\min(p,q))$ [GVL13]. On the other hand, given $Y \in \mathbb{R}^{p\times q}$, linear minimization on $\mathcal{C}_\delta$ is performed as

$$\begin{aligned} \underset{X\in\mathcal{C}_\delta}{\arg\min}\ \langle X, Y\rangle &= \underset{\lambda_i\geq 0,\ \sum_i \lambda_i = 1,\ \|u_i\|_2=\|v_i\|_2=\sqrt{\delta}}{\arg\min} \left\langle \sum_i \lambda_i u_i v_i^\top, Y \right\rangle \\ &= \underset{\|u\|_2=\|v\|_2=\sqrt{\delta}}{\arg\min} \langle uv^\top, Y\rangle \\ &= \underset{\|u\|_2=\|v\|_2=\sqrt{\delta}}{\arg\max} \text{Trace}\left(v^\top(-Y)u\right) \\ &= uv^\top, \end{aligned}$$

where u and v are the top left and right singular vectors of $-Y$. In other words, while the projection on $\mathcal{C}_\delta$ requires computing the full SVD of Y, the linear minimization only requires the top singular vectors. Those can be approximated via iterative methods whose complexity is $O\left(\gamma \ln(p+q)\sqrt{\sigma_1}/\sqrt{\epsilon_1}\right)$, where γ is the number of non-zero elements in Y, σ_1 is the top singular value of $-Y$, and ϵ_1 is the desired precision [KW92]. In many cases one has $\gamma << pq$, giving further theoretical motivation for the use of conditional gradient methods over projected gradient methods. The complexity bounds for projections and linear minimizations were derived in [CP21] and we also refer the reader to this work for computational examples where linear minimizations over $\mathcal{C}_\delta$ are orders of magnitude faster than projections. Finally, in the context of conditional gradient methods, the matrix Y typically has a low-rank known structure (in early iterations), which can be used to further speed up iterative methods relying on matrix-vector multiplications with Y to estimate the top singular vectors.

1.7 Thesis outline and contributions

We now outline the different contributions of this thesis.

Chapter 2: In order to achieve ϵ-precision on convex smooth optimization problems, we saw that the Frank-Wolfe method requires $O(1/\epsilon)$ iterations, while accelerated projected gradient methods require only $O(1/\sqrt{\epsilon})$ iterations. In this chapter, we prove that under parallelization, one can achieve accelerated rates using a variant of the Frank-Wolfe algorithm. More precisely, we show that this variant requires $\tilde{O}(1/\sqrt{\epsilon})$ iterations to reach an ϵ primal-dual gap, when $O(1/\sqrt{\epsilon})$ computing units are available in parallel to compute linear minimizations on the feasible set. This variant is based on a Bregman-type algorithm with accelerated convergence on stochastic composite minimization problems, and we deem this algorithm to be of interest independently of its application in the Frank-Wolfe setting. This chapter is based on the paper [DTBBT22]:

- Benjamin Dubois-Taine, Francis Bach, Quentin Berthet and Adrien Taylor. 'Fast Stochastic Composite Minimization and an Accelerated Frank-Wolfe Algorithm under Parallelization'. *Neural Information Processing Systems (NeurIPS)*, 2022.

Chapter 3: As previously seen, an important feature of conditional gradient methods is that their output is a convex combination of extreme points of the feasible set. Indeed, each linear minimization step can be assumed to yield an extreme point. In this chapter, we leverage this property to solve nonconvex separable optimization with affine constraints. For those problems, the Shapley-Folkman theorem provides a bound on the duality which depends on the nonconvexity of the component functions and on the underlying dimension of the affine constraints, but does not provide a way to obtain a solution achieving such bound. We provide a two-stage algorithm to build such a solution. The first stage approximates the optimal dual value by running a Frank-Wolfe algorithm on the epigraph of the convex relaxation of the original problem. Crucially here, the linear minimization oracle is as tractable as the Fenchel conjugate of the component functions. The second stage consists in trimming the obtained convex combination via algorithms computing Carathéodory representations. For example, approximate Carathéodory representations can be obtained using, once again, conditional gradient methods. Alternatively, we provide an algorithm computing exact conic Carathéodory representations with probability 1. Overall, the reconstructed solution achieves the duality gap bound obtained via Shapley-Folkman up to an arbitrarily small additive term. This chapter is based on the paper [DTd24]:

- Benjamin Dubois-Taine and Alexandre d'Aspremont. 'Frank-Wolfe meets Shapley-Folkman: a systematic approach for solving nonconvex separable problems with linear constraints'. *arXiv preprint arXiv:2406.18282*, 2024 (Submitted).

Chapter 4: Finally, the last chapter of this thesis is not directly linked with conditional gradient methods. Instead, we focus on the ℓ_1-norm phase unwrapping problem, which aims at recovering an image from its observation modulo 2π. In optimization terms, this is simple sparsity-inducing problem, albeit in very large dimension. We suggest using the iteratively reweighted least squares approach, which replaces the nonsmooth ℓ_1-norm objective by a sequence of weighted least squares subproblems. We solve those subproblems using a carefully designed preconditoned conjugate gradient method. We show that the resulting method enjoys a sublinear convergence rate in terms of function values. Moreover, the method is simple to implement and easily portable to GPUs, and we show through extensive experiments that the resulting implementation outperforms state of the art phase unwrapping methods. This chapter is based on the paper [DTAd24]:

- Benjamin Dubois-Taine, Roland Akiki and Alexandre d'Aspremont. 'Iteratively Reweighted Least Squares for Phase Unwrapping'. *arXiv preprint arXiv:2401.09961*, 2024 (Submitted).

Chapter

2 Acceleration of Frank-Wolfe under Parallelization

Chapter content

2.1 Introduction 20
2.1.1 Related Work 21
2.2 Notation and definitions 22
2.3 Fast Stochastic Composite Minimization 23
2.4 Accelerating Frank-Wolfe with parallelization 25
2.5 Experiments 28
Appendix 2.A Proofs for Stochastic Composite Minimization 31
2.A.1 Proof of Proposition 2.3.1 31
2.A.2 Proof of Theorem 2.3.1 35
Appendix 2.B Proofs for Accelerated Frank-Wolfe 36
2.B.1 Proof of Theorem 2.4.1 36
2.B.2 Proof of Theorem 2.4.2 39
2.B.3 Dependence on the norms 39
Appendix 2.C Experimental Details 42

2.1 Introduction

We consider the composite minimization problem

$$\min_{y \in \mathbf{V}} \{F(y) := G(y) + H(y)\}, \tag{2.1}$$

where $\mathbf{V}$ is some real vector space equipped with a norm $\|\cdot\|$ and an inner product $\langle \cdot, \cdot \rangle$, see Section 2.2 for details. For instance, one can consider $\mathbf{V} = \mathbb{R}^d$ or $\mathbf{V} = \mathbb{R}^{p \times q}$ equipped with the standard Euclidean inner product and norm. The function G is convex and β-smooth and the function H is μ-strongly convex, both with respect to (w.r.t.) some norm $\|.\|$ on $\mathbf{V}$ (see Section 2.2 for precise definitions). We assume that H is "simple", meaning that one can efficiently solve

$$\arg\min_{y \in \mathbf{V}} \{\langle z, y \rangle + H(y) + \alpha D_w(y, z_0)\}, \tag{2.2}$$

where D_w is the Bregman divergence induced by a strongly convex function w. For instance, when w is the standard squared Euclidean norm, this amounts to computing the proximal operator of H. When deterministic gradients of G are available, an accelerated method relying on (2.2) and achieving rates of the form $O\left(\exp\left(-k\sqrt{\frac{\mu}{\beta}}\right)\right)$ was proposed by [DG21, AGD+].

In this chapter, we are interested in the case where G is only available through a stochastic oracle. In particular, we provide an accelerated algorithm converging in function values to a neighborhood of the minimum with the same rate as above. The size of the neighborhood is of order $O\left(\frac{\sigma^2}{\sqrt{\mu\beta}}\right)$ where σ^2 is the variance of the stochastic gradients. The dependence of the noise term on $1/\sqrt{\mu\beta}$ is similar to that of previous stochastic accelerated methods in simpler settings, see, e.g., [AFGO19]. Although our rate is an extension of the result in [DG21], the parameters are different and tailored for the stochastic setup.

In Section 2.4, we focus on minimization problems over a compact convex set $\mathcal{C}$ for which we have access to a linear optimization oracle, just as in the Frank-Wolfe algorithm. Formally, we consider solving $\min_{x\in\mathcal{C}} f(x)$ where f is convex with Lipschitz-continuous gradients. In that case, the dual problem can be seen as a particular instance of (2.1) where $H := f^*$ is the Fenchel conjugate of f, and G is a smoothed version of the support function of $\mathcal{C}$. Such smoothed functions can only be accessed through a stochastic oracle whose computation boils down to solving a linear optimization problem on $\mathcal{C}$. An appropriate choice of w allows us to use the algorithmic framework by only computing gradients of f and solving linear optimization subproblems over $\mathcal{C}$. In short, minimizing f over the set $\mathcal{C}$ with an ϵ primal-dual gap requires $\tilde{O}\left(\max\left\{\frac{1}{\sqrt{\epsilon}}, \frac{1}{\epsilon m}\right\}\right)$ iterations when m computing units can be used in parallel (each of them only solving linear optimization subproblems on $\mathcal{C}$). In particular, when $m = 1/\sqrt{\epsilon}$, this effectively yields an accelerated rate of $\tilde{O}\left(1/\sqrt{\epsilon}\right)$ iterations with a single gradient evaluation per iteration to achieve the required accuracy.

We emphasize that the first contribution of this chapter is to provide an accelerated method for general stochastic composite problems (in Section 2.3). In Section 2.4, we then show that it can be used for obtaining an accelerated Frank-Wolfe algorithm.

2.1.1 Related Work

As previously emphasized, the main algorithmic ingredients which inspired this work were developped by [DG21] and [GN18]. For further references on acceleration and gradient-based composite convex minimization, we refer to the original works [Nes83, Nes13] or to a recent survey [dST21]. For the stochastic setup, we refer to [Lan12]. We refer to [Nes03, Dev13] for further references and pointers on those topics.

Let us briefly describe the main differences between this work and [DG21, GN18]. First, in [GN18], the authors obtain accelerated rates similar to ours when the strong convexity assumption is placed on G instead of H here (which has practical consequences, including well-posedness of (2.2) and the size of the constants β and μ, see discussion in [DG21, Introduction]). When the gradients of G are stochastic, [GN18] provides convergence rates, but only when the underlying norm is the Euclidean norm. Finally, whereas in [DG21] the authors consider more general assumptions in the deterministic non-Euclidean case, our work also covers stochastic approximations. We emphasize that each assumption we make (namely accelerated, stochastic, proximal Bregman methods) is necessary to yield acceleration of Frank-Wolfe under parallelization in Section 2.4.

Two key observations underlying this chapter are on the one hand the duality link between Frank-Wolfe methods and Bregman methods [Bac15], and on the other hand randomized smoothing techniques [NS17, DBW12, ALST14, ALT16, BBT+20, BBC+21] which can be naturally computed using linear optimization steps. The main question of interest was whether accelerated Bregman methods should naturally give rise to accelerated Frank-Wolfe methods on the dual.

The Frank-Wolfe (FW) method (a.k.a. conditional gradient method) and its variants were first introduced in [FW56] (see also [Jag13, Jag11] for more modern presentations). When considering optimization over a convex set $\mathcal{C}$, classical first-order methods are often naturally

embedded with a projection operator (onto $\mathcal{C}$). Depending on $\mathcal{C}$, those projections are potentially costly. An alternate approach for taking constraints into account within first-order method consists in using linear optimization oracles (a.k.a. Frank-Wolfe techniques). In many applications, such linear minimizations are much cheaper than projecting onto the feasible set, see for example [FGM17, HJN15]. Despite its wide use in practical applications, the main drawback of the Frank-Wolfe method lies in its slow convergence rate of $O(1/\epsilon)$, standing in sharp contrast with the $O(1/\sqrt{\epsilon})$ convergence of Nesterov's accelerated gradient descent [Nes83] relying on projections.

As "purely accelerated" rates of convergence are out of reach for vanilla Frank-Wolfe methods (see lower complexity bound in [Lan13]), most works on accelerating Frank-Wolfe have focused on exploiting specific additional assumptions on the problems at hand. Common such assumptions include strong convexity of the feasible set [DR70, LP66, GH15], and strong convexity of the objective function along with the assumption that the minimizer lies in the interior of the feasible set [GM86, LJJ15]. In both cases, Frank-Wolfe is known to improve on the $O(1/\epsilon)$ rate. Some efforts have also gone into finding rates matching performances of Nesterov's accelerated gradient descent [Nes83] without strong convexity. In particular, when $\mathcal{C}$ is a polytope and when a certain type of constraint qualification is satisfied, Frank-Wolfe converges asymptotically at rate $O(1/k^2)$ [Bac21]. Adding momentum to Frank-Wolfe has also been studied, and accelerated rates are attained on some ℓ_p-norm balls when the minimizer is on the boundary of the feasible set [LCGL20]. Our approach in this chapter is orthogonal to the aforementioned results, in that we do not make additional assumptions on the objective function or the feasible set, but instead show that parallelization can help reaching accelerated rates for a variant of Frank-Wolfe. Note that [LZ16] manages to reach a similar rate of $O(1/\sqrt{\epsilon})$ iterations for a variant of Frank-Wolfe, where each iteration requires one gradient evaluation and $O(1/\sqrt{\epsilon})$ calls to a linear optimization oracle. However, in contrast with our algorithm, their approach is non-parallelizable as the calls to the linear optimization oracle need to be made sequentially.

The rest of the chapter is organized as follows. In Section 2.2 we define notations and review some classical definitions from convex analysis. In Section 2.3 we provide worst-case rates for a stochastic Bregman dual-averaging-type algorithm, along with some intuitions and proof sketches. In Section 2.4 we show how a Frank-Wolfe method directly fits within this framework, thereby obtaining accelerated worst-case guarantees on the primal-dual gap. In Section 2.5, we illustrate our theoretical findings on a set of simple numerical examples.

2.2 Notation and definitions

Formally, we consider a real finite-dimensional vector space $\mathbf{V}$ and its dual space $\mathbf{V}^*$ consisting of all linear functions on $\mathbf{V}$, as well as a dual pairing denoted by $\langle \cdot, \cdot \rangle : \mathbf{V}^* \times \mathbf{V} \to \mathbb{R}$, and a norm $\|\cdot\| : \mathbf{V} \to \mathbb{R}$. We also consider the corresponding dual norm $\|\cdot\|_* : \mathbf{V}^* \to \mathbb{R}$ induced by the choice of $\|\cdot\|$ and $\langle \cdot, \cdot \rangle$ using $\|z\|_* = \sup_{\|x\| \leq 1} \langle z, x \rangle$. For instance, one can use $\mathbf{V} = \mathbf{V}^* = \mathbb{R}^d$ or $\mathbf{V} = \mathbf{V}^* = \mathbb{R}^{p \times q}$ equipped with the standard Euclidean inner product and norm. We insist on the fact that the formal notation $\mathbf{V}$ and $\mathbf{V}^*$ is used mostly for emphasizing differences between primal and dual spaces/problems below. For a closed, proper, convex function $\Psi : \mathbf{V} \to \mathbb{R}$, we denote $\partial \Psi(x)$ the set of all subgradients of Ψ at x. When $\partial \Psi(x)$ is a singleton, we denote its single element by $\nabla \Psi(x)$.

Definition 2.2.1. *$\Psi : \mathbf{V} \to \mathbb{R}$ is L-smooth w.r.t. $\|\cdot\|$ if for all $x, y \in \mathbf{V}$,*

$$\|\nabla \Psi(x) - \nabla \Psi(y)\|_* \leq L \|x - y\| . \tag{2.3}$$

The following proposition will be helpful throughout the chapter (see [Nes03] for a proof).

Proposition 2.2.1. *$\Psi : \mathbf{V} \to \mathbb{R}$ is L-smooth w.r.t. $\|\cdot\|$ if and only if for all $x, y \in \mathbf{V}$,*

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

Definition 2.2.2. *$\Psi : \mathbf{V} \to \mathbb{R}$ is μ-strongly convex w.r.t. $\|\cdot\|$ if for all $x, y \in \mathbf{V}$ and any $g_\Psi(y) \in \partial\Psi(y)$,*

$$\Psi(x) \geq \Psi(y) + \langle g_\Psi(y),\, x - y\rangle + \frac{\mu}{2}\|x - y\|^2 . \tag{2.4}$$

Definition 2.2.3. *For a closed, proper, convex function $\Psi : \mathbf{V} \to \mathbb{R}$, its conjugate function is defined as*

$$\Psi^*(z) = \sup_{x\in\mathbf{V}} \ \langle z,\, x\rangle - \Psi(x). \tag{2.5}$$

For later references, we need a few results related to smoothing the support function of a set $\mathcal{C} \subset \mathbf{V}^*$ ($\mathcal{C}$ will appear as a convex set in the dual problem to (2.1)).

Definition 2.2.4. *For a set $\mathcal{C} \subset \mathbf{V}^*$, we define $I_\mathcal{C}$ as the indicator function of $\mathcal{C}$, i.e., $I_\mathcal{C}(x) = 0$ for $x \in \mathcal{C}$ and $I_\mathcal{C}(x) = \infty$ for $x \in \mathbf{V}^* \setminus \mathcal{C}$. The support function of $\mathcal{C}$ is defined as*

$$s(y) = \sup_{x\in\mathcal{C}} \langle x,\, y\rangle. \tag{2.6}$$

If $\mathcal{C}$ is non-empty, convex and closed, the indicator and support functions of $\mathcal{C}$ are conjugates of each other, i.e., $(I_\mathcal{C})^* = s$ and $s^* = (I_\mathcal{C})^{**} = I_\mathcal{C}$ [Roc15]. The support function is convex yet not differentiable in general applications. For the purpose of our work, we consider smoothing the support function via randomization. Such stochastic smoothing has recently gained popularity in the optimization literature, see for example [NS17, DBW12, ALST14, ALT16, BBT+20]. We define the main tools and state the relevant properties behind this idea.

Definition 2.2.5. *For a set $\mathcal{C} \subset \mathbf{V}^*$, a scalar $\alpha > 0$ and a random variable Δ, we define the smoothed support function of $\mathcal{C}$ as*

$$s_\alpha(y) = \mathbb{E}_\Delta\left[s(y + \alpha\Delta)\right] = \mathbb{E}_\Delta\left[\sup_{x\in\mathcal{C}} \langle x,\, y + \alpha\Delta\rangle\right]. \tag{2.7}$$

We will use the following proposition on the smoothed support function, proved in [BBT+20].

Proposition 2.2.2. *Suppose $\mathcal{C} \subset \mathbf{V}^*$ is convex compact and let $R_\mathcal{C} = \max_{x\in\mathcal{C}} \|x\|_*$. Suppose the random variable Δ has positive differentiable density $d\pi(z) \propto \exp(-\eta(z))dz$ for some function $\eta(\cdot)$, and let $M^2 = \mathbb{E}\left[\|\nabla_z\eta(Z)\|_*^2\right]$. Then s_α is convex, $\frac{R_\mathcal{C} M}{\alpha}$-smooth w.r.t. $\|\cdot\|$ and for any $y \in \mathbf{V}$,*

$$\nabla s_\alpha(y) = \mathbb{E}_\Delta\left[\arg\max_{x\in\mathcal{C}} \langle x,\, y + \alpha\Delta\rangle\right] \quad \text{and} \quad s(y) \leq s_\alpha(y) \leq s(y) + \alpha s_1(0).$$

2.3 Fast Stochastic Composite Minimization

In this section we focus on solving the composite problem

$$\min_{y\in\mathbf{V}} \ \{F(y) := G(y) + H(y)\}, \tag{2.8}$$

where G is convex and β-smooth, and H is μ-strongly convex w.r.t. $\|\cdot\|$. Algorithm 8 summarizes our proposed algorithm, which resembles the general AGD+ algorithm from [DG21]. Notable differences include access to gradients of G through a stochastic oracle, and explicit dependencies on the smoothness and strong convexity constants for the different updates. Moreover, the presented analysis is different and specifically tailored to handle stochasticity in the gradients of G.

An actual implementation of Algorithm 8 requires the ability to efficiently solve the minimization step (2.13), which should be well-defined. This intermediate minimization subproblem is often referred to as a Bregman proximal problem, and a sufficient condition for this operation to be well-defined is to require w to be strongly convex. In the case where w is the Euclidean norm, (2.13) amounts to computing the proximal operator of H. Considering a general regularizer w has several benefits, in that the Euclidean norm might not well capture the geometry of the problem, and because particular choices of w might make (2.13) easily solvable. The latter will become particularly clear in Section 2.4.

Before we move on to the analysis, let us emphasize that the first step of Algorithm 8 consisting in picking $z_0 \in \arg\min_y w(y)$ is not restrictive. Indeed one could instead pick any $z_0 \in \mathbf{V}$ and set $\tilde{w}(y) = w(y) - \langle g_w(z_0), y\rangle$ where $g_w(z_0) \in \partial w(z_0)$. One could then run Algorithm 8 with the "shifted" $\tilde{w}$ instead of w. Doing so does not change the complexity of the minimization step (2.13), and the following analysis remains unchanged.

We are now ready to analyse Algorithm 8. For this purpose let us define

$$c_k := \sum_{i=0}^{k-1} (A_{i+1} - A_i)(G(y_i) - \langle \nabla G(y_i), y_i\rangle).$$

where the sequences $\{A_k\}_{k\in\mathbb{N}}$ and $\{y_k\}_{k\in\mathbb{N}}$ are defined in Algorithm 8. We also use the notation $\mathbb{E}_k$ for denoting the expectation at iteration k conditioned on the previous iterations (that is, $\mathbb{E}_k$ shortens $\mathbb{E}_k[\,\cdot\,] = \mathbb{E}[\,\cdot \,|\, y_k, z_k, d_k, c_k]$), while $\mathbb{E}$ denotes the total expectation. Before we can state our results, we need one more assumption on the stochastic gradients.

Assumption 2.3.1. *For any $k \in \mathbb{N}$, the stochastic gradients satisfy $\mathbb{E}_k[g_k] = \nabla G(v_k)$ and $\mathbb{E}_k \|g_k - \nabla G(v_k)\|_*^2 \leq \sigma^2$.*

The assumption on the variance of the stochastic gradients is common when studying stochastic first-order methods, and allows proving the next proposition, which relates consecutive iterations.

Proposition 2.3.1. *Suppose Assumption 2.3.1 holds and let $m_k(y) := \langle d_k, y\rangle + c_k + A_k H(y) + \beta w(y)$. At iteration k, the iterates of Algorithm 8 satisfy*

$$\mathbb{E}_k\left[A_{k+1}F(y_{k+1}) - m_{k+1}(z_{k+1})\right] \leq A_k F(y_k) - m_k(z_k) + (A_{k+1} - A_k)\frac{\sigma^2}{2\sqrt{\mu\beta}}. \tag{2.15}$$

We defer the full proof to Section 2.A, but highlight the main steps here. This way we also hope to shed light on the update of A_{k+1}. The first step is to compute an upper bound on $A_{k+1}G(y_{k+1})$ depending on $A_k G(y_k)$. The second step is similar, and computes an upper bound on $A_{k+1}H(y_{k+1})$ depending on $A_k H(y_k)$. Summing up the two inequalities and taking expectations yields inequality (2.15) with an additional term in $\|z_{k+1} - z_k\|^2$. To exactly obtain inequality (2.15) we set A_{k+1} so as to cancel out the coefficient multiplying $\|z_{k+1} - z_k\|^2$. This turns out to be equivalent to setting A_{k+1} as the root of a quadratic polynomial, explaining the form of update (2.9) in Algorithm 8.

Proposition 2.3.1 allows us to get the final rate of convergence of Algorithm 8. We again defer the proof to Section 2.A. The idea here is first to unroll the recursion in (2.15) so as to get a constant term on the right-hand side of the inequality. We then relate $m_N(z_N)$ to $m_N(y_\star)$

Algorithm 8 Stochastic Composite Minimization

Input: (β, μ, ν). β-smooth function G, μ-strongly convex function H, ν-strongly convex function w, all w.r.t. the same norm $\|\cdot\|$.
Pick $z_0 \in \arg\min_y w(y)$ and set $y_0 = z_0$.
Set $A_0 = 0$ and $d_0 = 0$.
for $k = 0, 1, \ldots,$ **do**

$$A_{k+1} = \frac{A_k(\mu+2\beta+\sqrt{\mu\beta})+\beta\nu+\sqrt{(\beta\nu+\mu A_k)^2+4A_k\beta^2\nu+5A_k^2\mu\beta+2A_k\sqrt{\mu\beta}(\beta\nu+A_k\mu)}}{2(\beta+\sqrt{\mu\beta})} \tag{2.9}$$

$$\tau_k = 1 - \frac{A_k}{A_{k+1}} \tag{2.10}$$

$$v_k = (1-\tau_k)y_k + \tau_k z_k \tag{2.11}$$

Compute a stochastic gradient g_k of function G at iterate v_k

$$d_{k+1} = d_k + (A_{k+1} - A_k)g_k \tag{2.12}$$

$$z_{k+1} \in \arg\min_{y\in\mathbf{V}} \{\langle d_{k+1}, y\rangle + A_{k+1}H(y) + \beta w(y)\} \tag{2.13}$$

$$y_{k+1} = (1-\tau_k)y_k + \tau_k z_{k+1} \tag{2.14}$$

end for

and to the minimum of F and finally, we show that $A_{k+1} \geq A_k\left(1 + \frac{\sqrt{\mu}}{2(\sqrt{\beta}+\sqrt{\mu})}\right)$, which gives the exponential decay term.

Theorem 2.3.1. *Suppose Assumption 2.3.1 holds, let $y_\star \in \arg\min F$ and define $D_w(y_\star, y_0) = w(y_\star) - w(y_0) \geq 0$. The convergence rate of Algorithm 8 after k iterations is*

$$\mathbb{E}[F(y_k) - F(y_\star)] \leq \frac{\sqrt{\beta}+\sqrt{\mu}}{\nu\sqrt{\beta}} \exp\left(-\frac{(k-1)\sqrt{\mu}}{4\left(\sqrt{\beta}+\sqrt{\mu}\right)}\right) \beta D_w(y_\star, y_0) + \frac{\sigma^2}{2\sqrt{\mu\beta}}. \tag{2.16}$$

This rate is a typical accelerated rate. In the (usual) case where $\mu \leq \beta$, the exponential decay is bounded above by $\exp\left(-\frac{(k-1)}{8}\sqrt{\frac{\mu}{\beta}}\right)$, which shows the natural dependence on $\sqrt{\frac{\mu}{\beta}}$. In addition, the neighborhood term is of the form $O\left(\frac{\sigma^2}{\sqrt{\mu\beta}}\right)$, which is again typical of accelerated stochastic methods [AFGO19]. Note that $w(y_\star) - w(y_0)$ is equal to the Bregman divergence $D_w(y_\star, y_0) := w(y_\star) - w(y_0) - \langle g_w(y_0); y_\star - y_0\rangle$ (with $g_w(y_0) \in \partial w(y_0)$) through the choice $g_w(y_0) = 0 \in \partial w(y_0)$, which is valid as y_0 minimizes $w(\cdot)$. We emphasize that we do not require differentiability of the function w anywhere.

In the next section, we show how the above algorithm can be directly applied to a smoothed dual of a minimization problem over a compact convex set, yielding a variant of the Frank-Wolfe algorithm which can achieve accelerated rates under parallelization.

2.4 Accelerating Frank-Wolfe with parallelization

We consider the following minimization problem over a compact convex set $\mathcal{C} \subset \mathbf{V}^*$

$$\min_{x\in\mathbf{V}^*} \{f(x) + I_{\mathcal{C}}(x)\}, \tag{2.17}$$

where f is convex and L-smooth w.r.t. $\|\cdot\|_*$. Its Fenchel-Rockafellar dual [Roc15] reads

$$\max_{y \in \mathbf{V}} \{d(y) := -s(-y) - f^*(y)\}. \tag{2.18}$$

The smoothness of f implies that f^* is $1/L$-strongly convex w.r.t. $\|\cdot\|$ [RW09, Proposition 12.60]. The term in s is however not smooth w.r.t. $\|\cdot\|$, which prevents us from directly applying results from the previous section. Instead, we choose some smoothing parameter $\alpha > 0$ and, following Definition 2.2.5 and Proposition 2.2.2, we consider the smoothed minimization problem

$$\min_{y \in \mathbf{V}} \{s_\alpha(-y) + f^*(y)\}. \tag{2.19}$$

Problem (2.19) fits within the framework of (2.8) with $G(y) = s_\alpha(-y)$, $H(y) = f^*(y)$, $\mu = 1/L$ and $\beta = \frac{R_{\mathcal{C}} M}{\alpha}$. Using Proposition 2.2.2, we see that an unbiased stochastic gradient g of G at some point y can be obtained by sampling Δ and computing $g = -\arg\max_{u \in \mathcal{C}} \langle u, -y + \alpha\Delta \rangle$. This boils down to a linear optimization oracle over $\mathcal{C}$, the same oracle as that of the Frank-Wolfe algorithm.

We now show how to pick the distance generating function w such that the minimization step of Algorithm 8 has a closed form. Recalling that z_0 must minimize w, we set $w(y) = f^*(y) - \langle g_{f^*}(z_0), y \rangle$ where $g_{f^*}(z_0) \in \partial f^*(z_0)$. Clearly, w is $1/L$-strongly convex w.r.t. $\|\cdot\|$ and is minimized at z_0. Moreover, first-order optimality conditions for the minimization step (2.13) of Algorithm 8 read

$$0 \in d_{k+1} + A_{k+1} \partial f^*(z_{k+1}) + \frac{R_{\mathcal{C}} M}{\alpha} \left(\partial f^*(z_{k+1}) - g_{f^*}(z_0) \right) \tag{2.20}$$

$$\iff \partial f^*(z_{k+1}) \ni \frac{\frac{R_{\mathcal{C}} M}{\alpha}}{A_{k+1} + \frac{R_{\mathcal{C}} M}{\alpha}} g_{f^*}(z_0) - \frac{d_{k+1}}{A_{k+1} + \frac{R_{\mathcal{C}} M}{\alpha}}. \tag{2.21}$$

Choosing some $x_0 \in \mathcal{C}$ and setting $z_k = \nabla f(x_k)$ for all k, one can replace (2.21) by

$$x_{k+1} = \frac{\frac{R_{\mathcal{C}} M}{\alpha}}{A_{k+1} + \frac{R_{\mathcal{C}} M}{\alpha}} x_0 - \frac{d_{k+1}}{A_{k+1} + \frac{R_{\mathcal{C}} M}{\alpha}}. \tag{2.22}$$

Doing this allows to avoid computing subgradients of f^*. Instead, whenever the value of z_k is needed, we compute $\nabla f(x_k)$. We summarize this fully primal algorithm in Algorithm 9. Note that similar tricks to obtain "dual-free" methods have already been used, see for example [Lan12, WX17, HBM20, CST21, JST22].

Observe that each iteration of Algorithm 9 requires the computation of one gradient of f, and m calls to a linear maximization oracle. When $m = 1$, an iteration of Algorithm 9 is as costly as one iteration of Frank-Wolfe. We now show how to further exploit parallelization. To do so, we need to ensure that using multiple samples appropriately improves the quality of the approximation of the true gradient. This the point of the following assumption.

Assumption 2.4.1. *There exists a norm-dependent constant $\rho_{\|\cdot\|_*}$ such that the variance verifies*

$$\sigma^2 = \mathbb{E}_k \left\| \frac{1}{m} \sum_{i=1}^{m} g_{k,i} - \nabla G(v_k) \right\|_*^2 \leq \frac{4 R_{\mathcal{C}}^2 \rho_{\|\cdot\|_*}}{m}. \tag{2.23}$$

Remark 2.4.1. *A few examples are in order. For the standard Euclidean norm $\|\cdot\| = \|\cdot\|_* = \|\cdot\|_2$ it holds that $\rho_{\|\cdot\|_2} = 1$. For ℓ_p-norms on a space of dimension d, we have*

$$\rho_{\|\cdot\|_p} = \begin{cases} d^{2/p-1} & \text{for } 1 \leq p < 2 \\ p - 1 & \text{for } 2 \leq p < \infty \\ e^2(\log d + 1) & \text{for } p = \infty. \end{cases} \tag{2.24}$$

Algorithm 9 Parallel Frank-Wolfe (PFW)

Input: $(L, x_0, R_{\mathcal{C}}, M, m, \alpha, T)$. L-smooth convex function f w.r.t. $\|\cdot\|_*$, $x_0 \in \mathcal{C}$, $R_{\mathcal{C}} = \max_{x\in\mathcal{C}} \|x\|_*$, distribution with density $d\pi(z) \propto \exp(-\eta(z))dz$ such that $M^2 = \mathbb{E}_\Delta \|\nabla\eta(\Delta)\|_*^2$, m computing units in parallel, smoothing parameter α, number of iterations T.
Set $A_0 = 0$, $d_0 = 0$, $\beta = \frac{R_{\mathcal{C}} M}{\alpha}$, $\mu = \nu = \frac{1}{L}$, $y_0 = \nabla f(x_0)$.
for $k = 0, 1, \ldots, T-1$ **do**

$$
\begin{aligned}
A_{k+1} &= \frac{A_k(\mu+2\beta+\sqrt{\mu\beta})+\beta\mu+\sqrt{\mu^2(\beta+A_k)^2+4A_k\beta^2\mu+5A_k^2\mu\beta+2A_k\mu\sqrt{\mu\beta}(\beta+A_k)}}{2(\beta+\sqrt{\mu\beta})} \\
\tau_k &= 1 - \frac{A_k}{A_{k+1}} \\
v_k &= (1-\tau_k)y_k + \tau_k \nabla f(x_k)
\end{aligned}
$$

For all $i \in [m]$ in parallel, sample $\Delta_i \sim d\pi$, compute $g_{k,i} = -\arg\max_{u\in\mathcal{C}} \langle u, -v_k + \alpha\Delta_i\rangle$

$$
\begin{aligned}
g_k &= \frac{1}{m}\sum_{i=1}^m g_{k,i} \\
d_{k+1} &= d_k + (A_{k+1} - A_k) g_k \\
x_{k+1} &= \frac{\beta}{A_{k+1}+\beta} x_0 - \frac{1}{A_{k+1}+\beta} d_{k+1} \\
y_{k+1} &= (1-\tau_k) y_k + \tau_k \nabla f(x_{k+1})
\end{aligned}
$$

end for

In the general case, as all norms are equivalent in finite dimensional spaces, there exist constants $c, C > 0$ such that $c\|\cdot\|_2 \le \|\cdot\|_* \le C\|\cdot\|_2$. Then $\rho_{\|\cdot\|_*} \le \frac{C^2}{c^2}$. We refer to Appendix 2.B.3 for more details and references.

As mentioned earlier, one can directly obtain convergence of the dual problem (2.19) by applying Theorem 2.3.1. The next theorem states that not only are the iterates $\{x_k\}_{k\in\mathbb{N}}$ always feasible, but also that the previous rate of convergence also holds for the primal-dual gap between the iterates $\{x_k\}_{k\in\mathbb{N}}$ and $\{y_k\}_{k\in\mathbb{N}}$. For simplicity of the exposition we assume that $\frac{R_{\mathcal{C}} M}{\alpha} \ge \frac{1}{L}$. In practice this is not an issue since we will choose small values of α.

Theorem 2.4.1. *Suppose f is convex and L-smooth w.r.t. $\|\cdot\|_*$ and let $x_0 \in \mathcal{C}$. Under Algorithm 9, the primal iterates are always feasible, i.e., $x_k \in \mathcal{C}$ for all $k \in \mathbb{N}$. Moreover, after k iterations, we have*

$$
\mathbb{E}\left[f(x_k) - d(y_k)\right] \le \exp\left(-(k-1)\frac{\sqrt{\alpha}}{8\sqrt{LR_{\mathcal{C}}M}}\right)\frac{2LR_{\mathcal{C}}M}{\alpha}\left(f(x_0) - f(x_\star)\right) + \frac{2R_{\mathcal{C}}^2\rho_{\|\cdot\|_*}}{m}\sqrt{\frac{\alpha L}{R_{\mathcal{C}}M}} + \alpha s_1(0).
$$

The above convergence result is obtained by first plugging the smoothness and strong convexity constants of this section within the result of Proposition 2.3.1. Then, one has to relate $m_k(z_k)$ to the primal objective $f(x_k)$. The first term in the upper bound then follows by recursion. The second term comes from the variance, which can be bounded by $\frac{4R_{\mathcal{C}}^2\rho_{\|\cdot\|}}{m}$ when m stochastic gradients are computed in parallel. The last term $\alpha s_1(0)$ is due to the error induced by the smoothing of the dual. The full proof can be found in Section 2.B. From Theorem 2.4.1 we are able to show that Algorithm 9 achieves acceleration under parallelization.

Theorem 2.4.2. *Suppose f is convex and L-smooth w.r.t. $\|\cdot\|_*$ and let $x_0 \in \mathcal{C}$. Under Algorithm 9 with $\alpha = \min\left\{\frac{\epsilon}{3s_1(0)}, \frac{M\epsilon^2 m^2}{36LR_{\mathcal{C}}^3\rho_{\|\cdot\|_*}^2}\right\}$, the number of iterations required to achieve an ϵ*

primal-dual gap, i.e., $\mathbb{E}[f(x_k) - d(y_k)] \leq \epsilon$*, is*

$$k \geq 1 + \frac{8\sqrt{LR_{\mathcal{C}}M}}{\sqrt{\alpha}} \log\left(\frac{6LR_{\mathcal{C}}M(f(x_0) - f(x_\star))}{\epsilon\alpha}\right). \tag{2.25}$$

The complexity is therefore $\tilde{O}\left(\max\left(\frac{1}{\sqrt{\epsilon}}, \frac{1}{\epsilon m}\right)\right)$.

The above complexity result shows that (i) when $m = 1/\sqrt{\epsilon}$, we get an accelerated rate $\tilde{O}(1/\sqrt{\epsilon})$, (ii) when $m = 1$, we recover the classical $\tilde{O}(1/\epsilon)$ complexity of Frank-Wolfe (up to logarithmic terms), (iii) there is no theoretical gain in going beyond $m = 1/\sqrt{\epsilon}$ computing units in parallel.

Moreover, while the total number of calls to the linear optimization oracle to reach an ϵ primal-dual gap is the same as in Frank-Wolfe for any value of m, in the case where $m > 1$, the number of required gradients of f is strictly less than in Frank-Wolfe. This is because in each iteration of Algorithm 9 we compute one gradient of f no matter what the value of m is. In particular, when $m = 1/\sqrt{\epsilon}$, we need only $\tilde{O}(1/\sqrt{\epsilon})$ gradients of f in total, compared to $O(1/\epsilon)$ in Frank-Wolfe.

One can note that the proposed algorithm in not as universal as the classical Frank-Wolfe method, as it requires upper bounds on several problem-specific constants. However, acceleration of Frank-Wolfe has been extensively studied in the literature (see Section 2.1) and to the best of our knowledge our work is the first to provide accelerated rates (under parallelization) without further assumptions on the constraint set and/or the objective function. We argue next that estimating upper bounds on most parameters does not pose significant challenges. Indeed, as the user is free to pick any distribution to sample Δ from (up to the assumptions of Proposition 2.2.2), one can choose a distribution for which the constant M is easy to compute (we give two examples in Section 2.5). Moreover, the user typically knows the set over which the optimization is carried over, and from such knowledge an upper bound on the diameter can often be easily computed. Finally, we also need to upper bound the Lipschitz constant L of the gradient. This seems to be more of a limiting factor compared to classical Frank-Wolfe, although an upper bound on L is often (but not always) necessary to implement other accelerated versions of Frank-Wolfe [GH15, LZ16] as well. While it might be possible to circumvent this problem using some sort of line search within our method, it is not straightforward as the smoothing of the dual induces stochasticity, and obtaining theoretical rates for line search techniques on stochastic objectives is notoriously tedious [BCN18, VML+19, VDTB22].

Finally, note that it might seem like a small modification to Algorithm 9 could give a variant of Frank-Wolfe capable of handling stochastic gradients of f (for previous works on stochastic Frank-Wolfe, see [HL16, Gha19, NDT+20] and references therein). Indeed, one could see the noise in $\nabla f(x_k)$ in the update of v_k as the random variable Δ itself and study the resulting algorithm. However, the current analysis would only work if the noise on $\nabla f(x_k)$ does not depend on the current iterate. Moreover, even under this simplifying assumption, the result does not follow immediately as the stochasticity would appear in both the updates of v_k and y_{k+1}, in contrast with the current analysis.

2.5 Experiments

We first consider minimizing a quadratic function in $\mathbb{R}^d$ over the simplex

$$\min_{x \in \mathcal{C}_1} \{f_1(x) := \frac{1}{2}\|Ax - b\|_2^2\} \quad \text{where} \quad \mathcal{C}_1 = \left\{x \in \mathbb{R}^d \mid x \geq 0, \sum_{i=1}^d x_i = 1\right\}, \tag{2.26}$$

where $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$. We choose Δ to have the Gumbel distribution [Gum54] with location and scale parameters equal to 0 and 1 respectively. In this case $R_{\mathcal{C}_1} = 1$ and $M = \sqrt{d}$

(see Section 2.C for details). We set $n = 200$, $d = 50$ and compare the bound from Theorem 2.4.1 with the practical performance of Algorithm 9 for both $m = 1$ and $m = 1/\sqrt{\alpha}$ parallel computer(s). Figure 2.1 shows that our upper bound captures well the speed of convergence to a neighborhood.

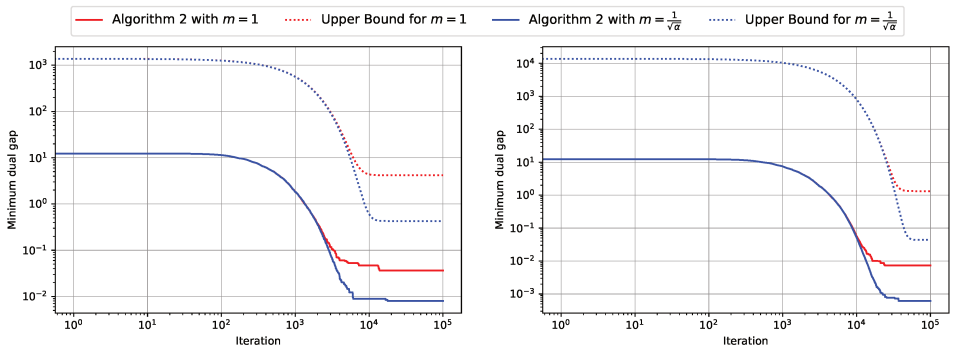

Figure 2.1: Comparisons between the behavior of Algorithm 9 and that of its theoretical upper bound (see Theorem 2.4.1) on a least-squares problem on the simplex for $\alpha = 10^{-2}$ (left) and $\alpha = 10^{-3}$ (right). The plots report the value of the best primal-dual gap incurred at the current iteration.

Next, in order to circumvent the inevitable stalling due to the fixed value of α observed above, we suggest a restarted algorithm which decreases the value of α during training. Starting from $\alpha = 1$ and some $x_0 \in \mathcal{C}$, we run Algorithm 9 for T_α iterations with m_α computers in parallel to obtain some approximate solution $x_\alpha \in \mathcal{C}$. We then decrease α by a constant factor $c < 1$ and run Algorithm 9 again with the new value of α starting from x_α. We repeat this process until a satisfying solution is obtained. We formalize this procedure in Algorithm 10. In practice we choose $c = 0.5$ and set $T_\alpha = \sqrt{\frac{L}{\alpha}} \log \frac{1}{\alpha}$ as a simplified version of the bound (2.25), and run the above procedure for both $m_\alpha = 1$ and $m_\alpha = 1/\sqrt{\alpha}$ computers in parallel.

Algorithm 10 Restarted Parallel Frank-Wolfe (R-PFW)

Input: $(L, x_0, R_{\mathcal{C}}, M, c)$. L-smooth convex function f, $x_0 \in \mathcal{C}$, $R_{\mathcal{C}} = \max_{x \in \mathcal{C}} \|x\|_*$, distribution with density $d\pi(z) \propto \exp(-\eta(z))dz$ such that $M^2 = \mathbb{E}_\Delta \|\nabla \eta(\Delta)\|_*^2$, decreasing factor c.

Set $\alpha = 1$.
for $i = 0, 1, \ldots$ **do**
  Set $T_\alpha = \sqrt{\frac{L}{\alpha}} \log \frac{1}{\alpha}$ and either set $m_\alpha = \frac{1}{\sqrt{\alpha}}$ or $m_\alpha = 1$.
  $x_\alpha = \text{Algorithm 9}(L, x_0, R_{\mathcal{C}}, M, m_\alpha, \alpha, T_\alpha)$
  Set $x_0 = x_\alpha$ and $\alpha = c\alpha$.
end for
Return x_α.

We test the restarted scheme on problem (2.26) as well as on a generalized matrix completion problem over the trace norm ball,

$$\min_{X \in \mathcal{C}_2} \{f_2(x) := \frac{1}{2} \|CX - D\|_F^2\} \quad \text{where} \quad \mathcal{C}_2 = \{X \in \mathbb{R}^{p \times q} \mid \|X\|_{tr} \leq 1\}, \tag{2.27}$$

where $C \in \mathbb{R}^{p \times p}$ and $D \in \mathbb{R}^{p \times q}$. Here $\|\cdot\|_F$ stands for the Frobenius norm and $\|\cdot\|_{tr}$ is the trace, or nuclear, norm. We choose Δ to have entries distributed according to a standard normal distribution. In this case $R_{\mathcal{C}_2} = 1$ and $M = \sqrt{pq}$. We compare the restarted scheme with the Frank-Wolfe algorithm with step-size $\eta_k = \frac{2}{k+1}$ (denoted FW in the plots) and with exact line search (denoted FW-LS in the plots). We set $p = 10$, $q = 8$ and plot the results

in Figure 2.2. For both problems, we observed that using $M = 1$ instead of the theoretical value ($M = \sqrt{d}$ for (2.26) and $M = \sqrt{pq}$ for (2.27)) led to significant speedups. We consequently plot both strategies. We observe that for $m_\alpha = 1/\sqrt{\alpha}$, Algorithm 10 significantly outperforms Frank-Wolfe, and for $m_\alpha = 1$ the performance is similar to Frank-Wolfe, which was expected from the discussion at the end of Section 2.4. Full code to reproduce the experiments can be found at the following link: https://github.com/bpauld/PFW.

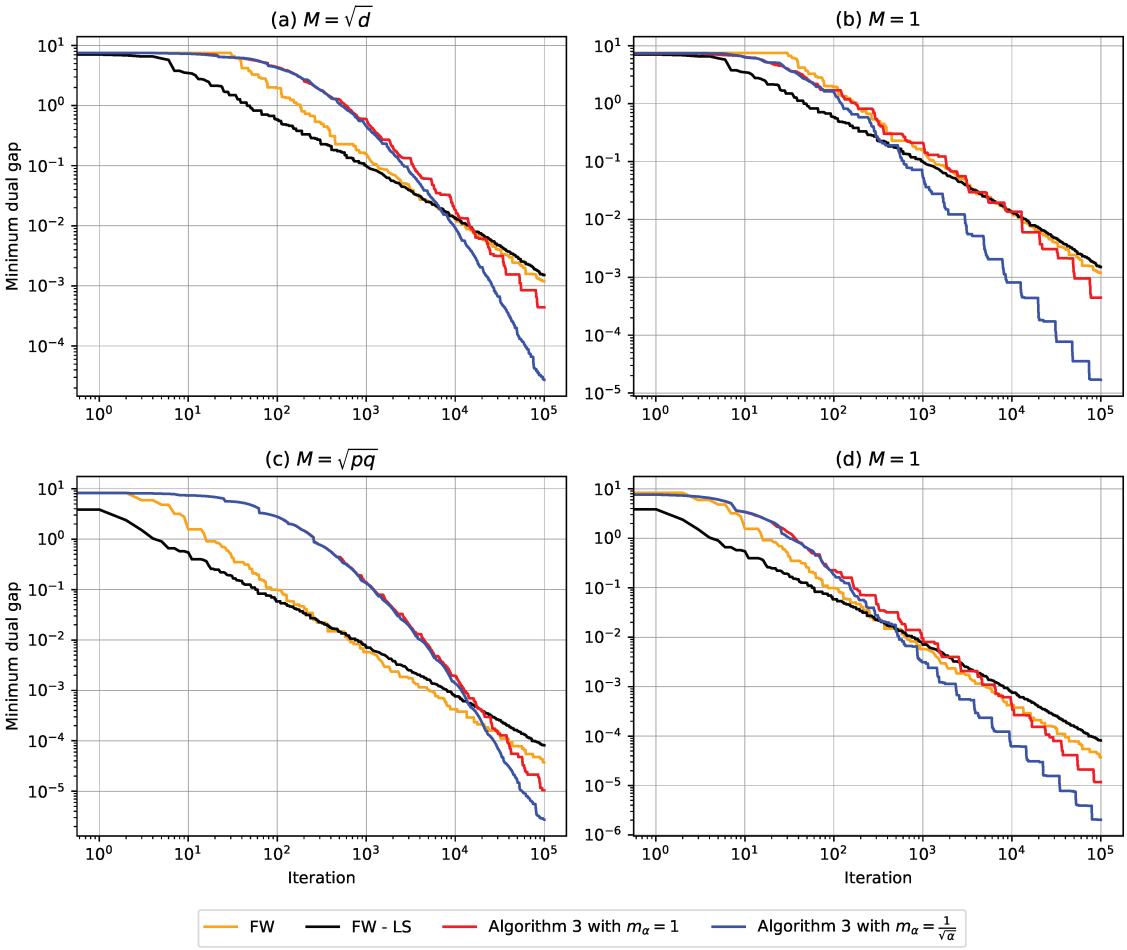

Figure 2.2: Comparisons between Frank-Wolfe and the restarting scheme (Algorithm 10): a least-squares problem on the simplex ((a) and (b)), and a generalized matrix completion problem on the trace ball ((c) and (d)). The plots report the value of the best primal-dual gap incurred at the current iteration.

Organization of the Appendix

2.A Proofs for Section 2.3

2.B Proofs for Section 2.4

2.C Experimental details

2.A Proofs for Stochastic Composite Minimization

2.A.1 Proof of Proposition 2.3.1

We break down the proof of Proposition 2.3.1 into different lemmas. The first one bounds the improvement on G over an iteration, the second one bounds the improvement H over an iteration, the third one combines the first two results and further exploits the structure of the functions at play. The proof of the proposition then follows by carefully choosing A_{k+1} as a function of A_k. In the remainder of the section, we work with the assumptions stated in Proposition 2.3.1 and do not restate them in the statements of the lemmas.

Lemma 2.A.1.

$$A_{k+1}G(y_{k+1}) - c_{k+1} \leq A_kG(y_k) - c_k + (A_{k+1} - A_k)\langle \nabla G(v_k),\, z_{k+1}\rangle + \frac{\beta}{2}\left(A_{k+1} - 2A_k + \frac{A_k^2}{A_{k+1}}\right)\|z_{k+1} - z_k\|^2 .$$

Proof. By smoothness

$$\begin{aligned}
A_{k+1}G(y_{k+1}) &\leq A_{k+1}\left(G(v_k) + \langle \nabla G(v_k),\, y_{k+1} - v_k\rangle + \frac{\beta}{2}\|y_{k+1} - v_k\|^2\right)\\
&= \underbrace{(A_{k+1} - A_k)\left(G(v_k) - \langle \nabla G(v_k),\, v_k\rangle\right)}_{=c_{k+1}-c_k} + A_k\left(G(v_k) - \langle \nabla G(v_k),\, v_k\rangle\right)\\
&\quad + A_{k+1}\langle \nabla G(v_k),\, y_{k+1}\rangle + A_{k+1}\frac{\beta}{2}\|y_{k+1} - v_k\|^2\\
&= c_{k+1} - c_k + A_k\left(G(v_k) + \langle \nabla G(v_k),\, y_k - v_k\rangle\right) - A_k\langle \nabla G(v_k),\, y_k\rangle\\
&\quad + A_{k+1}\langle \nabla G(v_k),\, y_{k+1}\rangle + A_{k+1}\frac{\beta}{2}\|y_{k+1} - v_k\|^2\\
&\leq c_{k+1} - c_k + A_kG(y_k) - A_k\langle \nabla G(v_k),\, y_k\rangle\\
&\quad + A_{k+1}\langle \nabla G(v_k),\, y_{k+1}\rangle + A_{k+1}\frac{\beta}{2}\|y_{k+1} - v_k\|^2
\end{aligned}$$

where the last inequality follows by convexity of G, Now, since $y_{k+1} = \frac{A_k}{A_{k+1}}y_k + \frac{A_{k+1}-A_k}{A_{k+1}}z_{k+1}$, we can simplify to

$$A_{k+1}G(y_{k+1}) \leq c_{k+1} - c_k + A_kG(y_k) + (A_{k+1} - A_k)\langle \nabla G(v_k),\, z_{k+1}\rangle + A_{k+1}\frac{\beta}{2}\|y_{k+1} - v_k\|^2 .$$

Finally,

$$A_{k+1}\|y_{k+1} - v_k\|^2 = A_{k+1}\tau_k^2\|z_k - z_{k+1}\|^2 = \frac{(A_{k+1} - A_k)^2}{A_{k+1}}\|z_{k+1} - z_k\|^2$$

which concludes the proof. □

Lemma 2.A.2. *For any $g_H(z_k) \in \partial H(z_k)$,*

$$A_{k+1}H(y_{k+1}) - A_{k+1}H(z_{k+1}) \leq A_kH(y_k) - A_kH(z_k) - A_k\langle g_H(z_k),\, z_{k+1} - z_k\rangle \\ - \frac{\mu}{2}A_k\left(1 - \frac{A_k}{A_{k+1}}\right)\|z_{k+1} - y_k\|^2 - A_k\frac{\mu}{2}\|z_{k+1} - z_k\|^2 .$$

Proof. By strong convexity of H, for any $g_H(y_{k+1}) \in \partial H(y_{k+1})$ we have

$$\begin{aligned} A_{k+1}H(y_{k+1}) &= (A_{k+1} - A_k)\,H(y_{k+1}) + A_kH(y_{k+1}) \\ &\leq (A_{k+1} - A_k)\left(H(z_{k+1}) - \langle g_H(y_{k+1}),\, z_{k+1} - y_{k+1}\rangle - \frac{\mu}{2}\|z_{k+1} - y_{k+1}\|^2\right) \\ &\quad + A_k\left(H(y_k) - \langle g_H(y_{k+1}),\, y_k - y_{k+1}\rangle - \frac{\mu}{2}\|y_k - y_{k+1}\|^2\right). \end{aligned}$$

Now observe that

$$\begin{aligned} z_{k+1} - y_{k+1} &= (1 - \tau_k)(z_{k+1} - y_k) = \frac{A_k}{A_{k+1}}(z_{k+1} - y_k) \\ y_k - y_{k+1} &= \tau_k(y_k - z_{k+1}) = -\frac{A_{k+1} - A_k}{A_{k+1}}(z_{k+1} - y_k) \end{aligned}$$

and thus we see that the two inner product terms cancel out. Moreover we can also simplify the norms and get

$$A_{k+1}H(y_{k+1}) \leq A_kH(y_k) + (A_{k+1} - A_k)H(z_{k+1}) \\ - \frac{\mu}{2}\|z_{k+1} - y_k\|^2 \underbrace{\left((A_{k+1} - A_k)\frac{A_k^2}{A_{k+1}^2} + A_k\frac{(A_{k+1} - A_k)^2}{A_{k+1}^2}\right)}_{=A_k - \frac{A_k^2}{A_{k+1}}}.$$

Finally, observe that by strong convexity of H again, for any $g_H(z_k) \in \partial H(z_k)$,

$$-A_kH(z_{k+1}) \leq -A_kH(z_k) - A_k\langle g_H(z_k),\, z_{k+1} - z_k\rangle - A_k\frac{\mu}{2}\|z_{k+1} - z_k\|^2 ,$$

which concludes the proof. □

Lemma 2.A.3. *Defining $m_k(x) = \langle d_k,\, x\rangle + c_k + A_kH(x) + \beta w(x)$, we have*

$$A_{k+1}F(y_{k+1}) - m_{k+1}(z_{k+1}) \leq A_kF(y_k) - m_k(z_k) \\ + (A_{k+1} - A_k)\langle \nabla G(v_k) - g_k,\, z_{k+1}\rangle \\ - \frac{1}{2}\left(A_k(\mu + 2\beta) + \beta(\nu - A_{k+1}) - \beta\frac{A_k^2}{A_{k+1}}\right)\|z_{k+1} - z_k\|^2 \\ - \frac{\mu}{2}A_k\left(1 - \frac{A_k}{A_{k+1}}\right)\|z_{k+1} - y_k\|^2 .$$

Proof. Summing the inequalities in the above two lemmas above we have that for any $g_H(z_k) \in \partial H(z_k)$,

$$A_{k+1}F(y_{k+1}) - c_{k+1} - A_{k+1}H(z_{k+1}) \leq A_kF(y_k) - c_k - A_kH(z_k) \\ + (A_{k+1} - A_k)\langle \nabla G(v_k),\, z_{k+1}\rangle - A_k\langle g_H(z_k),\, z_{k+1} - z_k\rangle \\ - \frac{1}{2}\left(A_k(\mu + 2\beta) - \beta A_{k+1} - \beta\frac{A_k^2}{A_{k+1}}\right)\|z_{k+1} - z_k\|^2 \\ - \frac{\mu}{2}A_k\left(1 - \frac{A_k}{A_{k+1}}\right)\|z_{k+1} - y_k\|^2 .$$

Substracting $\langle d_{k+1},\, z_{k+1}\rangle$ on both sides and adding/substracting $\langle d_k,\, z_{k+1}\rangle$ and $\langle d_k,\, z_k\rangle$ on the right-hand side, we get

$$\begin{aligned}
A_{k+1}F(y_{k+1}) - c_{k+1} - A_{k+1}H(z_{k+1}) - \langle d_{k+1},\, z_{k+1}\rangle & \\
&\leq A_kF(y_k) - c_k - A_kH(z_k) - \langle d_k,\, z_k\rangle \\
&\quad + \langle d_k,\, z_k\rangle - \langle d_{k+1},\, z_{k+1}\rangle - \langle d_k,\, z_{k+1}\rangle + \langle d_k,\, z_{k+1}\rangle \\
&\quad + (A_{k+1} - A_k)\langle \nabla G(v_k),\, z_{k+1}\rangle - A_k\langle g_H(z_k),\, z_{k+1} - z_k\rangle \\
&\quad - \frac{1}{2}\left(A_k(\mu + 2\beta) - \beta A_{k+1} - \beta\frac{A_k^2}{A_{k+1}}\right)\|z_{k+1} - z_k\|^2 \\
&\quad - \frac{\mu}{2}A_k\left(1 - \frac{A_k}{A_{k+1}}\right)\|z_{k+1} - y_k\|^2 \\
&= A_kF(y_k) - c_k - A_kH(z_k) - \langle d_k,\, z_k\rangle \\
&\quad + \langle d_k - d_{k+1},\, z_{k+1}\rangle \\
&\quad + (A_{k+1} - A_k)\langle \nabla G(v_k),\, z_{k+1}\rangle - \langle A_k g_H(z_k) + d_k,\, z_{k+1} - z_k\rangle \\
&\quad - \frac{1}{2}\left(A_k(\mu + 2\beta) - \beta A_{k+1} - \beta\frac{A_k^2}{A_{k+1}}\right)\|z_{k+1} - z_k\|^2 \\
&\quad - \frac{\mu}{2}A_k\left(1 - \frac{A_k}{A_{k+1}}\right)\|z_{k+1} - y_k\|^2 .
\end{aligned}$$

Now, by first-order optimality conditions of (2.13), $0 \in d_k + A_k\partial H(z_k) + \beta\partial w(z_k)$. Therefore there exist subgradients $g'_H(z_k) \in \partial H(z_k)$ and $g'_w(z_k) \in \partial w(z_k)$ such that $d_k + A_k g'_H(z_k) = -\beta g'_w(z_k)$. Since the above inequality is true for any $g_H(z_k) \in \partial H(z_k)$, it is in particular true for $g'_H(z_k)$, and thus we have

$$\begin{aligned}
A_{k+1}F(y_{k+1}) - c_{k+1} - A_{k+1}H(z_{k+1}) - \langle d_{k+1},\, z_{k+1}\rangle & \\
&\leq A_kF(y_k) - c_k - A_kH(z_k) - \langle d_k,\, z_k\rangle \\
&\quad + \langle d_k - d_{k+1},\, z_{k+1}\rangle \\
&\quad + (A_{k+1} - A_k)\langle \nabla G(v_k),\, z_{k+1}\rangle + \beta\langle g'_w(z_k),\, z_{k+1} - z_k\rangle \\
&\quad - \frac{1}{2}\left(A_k(\mu + 2\beta) - \beta A_{k+1} - \beta\frac{A_k^2}{A_{k+1}}\right)\|z_{k+1} - z_k\|^2 \\
&\quad - \frac{\mu}{2}A_k\left(1 - \frac{A_k}{A_{k+1}}\right)\|z_{k+1} - y_k\|^2 .
\end{aligned}$$

Since $d_k - d_{k+1} = -(A_{k+1} - A_k)g_k$ we get

$$\begin{aligned}
A_{k+1}F(y_{k+1}) - c_{k+1} - A_{k+1}H(z_{k+1}) - \langle d_{k+1},\, z_{k+1}\rangle \leq\ & A_kF(y_k) - c_k - A_kH(z_k) - \langle d_k,\, z_k\rangle \\
&+ (A_{k+1} - A_k)\langle \nabla G(v_k) - g_k,\, z_{k+1}\rangle \\
&+ \beta\langle g'_w(z_k),\, z_{k+1} - z_k\rangle \\
&- \frac{1}{2}\left(A_k(\mu + 2\beta) - \beta A_{k+1} - \beta\frac{A_k^2}{A_{k+1}}\right)\|z_{k+1} - z_k\|^2 \\
&- \frac{\mu}{2}A_k\left(1 - \frac{A_k}{A_{k+1}}\right)\|z_{k+1} - y_k\|^2 .
\end{aligned}$$

Finally, by strong convexity of w we have

$$\beta\langle g'_w(z_k),\, z_{k+1} - z_k\rangle \leq \beta\left(w(z_{k+1}) - w(z_k) - \frac{\nu}{2}\|z_{k+1} - z_k\|^2\right),$$

and thus the previous inequality becomes

$$\begin{aligned} A_{k+1}F(y_{k+1}) - m_{k+1}(z_{k+1}) \leq\, & A_kF(y_k) - m_k(z_k) \\ & + (A_{k+1} - A_k)\langle \nabla G(v_k) - g_k,\, z_{k+1}\rangle \\ & - \frac{1}{2}\left(A_k(\mu + 2\beta) + \beta(\nu - A_{k+1}) - \beta\frac{A_k^2}{A_{k+1}}\right)\|z_{k+1} - z_k\|^2 \\ & - \frac{\mu}{2}A_k\left(1 - \frac{A_k}{A_{k+1}}\right)\|z_{k+1} - y_k\|^2 . \end{aligned}$$

□

Those three lemmas allow us to prove Proposition 2.3.1.

Proof of Proposition 2.3.1.

We can rewrite the previous result as

$$\begin{aligned} A_{k+1}F(y_{k+1}) - m_{k+1}(z_{k+1}) \leq\, & A_kF(y_k) - m_k(z_k) \\ & + (A_{k+1} - A_k)\langle \nabla G(v_k) - g_k,\, z_{k+1} - z_k\rangle \\ & + (A_{k+1} - A_k)\langle \nabla G(v_k) - g_k,\, z_k\rangle \\ & - \frac{1}{2}\left(A_k(\mu + 2\beta) + \beta(\nu - A_{k+1}) - \beta\frac{A_k^2}{A_{k+1}}\right)\|z_{k+1} - z_k\|^2 \\ & - \frac{\mu}{2}A_k\left(1 - \frac{A_k}{A_{k+1}}\right)\|z_{k+1} - y_k\|^2 . \end{aligned}$$

Taking expectation at iteration k conditioned on the previous iterations, we have $\mathbb{E}_k[(A_{k+1} - A_k)\langle \nabla G(v_k) - g_k,\, z_k\rangle = 0]$.

Moreover, by Fenchel-Young inequality we have that for any $\rho > 0$,

$$\langle \nabla G(v_k) - g_k,\, z_{k+1} - z_k\rangle \leq \frac{1}{2\rho}\|\nabla G(v_k) - g_k\|_*^2 + \frac{\rho}{2}\|z_{k+1} - z_k\|^2 .$$

Taking expectation and using Assumption 2.3.1,

$$(A_{k+1} - A_k)\mathbb{E}_k[\langle \nabla G(v_k) - g_k,\, z_{k+1} - z_k\rangle] \leq \frac{1}{2}(A_{k+1} - A_k)\frac{\sigma^2}{\rho} + \frac{1}{2}(A_{k+1} - A_k)\rho\mathbb{E}_k\|z_{k+1} - z_k\|^2$$

Thus we have

$$\begin{aligned} \mathbb{E}_k\left[A_{k+1}F(y_{k+1}) - m_{k+1}(z_{k+1})\right] \leq\, & A_kF(y_k) - m_k(z_k) + \frac{1}{2}(A_{k+1} - A_k)\frac{\sigma^2}{\rho} \\ & - \frac{1}{2}\left(A_k(\mu + 2\beta) + \beta(\nu - A_{k+1}) - \beta\frac{A_k^2}{A_{k+1}}\right)\mathbb{E}_k\|z_{k+1} - z_k\|^2 \\ & + \frac{1}{2}\rho(A_{k+1} - A_k)\mathbb{E}_k\|z_{k+1} - z_k\|^2 \\ & - \frac{\mu}{2}A_k\left(1 - \frac{A_k}{A_{k+1}}\right)\mathbb{E}_k\|z_{k+1} - y_k\|^2 . \end{aligned}$$

Now, observe that since $0 \leq A_k/A_{k+1} \leq 1$, the term in $\mathbb{E}_k\|z_{k+1} - y_k\|^2$ is non-positive. Therefore, to obtain the final result, it suffices to set A_{k+1} so that the term in $\mathbb{E}_k\|z_{k+1} - z_k\|^2$ cancels

out. In other words, we require

$$
\begin{aligned}
& A_k(\mu+2\beta)+\beta(\nu-A_{k+1})-\beta\frac{A_k^2}{A_{k+1}}-\rho(A_{k+1}-A_k)=0 \\
\iff & A_{k+1}(\beta+\rho)-A_k(\mu+2\beta+\rho)-\beta\nu+\beta\frac{A_k^2}{A_{k+1}}=0 \\
\iff & A_{k+1}^2(\beta+\rho)-A_{k+1}\left(A_k(\mu+2\beta+\rho)+\beta\nu\right)+\beta A_k^2=0 \\
\Leftarrow \quad & A_{k+1}=\frac{A_k(\mu+2\beta+\rho)+\beta\nu+\sqrt{\left(A_k(\mu+2\beta+\rho)+\beta\nu\right)^2-4(\beta+\rho)\beta A_k^2}}{2(\beta+\rho)} \\
\iff & A_{k+1}=\frac{A_k(\mu+2\beta+\rho)+\beta\nu+\sqrt{\beta^2\nu^2+2\beta\nu A_k(\mu+2\beta+\rho)+A_k^2\mu^2+A_k^2\rho^2+2A_k^2\mu\rho+4A_k^2\mu\beta}}{2(\beta+\rho)} \\
\iff & A_{k+1}=\frac{A_k(\mu+2\beta+\rho)+\beta\nu+\sqrt{(\beta\nu+\mu A_k)^2+4A_k(\beta^2\nu+A_k\mu\beta)+2\beta\nu A_k\rho+A_k^2\rho^2+2A_k^2\mu\rho}}{2(\beta+\rho)}.
\end{aligned}
$$

Setting $\rho=\sqrt{\mu\beta}$ yields the update for A_{k+1} in Algorithm 8 and proves the proposition.

2.A.2 Proof of Theorem 2.3.1

Proof. Unrolling the recursion in Proposition 2.3.1 and taking total expectation, we have

$$
\mathbb{E}\left[A_kF(y_k)-m_k(z_k)\right]\le A_0F(y_0)-m_0(z_0)+A_k\frac{1}{2\sqrt{\mu\beta}}\sigma^2=-\beta w(z_0)+A_k\frac{1}{2\sqrt{\mu\beta}}\sigma^2
$$

Now,

$$
m_k(y_\star)=A_kH(y_\star)+\beta w(y_\star)+\sum_{t=0}^{k-1}\left(A_{t+1}-A_t\right)\left(G(v_t)-\langle\nabla G(v_t),\,v_t\rangle+\langle g_t,\,y_\star\rangle\right)
$$

Taking total expectation on the g_t we get

$$
\begin{aligned}
\mathbb{E}[m_k(y_\star)] &= A_kH(y_\star)+\beta w(y_\star)+\sum_{t=0}^{k-1}\left(A_{t+1}-A_t\right)\mathbb{E}\left[G(v_t)+\langle\nabla G(v_t),\,y_\star-v_t\rangle\right] \\
&\le A_kH(y_\star)+\beta w(y_\star)+A_kG(y_\star)=A_kF(y_\star)+\beta w(y_\star)
\end{aligned}
$$

where the inequality is by convexity of G. Moreover, $m_k(z_k)\le m_k(y_\star)$ by construction and thus

$$
\begin{aligned}
\mathbb{E}[A_kF(y_k)-A_kF(y_\star)-\beta w(y_\star)] &\le \mathbb{E}[A_kF(y_k)-m_k(y_\star)] \\
&\le \mathbb{E}[A_kF(y_k)-m_k(z_k)] \\
&\le -\beta w(z_0)+\frac{A_k}{2\sqrt{\mu\beta}}\sigma^2
\end{aligned}
$$

and thus

$$
\begin{aligned}
\mathbb{E}[F(y_k)-F(y_\star)] &\le \frac{\beta(w(y_\star)-w(z_0))}{A_k}+\frac{\sigma^2}{2\sqrt{\mu\beta}} \\
&= \frac{\beta D_w(y_\star,y_0)}{A_k}+\frac{\sigma^2}{2\sqrt{\mu\beta}}
\end{aligned}
$$

as $y_0=z_0$ and $w(y_\star)-w(y_0)$ is equal to $D_w(y_\star,y_0):=w(y_\star)-w(y_0)-\langle g_w(y_0);y_\star-y_0\rangle$ (with $g_w(y_0)\in\partial w(y_0)$) through the choice $g_w(y_0)=0\in\partial w(y_0)$, which is valid as y_0 minimizes $w(\cdot)$.

Finally, we can bound A_{k+1} as

$$\begin{aligned} A_{k+1} &\geq A_k \frac{\mu + 2\beta + \sqrt{\mu\beta} + \sqrt{\mu^2 + 4\mu\beta + \sqrt{\mu\beta}^2 + 2\mu\sqrt{\mu\beta}}}{2(\beta + \sqrt{\mu\beta})} \\ &\geq A_k \frac{\mu + 2\beta + \sqrt{\mu\beta} + 2\sqrt{\mu\beta}}{2(\beta + \sqrt{\mu\beta})} \\ &\geq A_k \left(1 + \frac{\sqrt{\mu\beta}}{2(\beta + \sqrt{\beta\mu})}\right) \\ &= A_k \left(1 + \frac{\sqrt{\mu}}{2(\sqrt{\beta} + \sqrt{\mu})}\right), \end{aligned}$$

with $A_1 = \frac{\nu\sqrt{\beta}}{\sqrt{\beta}+\sqrt{\mu}}$, leading to

$$\begin{aligned} A_k &\geq \frac{\nu\sqrt{\beta}}{\sqrt{\beta} + \sqrt{\mu}} \left(1 + \frac{\sqrt{\mu}}{2(\sqrt{\beta} + \sqrt{\mu})}\right)^{k-1} \\ &\geq \frac{\nu\sqrt{\beta}}{\sqrt{\beta} + \sqrt{\mu}} \exp\left(\frac{(k-1)\sqrt{\mu}}{4(\sqrt{\beta} + \sqrt{\mu})}\right), \end{aligned}$$

for $k \geq 1$, where we used that $1 + \frac{x}{2} \geq e^{\frac{x}{4}}$ for $0 \leq x \leq 1$. Therefore,

$$\frac{1}{A_k} \leq \frac{\sqrt{\beta} + \sqrt{\mu}}{\nu\sqrt{\beta}} \exp\left(-\frac{(k-1)\sqrt{\mu}}{4\left(\sqrt{\beta} + \sqrt{\mu}\right)}\right)$$

To conclude,

$$\mathbb{E}[F(y_k) - F^*] \leq \frac{\sqrt{\beta} + \sqrt{\mu}}{\nu\sqrt{\beta}} \exp\left(-\frac{(k-1)\sqrt{\mu}}{4\left(\sqrt{\beta} + \sqrt{\mu}\right)}\right) \beta D_w(y_\star, y_0) + \frac{\sigma^2}{2\sqrt{\mu\beta}}.$$

□

2.B Proofs for Accelerated Frank-Wolfe

2.B.1 Proof of Theorem 2.4.1

2.B.1.1 Proof of feasibility

First we show that $x_k \in \mathcal{C}$ for all $k \in \mathbb{N}$.

Proof. We prove this by induction. By assumption $x_0 \in \mathcal{C}$. Now suppose $x_k \in \mathcal{C}$ for some $k \in \mathbb{N}$. We then have

$$\begin{aligned} x_{k+1} &= \frac{\beta}{A_{k+1} + \beta} x_0 - \frac{d_{k+1}}{A_{k+1} + \beta} \\ &= \frac{\beta}{A_{k+1} + \beta} x_0 - \frac{d_k + (A_{k+1} - A_k) g_k}{A_{k+1} + \beta} \\ &= \frac{(A_k + \beta)\left(\frac{\beta}{A_k+\beta} x_0 - \frac{d_k}{A_k+\beta}\right) - (A_{k+1} - A_k) g_k}{A_{k+1} + \beta} \\ &= \frac{(A_k + \beta)}{A_{k+1} + \beta} x_k + \frac{A_{k+1} - A_k}{A_{k+1} + \beta}(-g_k) \\ &= \frac{A_k + \beta}{A_{k+1} + \beta} x_k + \left(1 - \frac{A_k + \beta}{A_{k+1} + \beta}\right)(-g_k). \end{aligned}$$

By induction hypothesis, $x_k \in \mathcal{C}$. We also have $-g_k \in \mathcal{C}$ since

$$\begin{aligned} -g_k &= -\frac{1}{m}\sum_{i=1}^{m} g_{k,i} \\ &= \frac{1}{m}\sum_{i=1}^{m} \arg\max_{u\in\mathcal{C}} \langle u,\, -v_k + \alpha\Delta_i\rangle. \end{aligned}$$

In other words, $-g_k$ is a convex combination of elements of $\mathcal{C}$ and is thus in $\mathcal{C}$. Therefore $x_{k+1} \in \mathcal{C}$ as a convex combination of elements of $\mathcal{C}$. □

2.B.1.2 Proof of dual gap convergence

Proof. With $H(y) = f^*(y)$, $G(y) = s_\alpha(-y)$, $\beta = \frac{R_{\mathcal{C}} M}{\alpha}$ and $\mu = \frac{1}{L}$, we can apply Proposition 2.3.1 to $F = H + G$ and get

$$\mathbb{E}_k[A_{k+1}F(y_{k+1}) - m_{k+1}(z_{k+1})] \leq A_k F(y_k) - m_k(z_k) + (A_{k+1} - A_k)\frac{\sigma^2}{2\sqrt{\mu\beta}}$$

where $m_k(y) = \langle d_k,\, y\rangle + c_k + A_k H(y) + \beta w(y)$, and $z_k = \nabla f(x_k)$. Unrolling the recursion as before and taking total expectation we have

$$\mathbb{E}[A_k F(y_k) - m_k(z_k)] \leq -\beta w(z_0) + A_k \frac{\sigma^2}{2\sqrt{\mu\beta}}.$$

Recall that we set $w(y) = f^*(y) - \langle x_0,\, y\rangle$. Plugging in the function $H = f^*$, and recalling that z_k minimizes $m_k(y)$, we get

$$\begin{aligned} m_k(z_k) &= \inf_y \{\langle d_k,\, y\rangle + c_k + (A_k + \beta) f^*(y) - \beta\langle x_0,\, y\rangle\} \\ &= c_k - \sup_y \{\langle -d_k + \beta x_0,\, y\rangle - (A_k + \beta) f^*(y)\} \\ &= c_k - (A_k + \beta) f\left(\frac{-d_k + \beta x_0}{A_k + \beta}\right) \\ &= c_k - (A_k + \beta) f(x_k). \end{aligned}$$

Thus we can conclude that

$$\mathbb{E}[A_k F(y_k) + (A_k + \beta) f(x_k)] \leq -\beta w(z_0) + A_k \frac{\sigma^2}{2\sqrt{\mu\beta}} + c_k,$$

and in particular

$$\mathbb{E}[A_k F(y_k) + A_k f(x_k)] \leq -\beta f(x_\star) - \beta w(z_0) + A_k \frac{\sigma^2}{2\sqrt{\mu\beta}} + c_k. \tag{2.28}$$

Now,

$$c_k = \sum_{i=0}^{k-1} (A_{i+1} - A_i)(G(v_i) - \langle \nabla G(v_i),\, v_i\rangle).$$

For any $v \in \mathbf{V}$, $G(v) = s_\alpha(-v)$ and thus

$$G(v) - \langle \nabla G(v),\, v\rangle = s_\alpha(-v) + \langle \nabla s_\alpha(-v),\, v\rangle.$$

From Fenchel-Young, $\langle \nabla s_\alpha(-v),\, -v\rangle = s_\alpha(-v) + s_\alpha^*(\nabla s_\alpha(-v))$, and thus

$$G(v) - \langle \nabla G(v),\, v\rangle = -s_\alpha^*(\nabla s_\alpha(-v)).$$

Now, for all $u \in \mathbf{V}^*$,

$$\begin{aligned} s_\alpha^*(u) &= \sup_{v \in \mathbf{V}} \{\langle u, v\rangle - s_\alpha(v)\} \\ &\geq \sup_{v \in \mathbf{V}} \{\langle u, v\rangle - s(v) - \alpha s_1(0)\} \\ &= s^*(u) - \alpha s_1(0) \\ &= I_{\mathcal{C}}(u) - \alpha s_1(0) \end{aligned}$$

where the inequality is from Proposition 2.2.2. In particular, since $\nabla s_\alpha(-v)$ is always feasible, we have $s_\alpha^*(\nabla s_\alpha(-v)) \geq -\alpha s_1(0)$. and thus

$$c_k \leq \sum_{i=0}^{k-1} (A_{i+1} - A_i)\alpha s_1(0) = A_k \alpha s_1(0).$$

We can then rewrite (2.28) as

$$\mathbb{E}[A_k f^*(y_k) + A_k s_\alpha(-y_k) + A_k f(x_k)] \leq -\beta w(z_0) - \beta f(x_\star) + A_k \frac{\sigma^2}{2\sqrt{\mu\beta}} + A_k \alpha s_1(0).$$

Now,

$$w(z_0) = f^*(z_0) - \langle x_0, z_0\rangle = -f(x_0)$$

by Fenchel-Young and since $z_0 = \nabla f(x_0)$. Thus

$$\mathbb{E}[A_k f^*(y_k) + A_k s_\alpha(-y_k) + A_k f(x_k)] \leq \beta(f(x_0) - f(x_\star)) + A_k \frac{\sigma^2}{2\sqrt{\mu\beta}} + A_k \alpha s_1(0).$$

Finally, from Proposition 2.2.2,

$$s_\alpha(-y) \geq s(-y)$$

for any y. We can then conclude

$$\mathbb{E}\left[f^*(y_k) + s(-y_k) + f(x_k)\right] \leq \frac{\beta(f(x_0) - f(x_\star))}{A_k} + \frac{\sigma^2}{2\sqrt{\mu\beta}} + \alpha s_1(0)$$

Bounding A_k as in Theorem 2.3.1 yields

$$\mathbb{E}\left[f^*(y_k) + s(-y_k) + f(x_k)\right] \leq \frac{\sqrt{\beta} + \sqrt{\mu}}{\nu\sqrt{\beta}} \exp\left(-\frac{(k-1)\sqrt{\mu}}{4(\sqrt{\beta} + \sqrt{\mu})}\right) \beta(f(x_0) - f(x_\star)) + \frac{\sigma^2}{2\sqrt{\mu\beta}} + \alpha s_1(0) \tag{2.29}$$

It remains to bound the variance σ^2. Using Assumption 2.4.1 we have

$$\sigma^2 = \mathbb{E}\left\|g_k - \nabla G(v_k)\right\|_*^2 \leq \frac{4R_{\mathcal{C}}^2 \rho_{\|\cdot\|_*}}{m}.$$

Plugging this back into equation (2.29), and plugging in the values of $\beta = \frac{R_{\mathcal{C}} M}{\alpha}$ and $\mu = \frac{1}{L}$ gives

$$\begin{aligned} \mathbb{E}\left[f(x_k) - d(y_k)\right] &\leq \frac{\sqrt{\frac{R_K M}{\alpha}} + \sqrt{\frac{1}{L}}}{\frac{1}{L}\sqrt{\frac{R_K M}{\alpha}}} \exp\left(-\frac{(k-1)\sqrt{\frac{1}{L}}}{4\left(\sqrt{\frac{R_K M}{\alpha}} + \sqrt{\frac{1}{L}}\right)}\right) \frac{R_K M}{\alpha}\left(f(x_0) - f(x_\star)\right) \\ &\quad + \frac{2R_K^2 \rho_{\|\cdot\|_*}}{m}\sqrt{\frac{\alpha L}{R_K M}} + \alpha s_1(0). \\ &= \exp\left(-\frac{(k-1)\sqrt{\frac{1}{L}}}{4\left(\sqrt{\frac{R_K M}{\alpha}} + \sqrt{\frac{1}{L}}\right)}\right) L\left(\sqrt{\frac{R_K M}{\alpha}} + \sqrt{\frac{1}{L}}\right)\sqrt{\frac{R_K M}{\alpha}}\left(f(x_0) - f(x_\star)\right) \\ &\quad + \frac{2R_K^2 \rho_{\|\cdot\|_*}}{m}\sqrt{\frac{\alpha L}{R_K M}} + \alpha s_1(0). \end{aligned}$$

Assuming $\frac{R_{\mathcal{C}} M}{\alpha} \geq \frac{1}{L}$ yields the result. □

2.B.2 Proof of Theorem 2.4.2

Proof. Setting

$$\alpha = \min\left\{\frac{\epsilon}{3s_1(0)}, \frac{M\epsilon^2 m^2}{36LR_{\mathcal{C}}^3\rho_{\|\cdot\|_*}^2}\right\},$$

it is easy to verify that

$$\alpha s_1(0) \leq \frac{\epsilon}{3},$$

and that

$$\frac{2R_{\mathcal{C}}^2\rho_{\|\cdot\|_*}}{m}\sqrt{\frac{\alpha L}{R_{\mathcal{C}}M}} \leq \frac{\epsilon}{3}.$$

It remains to compute k such that the first term in the bound of Theorem 2.4.1 is also smaller than $\epsilon/3$. This gives

$$\begin{aligned}
&\exp\left(-(k-1)\frac{\sqrt{\alpha}}{8\sqrt{LR_{\mathcal{C}}M}}\right)\frac{2LR_{\mathcal{C}}M}{\alpha}\left(f(x_0)-f(x_\star)\right) \leq \frac{\epsilon}{3}\\
\iff &k \geq 1 + \frac{8\sqrt{LR_{\mathcal{C}}M}}{\sqrt{\alpha}}\log\left(\frac{6LR_{\mathcal{C}}M(f(x_0)-f(x_\star))}{\epsilon\alpha}\right).
\end{aligned}$$

The $\tilde{O}$-complexity follows directly from plugging the value of α in the bound. □

2.B.3 Dependence on the norms

In this section, we compute the value of $\rho_{\|\cdot\|_*}$ for different ℓ_p norms when the underlying vector space has dimension d.

2.B.3.1 Euclidean norm

In the case of the Euclidean norm, we have $\|\cdot\| = \|\cdot\|_* = \|\cdot\|_2$ and thus

$$\begin{aligned}
\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^m g_{k,i} - \nabla G(v_k)\right\|^2 &= \mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^m g_{k,i} + \nabla s_\alpha(-v_k)\right\|^2\\
&= \frac{1}{m^2}\sum_{i=1}^m \mathbb{E}\left\|g_{k,i} + \nabla s_\alpha(-v_k)\right\|^2\\
&\quad + \frac{2}{m^2}\sum_{1\leq i<j\leq m}\mathbb{E}[\langle g_{k,i} + \nabla s_\alpha(-v_k),\, g_{k,j} + \nabla s_\alpha(-v_k)\rangle]\\
&= \frac{1}{m^2}\sum_{i=1}^m \mathbb{E}\left\|g_{k,i} + \nabla s_\alpha(-v_k)\right\|^2\\
&= \frac{1}{m^2}\sum_{i=1}^m \mathbb{E}\left\|-\arg\max_{u\in\mathcal{C}}\langle u,\, -v_k + \alpha\Delta_i\rangle + \nabla s_\alpha(-v_k)\right\|^2
\end{aligned}$$

where the second equality simply comes from the properties of the Euclidean norm, and the third one comes from the fact that $g_{k,i} + \nabla s_\alpha(-v_k)$ and $g_{k,j} + \nabla s_\alpha(-v_k)$ are zero-mean independent random variables for all $i \neq j$.

Finally, $\arg\max_{u\in\mathcal{C}}\langle u,\, v\rangle \in \mathcal{C}$ for any v, and similarly $\nabla s_\alpha(v) = \mathbb{E}\left[\arg\max_{u\in\mathcal{C}}\langle u,\, v+\alpha\Delta\rangle\right] \in \mathcal{C}$ for all v. Therefore we have

$$\begin{aligned}\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^{m} g_{k,i} - \nabla G(v_k)\right\|^2 &\le \frac{1}{m^2}\sum_{i=1}^{m}\max_{u,v\in\mathcal{C}}\|u-v\|^2 \\ &\le \frac{1}{m}\max_{u,v\in\mathcal{C}}\left(\|u\|+\|v\|\right)^2 \\ &\le \frac{1}{m}\max_{u,v\in\mathcal{C}} 2\|u\|^2 + 2\|v\|^2 \\ &= \frac{4R_{\mathcal{C}}^2}{m}\end{aligned}$$

and we see that in this case $\rho_{\|\cdot\|_2} = 1$.

2.B.3.2 ℓ_p-norms for $2 \le p < \infty$

When $\|\cdot\|_* = \|\cdot\|_p$, we have $\|\cdot\| = \|\cdot\|_q$ for $\frac{1}{p}+\frac{1}{q}=1$. Since $q \in (1,2]$, from [BCL94] we know that $\frac{1}{2}\|\cdot\|_q^2$ is $(q-1)$-strongly convex with respect to $\|\cdot\|_q$. Therefore $\frac{1}{2}\|\cdot\|_p^2$ is $\frac{1}{q-1}$-smooth with respect to $\|\cdot\|_p$ [KSST+09]. We have

$$\frac{1}{q-1} = \frac{1/q}{1-1/q} = \frac{p}{q} = p-1$$

so $\frac{1}{2}\|\cdot\|_p^2$ is $(p-1)$-smooth with respect to $\|\cdot\|_p$. We now closely follow the proof from [KST08, Lemma 2]. Let $F = \frac{1}{2}\|\cdot\|_p^2$ and let $Z_i = g_{k,i} - \nabla G(v_k)$ so that $\mathbb{E}\|Z_i\|_p^2 \le 4R_{\mathcal{C}}^2$. Let $S_i = \sum_{j=1}^{i-1} S_j$. By smoothness of F we have

$$F(S_{i-1}+Z_i) \le F(S_i) + \langle \nabla F(S_{i-1}),\, Z_i\rangle + \frac{p-1}{2}\|Z_i\|_p^2$$

Taking conditional expectation with respect to $Z_1,\ldots,Z_{i-1}$, since $\mathbb{E}[Z_i]=0$ we have

$$\begin{aligned}\mathbb{E}_i[F(S_i) \mid Z_1,\ldots,Z_{i-1}] &\le F(S_{i-1}) + \frac{p-1}{2}\mathbb{E}\left[\|Z_i\|_p^2 \mid Z_1,\ldots,Z_{i-1}\right] \\ &\le F(S_{i-1}) + \frac{p-1}{2}4R_{\mathcal{C}}^2.\end{aligned}$$

Thus, $F(S_i) - \frac{i(p-1)}{2}$ is a supermartingale and therefore

$$\mathbb{E}[F(S_n)] = \mathbb{E}\left[\frac{1}{2}\left\|\sum_{i=1}^{m} g_{k,i} - \nabla G(v_k)\right\|_p^2\right] \le \frac{m(p-1)4R_{\mathcal{C}}^2}{2}$$

which shows that $\rho_{\|\|_p} = p-1$.

2.B.3.3 ℓ_p-norms for $1 \le p < 2$

Recall that for $\infty \ge q > r \ge 1$,

$$\|\cdot\|_q \le \|\cdot\|_r \le d^{1/r-1/q}\|\cdot\|_q. \tag{2.30}$$

If the norm of interest is $\|\cdot\|_* = \|\cdot\|_p$ for $1 \le p < 2$, we have

$$\begin{aligned}\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^m g_{k,i} - \nabla G(v_k)\right\|_*^2 &= \mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^m g_{k,i} - \nabla G(v_k)\right\|_p^2 \\ &\le \left(d^{1/p-1/2}\right)^2 \mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^m g_{k,i} - \nabla G(v_k)\right\|_2^2 \\ &\le d^{(2/p-1)}\frac{1}{m}\max_{u,v\in\mathcal{C}}\|u-v\|_2^2\end{aligned}$$

where the second inequality comes from the derivation for the Euclidean norm in Section 2.B.3.1. Using (2.30) we get

$$\begin{aligned}\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^m g_{k,i} - \nabla G(v_k)\right\|_*^2 &\le d^{(2/p-1)}\frac{1}{m}\max_{u,v\in\mathcal{C}}\|u-v\|_p^2 \\ &= d^{(2/p-1)}\frac{4R_{\mathcal{C}}^2}{m}\end{aligned}$$

and thus $\rho_{\|\cdot\|_p} = d^{(2/p-1)}$.

2.B.3.4 ℓ_∞-norm

When $\|\cdot\|_* = \|\cdot\|_\infty$, we use inequality (2.30) to get that for any $1 \le r < \infty$,

$$\begin{aligned}\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^m g_{k,i} - \nabla G(v_k)\right\|_*^2 &= \mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^m g_{k,i} - \nabla G(v_k)\right\|_\infty^2 \\ &\le \mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^m g_{k,i} - \nabla G(v_k)\right\|_r^2.\end{aligned}$$

Now, if $r \ge 2$, from Section 2.B.3.2 we have that

$$\begin{aligned}\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^m g_{k,i} - \nabla G(v_k)\right\|_r^2 &\le \frac{r-1}{m}\max_{u,v\in\mathcal{C}}\|u-v\|_r^2 \\ &\le \frac{(r-1)}{m}d^{2/r}\max_{u,v\in\mathcal{C}}\|u-v\|_\infty^2 \\ &= \frac{(r-1)4R_{\mathcal{C}}^2}{m}d^{2/r}.\end{aligned}$$

Taking $r = 2 + \log d$ (so that $r \ge 2$), we then have

$$\begin{aligned}\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^m g_{k,i} - \nabla G(v_k)\right\|_*^2 &\le \frac{(r-1)4R_{\mathcal{C}}^2}{m}e^{\frac{2}{r}\log(d)} \\ &= \frac{4R_{\mathcal{C}}^2}{m}(\log d + 1)e^{\frac{2\log d}{2+\log d}} \\ &\le \frac{4R_{\mathcal{C}}^2}{m}2(\log d + 1)\end{aligned}$$

and thus we have $\rho_{\|\cdot\|_\infty} = 2(\log d + 1)$.

2.B.3.5 General norm

For a general norm, since we are in a finite-dimensional space, there exist constants $c, C > 0$ such that $c\|\cdot\|_2 \leq \|\cdot\|_* \leq C\|\cdot\|_2$. We then have

$$\begin{aligned}
\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^{m} g_{k,i} - \nabla G(v_k)\right\|_*^2 &\leq C^2\mathbb{E}\left\|\frac{1}{m}\sum_{i=1}^{m} g_{k,i} - \nabla G(v_k)\right\|_2^2 \\
&\leq C^2\frac{1}{m^2}\sum_{i=1}^{m}\max_{u,v\in\mathcal{C}}\|u-v\|_2^2 \\
&\leq \frac{C^2}{c^2}\frac{1}{m^2}\sum_{i=1}^{m}\max_{u,v\in\mathcal{C}}\|u-v\|_*^2 \\
&\leq \frac{C^2}{c^2}\frac{4R_{\mathcal{C}}^2}{m}
\end{aligned}$$

where the second inequality comes from the derivation in Section 2.B.3.1. Thus we $\rho_{\|\cdot\|_*} \leq \frac{C^2}{c^2}$.

2.C Experimental Details

In the experiments we only consider Euclidean norms so that $\|\cdot\| = \|\cdot\|_*$.
If the entries of z are independently distributed according to a Gumbel distribution with location 0 and scale 1, the probability density function reads

$$p(z) = e^{-\left(\sum_{i=1}^{d} z_i + e^{-z_i}\right)}$$

so that $\eta(z) = \sum_{i=1}^{d} z_i + e^{-z_i}$. Thus we have

$$\nabla\eta(z) = \begin{pmatrix} 1 - e^{-z_1} \\ 1 - e^{-z_2} \\ \vdots \\ 1 - e^{-z_d} \end{pmatrix}$$

and

$$\|\nabla\eta(z)\|^2 = \sum_{j=1}^{d}(1 - e^{-z_j})^2$$

Algorithm 9 and Algorithm 10 require a bound on the value of M. We compute it now.

$$\begin{aligned}
M^2 &= \mathbb{E}\,\|\nabla\eta(Z)\|^2 \\
&= \int_{\mathbb{R}^d}\sum_{j=1}^{d}(1 - e^{-z_j})^2 e^{-\left(\sum_{i=1}^{d} z_i + e^{-z_i}\right)} dz \\
&= d\int_{\mathbb{R}^d}(1 - e^{-z_1})^2 e^{-\left(\sum_{i=1}^{d} z_i + e^{-z_i}\right)} dz \\
&= d\int_{\mathbb{R}}(1 - e^{-z_1})^2 e^{-(z_1 + e^{-z_1})}\int_{\mathbb{R}} e^{-(z_2 + e^{-z_2})}\int_{\mathbb{R}}\cdots\int_{\mathbb{R}} e^{-(z_d + e^{-z_d})} dz_d \ldots dz_2 dz_1
\end{aligned}$$

For any $i \geq 2$,

$$\int_{\mathbb{R}} e^{-(z_i + e^{-z_i})} dz_i = 1.$$

Moreover, one can check that

$$\int (1 - e^{-z_1})^2 e^{-(z_1 + e^{-z_1})} dz_1 = e^{-e^{-z_1}} + e^{-2z_1 - e^{-z_1}} + C$$

where C is some constant. Taking limits one gets

$$\int_{\mathbb{R}} (1 - e^{-z_1})^2 e^{-(z_1 + e^{-z_1})} dz_1 = 1$$

and thus

$$M^2 = d.$$

The derivation in the case of a multivariate normal distribution is similar.

Chapter

3 Nonconvex optimization via Frank-Wolfe & Carathéodory

Chapter content

3.1 Introduction 45
3.1.1 Related work 45
3.1.2 Contributions 45
3.1.3 Chapter organization 46
3.2 Preliminaries 46
3.2.1 Dual and convex relaxation 47
3.2.2 Carathéodory 47
3.2.3 Shapley Folkman and duality gap estimation 52
3.3 Solving nonconvex separable problems 54
3.3.1 Obtaining primal points 54
3.3.2 Trimming the solution 57
3.3.3 Final reconstruction 59
3.4 The special case of mixed integer linear programs 65
3.5 Numerical results 66
3.5.1 Computing the conjugate 67
3.5.2 Linear minimization over the domain 68
3.5.3 Results 68
Appendix 3.A Additional proofs 70
3.A.1 Proof of Theorem 3.2.3 70
3.A.2 Proof of Theorem 3.2.7 70
3.A.3 Proof of Theorem 3.2.8 71
3.A.4 Proof of Theorem 3.2.9 71
3.A.5 Proof of Lemma 3.3.3 72
3.A.6 Proof of Theorem 3.3.3 73
3.A.7 Proof of Theorem 3.3.6 75
3.A.8 Proof of Theorem 3.3.7 76
Appendix 3.B Auxiliary lemma 78
Appendix 3.C Technical lemmas 79
Appendix 3.D Experimental details 81

3.1 Introduction

We consider nonconvex separable convex optimization problems of the form

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} \quad & \sum_{i=1}^{n} A_i x_i \leq b, \\ & x_i \in X_i, \ i = 1, \ldots, n, \end{aligned} \tag{P}$$

in the variables $x_i \in \mathbb{R}^{d_i}$, with $A_i \in \mathbb{R}^{m \times d_i}$, $b \in \mathbb{R}^m$. We make the following assumption on the structure of the functions considered.

Assumption 3.1.1. *For all $i \in \{1, \ldots, n\}$, f_i is proper, lower semicontinous and has an affine minorant. Moreover $X_i = \operatorname{dom} f_i$ is compact.*

Note that we do not assume convexity of either the functions f_i or their domains X_i. We also assume that one can efficiently compute the conjugates of the functions f_i and theirs gradients (or subgradients). We let $d = \sum_{i=1}^{n} d_i$ and $A = [A_1 \ldots A_n] \in \mathbb{R}^{m \times d}$ so that the constraints can be expressed in the more compact form $Ax \leq b$, where $x = [x_1^\top x_2^\top \ldots x_n^\top]^\top$.

3.1.1 Related work

The Shapley-Folkman theorem states that the Minkowski sum of arbitrary sets gets increasingly close to its convex hull as the number of terms in the sum increases. This was first established by Shapley and Folkman in private communications, and first published in [Sta69]. This result can be used to show that the duality gap of (P) vanishes when the number n of component functions goes to infinity, while the dimension of the constraints m remains bounded [AE76]. More precisely, the duality gap can be bounded by $m \max_i \rho(f_i)$, where $\rho(f_i)$ is a measure of the nonconvexity of the objective function f_i [AE76, BT16, UB16]. In particular, when the underlying domain is convex, [UB16] describe a randomized algorithm achieving those bounds with probability one. In cases where the domain of the component functions is not convex, the Shapley-Folkman theorem can also be used to prove similar bounds on the duality gap, this time based on more general measures of nonconvexity than the ones mentioned above [Ber14, VEG+16]. In particular, under additional structural assumptions on the primal domain, [VEGM14, VEG+16] describe algorithms to achieve those bounds in mixed-integer linear programming (MILP).

3.1.2 Contributions

The Shapley-Folkman theorem is a direct consequence of the Carathéodory theorem and its conic version. Our first contribution adapts the method in [Wir24], to provide an algorithm that computes the conic Carathédory representation of a vector given its conic decomposition.

We then describe a systematic approach to approximately solve problem (P), assuming that the dual or bi-dual has been solved. The first step is a conditional gradient method on a smooth convex objective to obtain a conic combination of extreme points which approximates the optimal dual value. This method converges sublinearly and, crucially here, we show that the linear minimization oracle comes down to computing the Fenchel conjugate of the functions f_i. The second step requires trimming the conic combination to a conic Carathéodory representation using only $n+m+1$ extreme points. This can be done using the previously derived algorithm to compute conic Carathéodory representations. For convex domains, this directly yields a primal feasible solution which attains the classical bound of [AE76, UB16], up to an arbitrarily small additive term. Alternatively, we show that trimming using conditional gradient methods, as

in [CP23], allows us to reduce the dependency on m in the duality gap bound, at the price of an exponentially decreasing error term, which depends on the number of iterations of the conditional gradient method.

In the case of nonconvex domains, we show that the obtained solution is not feasible on at most $m+1$ of the n component functions. Under various additional assumptions, we show that a final reconstruction step can be carried out to obtain fully feasible solutions, and we obtain the same approximation results as above, this time based on a more general notion of nonconvexity [Ber14, VEGM14, VEG+16]. Those results were derived in [VEGM14, VEG+16] for MILPs with unique solutions, but here we make no such assumption.

3.1.3 Chapter organization

The chapter is organized as follows. In Section 3.2, we give some preliminaries in convex optimization, and recall the Carathéodory theorem and its link with Shapley-Folkman. We also describe an algorithm to obtain exact Carathéodory representations, for which we provide a complexity bound. We also recall some known algorithms for computing approximate Carathéodory representations. We then highlight several known results from duality gap estimation based on Shapley-Folkman. In Section 3.3, we describe the core of our method to find approximate solutions to problem (P), which entails a first conditional gradient method, a Carathéodory algorithm on the obtained extreme points, and a final reconstruction step. We prove bounds on the duality gap, for both convex and general domains. In Section 3.4, we show that when dealing with MILPs, we can replace the convex objective function in the first conditional gradient method by a strongly convex quadratic, and hence obtain linear rates of convergence. This in turn improves the duality gap bound error term. Finally, in Section 3.5, we showcase the effectiveness of our method on a power system application.

3.2 Preliminaries

For $k \in \mathbb{N}$, Δ_k is the simplex of dimension $k-1$. For a set S, we write its closure $\operatorname{cl}(S)$, its convex hull $\operatorname{conv}(S)$ and the set of its extreme points $\operatorname{ext}(S)$. Throughout this chapter, we only consider the Euclidean norm, which we write $\|\cdot\|$ for simplicity. Moreover, for a vector $x \in \mathbb{R}^d$, we define $\|x\|_+ = \sqrt{\sum_{j=1}^d \max(x_j, 0)^2}$.

If f is a function not everywhere $+\infty$, with an affine minorant, the (Fenchel) conjugate of f is defined as

$$f^*(y) = \sup_{x \in \operatorname{dom} f} \{y^\top x - f(x)\}. \tag{3.1}$$

The biconjugate of f, defined as the conjugate of f^* and written f^{**}, is also the pointwise supremum of all affine functions minorizing f [HUL96, Theorem X.1.3.5]. In particular, we have $\operatorname{epi} f^{**} = \operatorname{cl} \operatorname{conv} \operatorname{epi} f$. Moreover, when f is lower semicontinuous and 1-coercive, $\operatorname{conv} \operatorname{epi} f$ is closed and hence $\operatorname{epi} f^{**} = \operatorname{conv} \operatorname{epi} f$ [HUL96, Theorem X.1.5.3].

The nonconvexity of a function $f : \mathbb{R}^d \to \mathbb{R}$ is defined as [AE76]

$$\rho(f) = \sup \left\{ f\left(\sum_{j=1}^p \alpha_j x_j\right) - \sum_{j=1}^p \alpha_j f(x_j) \mid p \in \mathbb{N}, x_j \in \operatorname{dom} f, \alpha \in \Delta_p \right\}. \tag{3.2}$$

Clearly $\rho(f) = 0$ if f is convex. Under Assumption 3.1.1, the nonconvexity $\rho(f)$ can also be expressed in terms of the biconjugate of f [BT16, Proposition 6],

$$\rho(f) = \sup_{x \in \operatorname{dom} f} \{f(x) - f^{**}(x)\}.$$

3.2.1 Dual and convex relaxation

Following [KCd17] we define, for $\lambda \in \mathbb{R}^m$,

$$\Psi(\lambda) = \inf_{x_i \in X_i} \left\{ \sum_{i=1}^{n} f_i(x_i) + \lambda^\top (\sum_{i=1}^{n} A_i x_i - b) \right\} \tag{3.3}$$

The Lagrange dual of (P) is given by

$$\begin{aligned} &\text{maximize } \Psi(\lambda) \\ &\text{subject to } \lambda \geq 0. \end{aligned} \tag{D}$$

The bidual of (P), which is the dual of (D), is then given by

$$\begin{aligned} \text{minimize } & \sum_{i=1}^{n} f_i^{**}(x_i) \\ \text{subject to } & Ax \leq b, \\ & x_i \in \operatorname{conv}(X_i), \quad i = 1, \ldots, n. \end{aligned} \tag{CoP}$$

The fact that the domain is now $\operatorname{conv}(X_i)$ follows from $\operatorname{dom} f_i^{**} = \operatorname{conv}(\operatorname{dom} f_i)$. From weak duality, the optimal value of (D) is a lower bound on the optimal value of (P). The following theorem from [LR01] gives a sufficient condition for the optimal values of (D) and (CoP) to match.

Theorem 3.2.1. *Assume Ψ is not everywhere $-\infty$ and there exists a feasible $\hat{x}$ in the interior of $\operatorname{dom} \sum_{i=1}^{n} f_i^{**}$. Then strong duality holds, i.e. the optimal values of (D) and (CoP) match.*

From now on, we assume that strong duality holds and let

$$v^* := \max_{\lambda \geq 0} \Psi(\lambda). \tag{3.4}$$

In the rest of the chapter, we assume that one can efficiently obtain v^*, by either solving (D) or (CoP). This is a relatively weak assumption, as both (D) and (CoP) are convex problems, for which standard convex optimization solvers can be used.

3.2.2 Carathéodory

Carathéodory's theorem and its conic version are core ingredients in proving the Shapley-Folkam theorem and the resulting duality gap bounds. We recall both results below, and also present the proof of the conic version, as it will be useful later in the chapter. The classical Carathéodory theorem states that a vector in the convex hull of a set in $\mathbb{R}^p$ can be written as a convex combination of most $p+1$ points of the original set.

Theorem 3.2.2 (Carathéodory)**.** *Let $S \subset \mathbb{R}^p$ and $w \in \operatorname{conv} S$. Then there exists $\{\lambda_j\}_{j=1}^{p+1} \subset \mathbb{R}_+$, $\{w_j\}_{j=1}^{p+1} \subset S$ with $\sum_{j=1}^{p+1} \lambda_j = 1$, such that $x = \sum_{j=1}^{p+1} \lambda_j w_j$.*

Proof. See [Roc15, Theorem 17.1]. □

Let $\operatorname{Po}(S) = \left\{ \sum_j \lambda_j v_j \mid \lambda_j \geq 0, v_j \in S \right\}$ be the conic hull of S. The conic Carathéodory theorem states that any point in the conic hull can be written as a conic combination of at most p points in the original set.

Theorem 3.2.3 (Conic Carathéodory)**.** *Let $S \subset \mathbb{R}^p$ and $w \in \operatorname{Po} S$. Then there exists $\{\lambda_j\}_{j=1}^{p} \subset \mathbb{R}_+$, $\{w_j\}_{j=1}^{p} \subset S$, such that $x = \sum_{j=1}^{p} \lambda_j w_j$.*

Proof. See Section 3.A. □

The next two subsections are dedicated to computing Carathéodory representations. We consider a vector $w^* \in \mathbb{R}^p$, and a subset $S \subset \mathbb{R}^p$ of size N which we write $S = \{w_1, \dots, w_N\}$. We will either assume $w^* \in \mathrm{Po}(S)$ or $w^* \in \mathrm{conv}(S)$. In any case, for coefficients $\lambda_1, \dots, \lambda_N > 0$, we have

$$w^* = \sum_{j=1}^{N} \lambda_j w_j. \tag{3.5}$$

3.2.2.1 Computing exact Carathéodory representations

In this section we assume that $w^* \in \mathrm{Po}(S)$, and that the vectors $w_1, \dots, w_N$ and the coefficients $\lambda_1, \dots, \lambda_N$ are known. The proof of Theorem 3.2.3 provides an algorithm to reduce the number of nonzero coefficients in the expression of w^* given in (3.5) from N to the dimension p. We summarize the method in Algorithm 11. While the proof of Theorem 3.2.3 suggests solving linear systems of dimension $\mathbb{R}^{p\times N}$, we show that it is sufficient to solve them in dimension $\mathbb{R}^{(p+1)\times(p+1)}$, where the additional row is necessary to obtain non-trivial solutions. For a matrix W, we write $W_{:,k}$ as its k-th column and $W_{k,:}$ as its k-th row. Moreover, $W_{:,k_1:k_2}$ is the submatrix of W composed of the $k_2 - k_1 + 1$ columns between k_1 and k_2.

Algorithm 11 Exact Carathéodory

Input: $w_1, \dots, w_N$ and nonnegative coefficients $\lambda_1, \dots, \lambda_N$ such that $w^* = \sum_{j=1}^N \lambda_j w_j$.
Build $W \in \mathbb{R}^{p\times N}$, the matrix whose columns are the vectors w_j.
Draw vector $u \in \mathbb{R}^N$ with uniform distribution on the unit sphere and let $\tilde{W}_u = \begin{pmatrix} W \\ u^\top \end{pmatrix}$.
$\tilde{W}_u^s \leftarrow (\tilde{W}_u)_{:,p+1}$
$\alpha \leftarrow \lambda_{1:p+1}$
$c \leftarrow p + 2$
while $c \leq N + 1$ **do**
  Solve $(\tilde{W}_u^s)\delta = \begin{pmatrix} \mathbf{0}_p \\ 1 \end{pmatrix}$
  $t^* \leftarrow \min\{\frac{\alpha_j}{\delta_j} \mid \delta_j > 0\}$
  $\alpha' \leftarrow \alpha - t^*\delta$
  $j^* \leftarrow \arg\min_j \alpha'_j$
  Remove element at index j^* from α' and column j^* from $\tilde{W}_u^s$
  if $c = N + 1$ **then**
    $\alpha \leftarrow \alpha'$
    Break
  else
    $\alpha \leftarrow (\alpha', \lambda_c)$
    $\tilde{W}_u^s \leftarrow \left(\tilde{W}_u^s, (\tilde{W}_u)_{:,c}\right)$
    $c \leftarrow c + 1$
  end if
end while
Return $\alpha, (\tilde{W}_u^s)_{1:p,:}$.

Theorem 3.2.4. *With probability 1, Algorithm 11 outputs $\alpha \in \mathbb{R}_+^p$ and p vectors $\{w_{j_l}\}_{l=1}^p \subset S$,*

such that

$$w^* = \sum_{l=1}^{p} \alpha_l w_{j_l}. \tag{3.6}$$

Proof. Our proof basically mirrors the one of Theorem 3.2.3. By the definition of W in Algorithm 11, we have $W\lambda = w^*$. Let W^s be the matrix $\tilde{W}_u^s$ to which we have removed the last row (namely the row composed of elements of the random vector u). At the beginning of the first iteration, we then have

$$\begin{aligned} W\lambda &= W^s\alpha + (W_{:,c:N})\lambda_{c:N}, \\ \Rightarrow W^s\alpha &= w^* - (W_{:,c:nK})\lambda_{c:N}. \end{aligned} \tag{3.7}$$

We claim that (3.7) is true at the beginning of every iteration. We saw that it was true at the beginning of the first iteration, now suppose it is true at the beginning of some iteration.

Since $W^s \in \mathbb{R}^{p\times(p+1)}$, the system $W^s\delta = \mathbf{0}_p$ must have a non-trivial solution. Let δ^* be one such solution. Observe that

$$\tilde{W}_u^s \frac{\delta^*}{((\tilde{W}_u^s)_{p+1,:})^\top \delta^*} = \begin{pmatrix} \mathbf{0}_p \\ 1 \end{pmatrix},$$

and hence a solution of the system on line 8 exists as long as $((\tilde{W}_u^s)_{p+1,:})^\top \delta^* \neq 0$. Since the last row $\tilde{W}_u^s$ has uniform distribution on the unit sphere, this holds with probability 1.

One can then check that lines 9-11 in the algorithm are simple computations to find $\alpha' \geq 0$ such that $W^s\alpha' = W^s\alpha$ and $\alpha'_{j^*} = 0$. Removing the index j^* from α' and the column j^* from W^s, we get $\alpha' \in \mathbb{R}^p$, $W^s \in \mathbb{R}^{p\times p}$ and

$$W^s\alpha' = w^* - (W_{:,c:N})\lambda_{c:N}.$$

At this point two cases arise.

Case 1: If $c = N+1$, it must be that $W^s\alpha' = w^*$, and we have therefore found a conic combination of w^* with p coefficients, so we are done.

Case 2: If $c < N+1$, we can append λ_c to α' and column c of $\tilde{W}$ to $\tilde{W}^s$, as on lines 17 and 18. This results in $\alpha \in \mathbb{R}^{p+1}$ and $W^s \in \mathbb{R}^{p\times(p+1)}$ such that

$$W^s\alpha = w^* - (W_{:,c+1:N})\lambda_{c+1:N}.$$

Updating $c \leftarrow c+1$ as on line 19, and starting a new iteration, we are back to the equation (3.7), except that we have reduced the column dimension of the matrix $W_{:,c:N}$ by 1. We can iterate this argument until case 1 above is true. □

Note that similar algorithms to obtain Carathéodory representations have been developed in [BS17] and in [Wir24]. Those approaches are however aimed at reducing the size of the active set of conditional gradient methods, and hence are looking for convex representations via the classic Carathéodory result (Theorem 3.2.2). In contrast, we are looking for representations in the conic hull of S via the proof of Theorem 3.2.3. Our approach is closer to the one of [Wir24], in the sense that we make no assumption on how the linear system is solved on line 8. In particular, when the vectors in S are sparse, efficient direct or indirect sparse solvers may be used. In contrast, the approach in [BS17] requires maintaining a matrix in row-echelon form which, depending on the application, might not be computationnaly efficient. Without further assumption on the structure of the linear systems, we suggest a strategy to achieve complexity $O(p^2)$ at each iteration, which is similar to the complexity achieved in [BS17] and in [Wir24].

Indeed, at each iteration, the linear system to be solved is the same as the one at the previous iteration, up to one column deletion and one column insertion. On can therefore compute the QR decomposition of $\tilde{W}_u^s$ on line 4, solve the linear systems on line 8 using the decomposition, and update the QR decomposition by either deleting a column (line 12) or by appending a column (line 18). We give the complexity of this strategy in the next lemma.

Lemma 3.2.1. *Using the above strategy, the complexity of Algorithm 11 is $O(Np^2)$.*

Proof. The complexity of computing the first QR decomposition is $O(p^3)$. The complexity of solving the linear system given the QR decomposition is $O(p^2)$, while the complexity of deleting/appending a column is also $O(p^2)$ [GVL13, Section 6.5.2]. We must do the above two tasks at each iteration, and there are $N - p$ iterations in total. Since $N \geq p$, the complexity is indeed $O(Np^2)$. □

3.2.2.2 Computing approximate Carathéodory representations

We assume now that w^* is in the convex hull of S, and in contrast with the previous section, we do not require knowledge of the vectors $w_1, \ldots, w_N$, nor of the coefficients $\lambda_1, \ldots, \lambda_N$. Since we assumed $\lambda > 0$, we actually have w^* in the relative interior of $\mathrm{conv}(S)$ [HUL96, Remark III.2.1.4].

We aim to extract as few vectors as possible in S whose convex combination is a good approximation of w^*. It is then natural to apply Frank-Wolfe like algorithms on the following problem

$$\min_{w \in \mathrm{conv}(S)} \left\{ h(w) := \frac{1}{2} \|w - w^*\|^2 \right\}. \tag{3.8}$$

The above objective function h is strongly convex, and the feasible set is a convex polyhedra. Following [CP23], we suggest using the fully-corrective Frank-Wolfe (FCFW) algorithm. While other conditional gradient mehods like away-step Frank-Wolfe [Wol70, GM86] can be used, FCFW tends to yield the best numerical results, and has the best per-iteration improvement among all conditional gradient methods [CP23]. We recall the algorithm here for completeness.

The method starts with some $w^0 \in S$, sets $s^0 = w^0$ and defines an active set $\mathcal{S}^0 = \{s^0\}$. At each iteration $t \geq 1$, given some w^{t-1} and an active set $\mathcal{S}^{t-1}$, the method makes a call to the linear minimization oracle

$$s^t \in \operatorname*{arg\,min}_{s \in \mathrm{conv}(S)} s^\top \nabla h(w^{t-1}). \tag{3.9}$$

Since this is a linear objective, the minimizer is attained at an extreme point of $\mathrm{conv}(S)$, and hence the search space can be reduced to S. Finally, instead of computing a convex combination of s^t and w^{t-1} as in the classical Frank-Wolfe algorithm, the method optimizes over the convex hull of the new active set, namely

$$\begin{aligned} \mathcal{S}^t &= \mathcal{S}^{t-1} \cup \{s^t\} \\ w^t &= \operatorname*{arg\,min}_{w \in \mathrm{conv}\, \mathcal{S}^t} h(w) \end{aligned} \tag{3.10}$$

Depending on the structure of the objective function, step (3.10) can be intractable or too costly to compute, which limits the use of FCFW in some practical applications. However, since h is a quadratic here, (3.10) can be rewritten as

$$\operatorname*{arg\,min}_{\alpha \in \Delta_t} \frac{1}{2} \left\| \sum_{l=0}^{t} \alpha_l s^l - w^* \right\|^2.$$

This a quadratic programming (QP) problem, which can be solved using classical off-the-shelf QP solvers [SBG+20, ApS19, FKP+14, BSEK+23], or via first-order methods like gradient descent, since projection on the simplex can be efficiently computed. We summarize the full method in Algorithm 12.

Algorithm 12 Approximate Carathéodory via fully corrective Frank Wolfe

Input: $w^* \in \mathbb{R}^p$, set $S = \{w_1, \dots, w_N\}$, number of iterations T.
Pick $j \in \{1, \dots, N\}$ and set $w^0 = w_j$, $s^0 = w^0$ and $\mathcal{S}^0 = \{s^0\}$.
for $t = 1, \dots, T$ **do**
    Find s^t such that

$$s^t \in \arg\min_{w \in S} w^\top (w^{t-1} - w^*).$$

    Set $\mathcal{S}^t = \mathcal{S}^{t-1} \cup \{s^t\}$.
    Build $S_t \in \mathbb{R}^{p \times (t+1)}$, the matrix whose columns are the vectors in $\mathcal{S}^t$.
    Solve the following QP:

$$\alpha^t \in \arg\min_{\alpha \geq 0, \mathbf{1}^\top \alpha = 1} \frac{1}{2} \alpha^\top S_t^\top S_t \alpha - (w^*)^\top S_t \alpha$$

    $w^t = \sum_{l=0}^t \alpha_l^t s^l$.
end for
return w^T

After running Algorithm 12 for T iterations, the output is a convex combination of points in S, i.e.

$$w^T = \sum_{t=0}^{T} \alpha_t w_{j_t},$$

with $\alpha_t \geq 0$, $\sum_{t=0}^T \alpha_t = 1$, and $w_{j_t} \in S$ for all t. We now give convergence guarantees on the distance between w^T and w^* for Algorithm 12. Linear convergence rates for conditional gradient methods usually depend either on the distance from the minimizer to the relative boundary of the feasible set [GM86, GH15], or on constants quantifying the conditioning of the feasible set, such as the *pyramidal width* [LJJ15], the *vertex-facet distance* [BS17], or the *restricted width* [PRS16]. Those constants are notoriously difficult to apprehend, let alone compute in practical settings.

We give two convergence results, one based on the distance r between w^* and the relative boundary of $\operatorname{conv} S$ [GH15], and the second one based on the notion of *facial distance* of the feasible set [PR19], which turns out to be equivalent to the pyramidal width defined in [LJJ15].

Definition 3.2.1. *The facial distance of S is defined as*

$$\omega_S := \min_{\substack{F \in faces(\operatorname{conv} S), \\ \emptyset \subsetneq F \subsetneq \operatorname{conv} S}} dist(F, \operatorname{conv}(S \setminus F)) \tag{3.11}$$

We summarize the two convergence results in the next theorem.

Theorem 3.2.5. *Let w^T be the output of Algorithm 12 after T iterations. Then*

$$\left\| w^T - w^* \right\|^2 \leq \left\| w^0 - w^* \right\|^2 \left(1 - \frac{\delta(S, w^*)}{4 D_S^2} \right)^T, \tag{3.12}$$

where D_S is the diameter of $\operatorname{conv}(S)$ and

$$\delta(S, w^*) := \max \left(\omega_S^2, (r^*)^2 \right),$$

where r^* is the radius of the largest relative ball centered at w^* contained in $\operatorname{conv}(S)$, and ω_S is the facial distance of S defined in (3.11).

Proof. Observe that the objective function h is 1-strongly convex and its gradient is 1-Lipschitz, and its optimal value is 0. The rate with respect to the facial distance is then a consequence of [LJJ15, Theorem 1] and the fact that the facial distance is equivalent to the pyramidal width [PR19, Theorem 2]. The rate with respect to the distance to the relative boundary is a consequence of [GH15, Section 4.2], and was also derived in [CP23]. □

3.2.3 Shapley Folkman and duality gap estimation

Both versions of Carathéodory's theorem are central to the proof of the Shapley Folkman theorem, which states that the Minkowski sum of arbitrary sets gets increasingly closer to its convex hull as the number of terms grows. Established in private communications between Shapley and Folkman, it was first published in [Sta69].

Theorem 3.2.6 (Shapley Folkman)**.** *Let $V_i \subset \mathbb{R}^d$, $i = 1, \dots, n$ and $x \in \sum_{i=1}^n \operatorname{conv}(V_i)$. Then for some set $S \subset \{1, \dots, n\}$ with $|S| \leq d$, we have*

$$x \in \sum_{i \in \{1,\dots,n\} \setminus S} V_i + \sum_{i \in S} \operatorname{conv}(V_i)$$

Proof. Since $x \in \sum_{i=1}^n \operatorname{conv}(V_i)$, by classic Carathédory we have

$$x = \sum_{i=1}^{n} \sum_{j=1}^{d+1} \lambda_{ij} v_{ij}$$

with $v_{ij} \in V_i$ for all i, and $\sum_{j=1}^{d+1} \lambda_{ij} = 1$ for all i. We can then write

$$\begin{pmatrix} x \\ \mathbf{1}_n \end{pmatrix} = \sum_{i=1}^{n} \sum_{j=1}^{d+1} \lambda_{ij} \begin{pmatrix} v_{ij} \\ e_i \end{pmatrix}.$$

This is a conic combination in $\mathbb{R}^{n+d}$, so by conic Carathéodory, there exists $\mu \in \mathbb{R}_+^{n(d+1)}$ with at most $n + d$ nonzero such that

$$\begin{pmatrix} x \\ \mathbf{1}_n \end{pmatrix} = \sum_{i=1}^{n} \sum_{j=1}^{d+1} \mu_{ij} \begin{pmatrix} v_{ij} \\ e_i \end{pmatrix}.$$

In particular, for each i, there must be at least one of the μ_{ij} not equal to 0. Since there are only $n + d$ nonzero coefficients, this implies that for at least $n - d$ indices, the convex combination is trivial, i.e. has only one nonzero coefficient, which is equal to one. For the remaining d indices, the convex combination is non-trivial. In other words, for a set S with $|S| \leq d$,

$$x \in \sum_{i \notin S} V_i + \sum_{i \in S} \operatorname{conv}(V_i).$$

□

3.2.3.1 Duality gap estimation: convex domain, nonconvex objectives

When dealing with convex domains X_i, we have $X_i = \operatorname{conv}(X_i)$. We now review some results covering duality gap estimation. We first recall the original result of [AE76] (see also [ET99]), giving a bound on the duality gap which depends on the dimension of the constraints m and the nonconvexity of the considered functions, but which is independent of the number of functions n.

Theorem 3.2.7. *Let v^* be the dual optimal value defined in (3.4) and $\rho(f_i)$ be as defined in (3.2). Assume that X_i is convex for all i. There exists $x^* \in \mathbb{R}^d$ such that $Ax^* \leq d$ and*

$$v^* \leq \sum_{i=1}^{n} f_i(x_i^*) \leq v^* + (m+1) \max_i \rho(f_i).$$

Proof. See Section 3.A.2. □

The above theorem is an existence result, and the proof does not provide a way to construct a solution satisfying the bound.

In [UB16], the authors provide an algorithm achieving the bound of Theorem 3.2.7 with probability 1, by solving a randomized version of the convex relaxation (CoP). Although they claim otherwise, their analysis implicitly assumes that the function domains X_i are convex. Indeed, their randomized approach yields a solution $x_i \in \operatorname{conv} X_i$, $i = 1, \ldots, n$. On a certain index set $\mathcal{J}$ of cardinality at least $n - m$, they show that $i \in \mathcal{J}$ implies that x_i is an extreme point of $\operatorname{conv} X_i$. This in turn implies that $x_i \in X_i$ and $f_i(x_i) = f_i^{**}(x_i)$. On non-extreme points, i.e. for $i \notin \mathcal{J}$, the authors bound $f_i(x_i) - f_i^{**}(x_i)$ by $\rho(f_i)$. However, this implicitly assumes X_i is convex, since otherwise $\rho(f_i) = \infty$ and the bound is meaningless. When X_i is nonconvex, an additional step is required to 'primalize' the solutions for $i \notin \mathcal{J}$. We explore such primalization step in Section 3.3.3. Moreover, the approach described in [UB16] requires optimizing over the solution set of the convex relaxation (CoP), which in some cases might be a difficult task. In contrast, our approach only assumes access to the value of v^*, which can be obtained by either solving the dual (D) or the convex relaxation (CoP), whichever one is easier to optimize.

In [KCd17], the authors describe an approach to obtain approximate Carathéodory representations with high sampling ratios, which they use to prove approximate Shapley-Folkman theorems. Those in turn can be used to prove duality gap bounds similar to the one in Theorem 3.2.7, except that they reduce the dependency on $m + 1$, at the price of an additional term due to the approximation error. Their approach is however not constructive, and the duality gap bounds provided are on a perturbed version of the primal problem (P).

3.2.3.2 Duality gap estimation: general domains

When dealing with nonconvex domains, we have $\operatorname{dom} f_i^{**} = \operatorname{conv} X_i \neq X_i$. In particular, $\rho(f_i) = \infty$, and thus a more general notion of nonconvexity is needed. As in [Ber14], let us define

$$\gamma(f_i) := \sup_{x \in X_i} f_i(x) - \inf_{y \in X_i} f_i(y). \tag{3.13}$$

As in Theorem 3.2.7, the duality gap is bounded by a term depending on the number of constraints m and on $\max_i \gamma(f_i)$, but not on the number of functions n [AE76, Ber14].

Theorem 3.2.8. *Let v^* be the dual optimal value defined in (3.4) and $\gamma(f_i)$ be defined in (3.13). Suppose that there exists $\hat{x}_i \in X_i$ such that $A_i \hat{x}_i \leq A_i x_i$ for all $x_i \in \operatorname{conv} X_i$. There exists $x^* \in \mathbb{R}^d$ such that $x_i^* \in X_i$ for all i, $Ax^* \leq d$ and*

$$v^* \leq \sum_{i=1}^{n} f_i(x_i^*) \leq v^* + (m+1) \max_i \gamma(f_i).$$

Proof. See Section 3.A.3. □

As for Theorem 3.2.7, this theorem is an existence result, and its proof does not yield a constructive algorithm. For MILPs whose bidual (CoP) has a unique solution, [VEGM14] design an algorithm satisfying the bound of Theorem 3.2.8.

The assumption on the existence of $\hat{x}_i$ in Theorem 3.2.8 can be quite restrictive, although there exist practical cases for which it is satisfied [VEGM14]. It is true for example when $A_i x_i \geq 0$ and $0 \in X_i$ for all i. To lift this restriction, one can solve the following perturbed problem

$$\begin{aligned} \min_{x_i \in \operatorname{conv} X_i} \ & \sum_{i=1}^{n} f_i^{**}(x_i) \\ \text{subject to} \ & Ax \leq b - \theta. \end{aligned} \tag{CoP$_\theta$}$$

The following theorem states that for a well chosen value of θ, there exists a solution of (CoP$_\theta$) whose duality gap can be bounded by the sum of two terms, one depending on the number of constraint m, and the second one characterizing the feasibility of (CoP$_\theta$) and the perturbation parameter θ. This result was proven for MILPs in [VEG+16], and here we extend it to general nonconvex problems. Recall that $(A_i)_{k,:}$ corresponds to the k-th row of the matrix A_i.

Theorem 3.2.9. *Let v^* be the dual optimal value as defined in (3.4) and $\gamma(f_i)$ be as defined in (3.13). Let $\theta \in \mathbb{R}^m$ be defined as*

$$\theta_k = (m+1) \max_i \left(\max_{x_i \in X_i} ((A_i)_{k,:})^\top x_i - \min_{x_i \in X_i} ((A_i)_{k,:})^\top x_i \right),$$

and suppose that there exists $\xi > 0$ and $\hat{x}_i \in \operatorname{conv} X_i$ for all i such that

$$\sum_{i=1}^{n} A_i \hat{x}_i \leq b - \theta - n\xi \mathbf{1}_m.$$

Then, there exists a solution $x^* \in \mathbb{R}^d$, such that $x_i^* \in X_i$, $Ax^* \leq b$, and

$$v^* \leq \sum_{i=1}^{n} f_i(x_i^*) \leq v^* + \left((m+1) + \frac{\|\theta\|_\infty}{\xi} \right) \max_i \gamma(f_i).$$

Proof. See Section 3.A.4. □

3.3 Solving nonconvex separable problems

We now come back to our original task of finding approximate solutions to problem (P). Recall that v^* is the optimal value of the dual (D).

3.3.1 Obtaining primal points

Let us define $z^* \in \mathbb{R}^{1+m}$ as

$$z^* = \begin{pmatrix} v^* \\ b \end{pmatrix}. \tag{3.14}$$

Since v^* is the optimal value of (CoP), there exists $x^* \in \mathbb{R}^d$ such that $Ax^* \leq b$ and $v^* = \sum_{i=1}^n f_i^{**}(x_i^*)$. In other words,

$$z^* \in \sum_{i=1}^{n} \left\{ \begin{pmatrix} f_i^{**}(x_i) \\ A_i x_i \end{pmatrix} \mid x_i \in \operatorname{conv} X_i \right\} + \mathbb{R}_+^{m+1}. \tag{3.15}$$

Let us define the corresponding sets

$$\mathcal{C}_i = \left\{ \begin{pmatrix} f_i^{**}(x_i) \\ A_i x_i \end{pmatrix} \mid x_i \in \operatorname{conv} X_i \right\}, \quad i = 1, \dots, n,$$
$$\mathcal{C} = \sum_{i=1}^{n} \mathcal{C}_i.$$

We aim to approximate z^* with elements of $\mathcal{C}_i$ where f_i and f_i^{**} match. The next lemma, proved in Section 3.C, states that this is the case at extreme points of $\mathcal{C}_i$.

Lemma 3.3.1. *Let $\tilde{x}_i \in \operatorname{conv} X_i$ such that $(f_i^{**}(\tilde{x}_i), A_i\tilde{x}_i)$ is an extreme point of $\mathcal{C}_i$. Then $\tilde{x}_i \in X_i$ and $f_i(\tilde{x}_i) = f_i^{**}(\tilde{x}_i)$.*

To extract extreme points of $\mathcal{C}_i$, conditional gradient methods are strong candidates. At first sight, a natural objective function is the quadratic loss, giving rise to the following optimization problem

$$\min_z \frac{1}{2} \|z - z^*\|^2 \quad \text{subject to } z \in \mathcal{C} + \mathbb{R}_+^{m+1}. \tag{3.16}$$

However, linear minimization on the unbounded set $\mathcal{C} + \mathbb{R}_+^{m+1}$, which is required for conditional gradient methods, is not achievable. Alternatively, one can solve

$$\min_z \frac{1}{2} \|z - z^*\|_+^2 \quad \text{subject to } z \in \mathcal{C}. \tag{3.17}$$

Lemma 3.3.2. *Problems (3.16) and (3.17) are equivalent.*

Proof. For any $c \in \mathcal{C}$ and $\delta \in \mathbb{R}_+^{m+1}$, we have

$$\|c - z^*\|_+^2 = \sum_{j=1}^{m+1} \max(c_j - z_j^*, 0)^2 \leq \sum_{j=1}^{m+1} \max(c_j + \delta_j - z_j^*, 0)^2 \leq \|c + \delta - z^*\|^2 .$$

Moreover, for $\delta \in \mathbb{R}_+^{m+1}$ defined as $\delta_j = \max\left(0, z_j^* - c_j\right)$, we have

$$\begin{aligned} \|c + \delta - z^*\| &= \sum_{j=1}^{m+1} \left(c_j - z_j^* + \max\left(0, z_j^* - c_j\right)\right)^2 \\ &= \sum_{j=1}^{m+1} \max\left(c_j - z_j^*, 0\right)^2 = \|c - z^*\|_+^2 . \end{aligned}$$

In particular, the above shows that for any $c \in \mathcal{C}$,

$$\|c - z^*\|_+^2 = \inf_{\delta \in \mathbb{R}_+^{m+1}} \|c + \delta - z^*\|^2 .$$

Hence

$$\inf_{c \in \mathcal{C}} \left\{ \|c - z^*\|_+^2 \right\} = \inf_{c \in \mathcal{C}} \inf_{\delta \in \mathbb{R}_+^{m+1}} \left\{ \|c + \delta - z^*\|^2 \right\} = \inf_{z \in \mathcal{C} + \mathbb{R}_+^{m+1}} \left\{ \|z - z^*\|^2 \right\}.$$

□

The next lemma, proved in Section 3.A.5, shows that any solution of (3.17) has optimal value 0 and yields a solution of (CoP).

Lemma 3.3.3. *Suppose* $\tilde{z} = \begin{pmatrix} \sum_{i=1}^n f_i^{**}(\tilde{x}_i) \\ A\tilde{x} \end{pmatrix}$ *is a solution of* (3.17). *Then* $\|\tilde{z} - z^*\|_+ = 0$ *and* $\sum_{i=1}^n f_i^{**}(\tilde{x}_i) = v^*$ *and* $A\tilde{x} \leq b$.

We use the classical Frank Wolfe algorithm to solve (3.17). The algorithm starts from some guess $z_0 \in \mathcal{C}$. At each iteration k, the gradient of the objective function with respect to z^k is given by $(\alpha_k, g^k) \in \mathbb{R} \times \mathbb{R}^m$ with $\alpha_k = \max(z_1^k - v^*, 0)$ and $g^k = \max(z_{2:m+1}^k - b, 0)$. The linear minimization steps then reads

$$s^k \in \operatorname*{arg\,min}_{z \in \sum_{i=1}^n \mathcal{C}_i} z^\top (\alpha_k, g^k). \tag{3.18}$$

This amounts to solving, for each $i \in \{1, \ldots, n\}$,

$$y_i^k \in \operatorname*{arg\,min}_{x_i \in \operatorname{conv} X_i} \alpha_k f_i^{**}(x_i) + (g^k)^\top A_i x_i, \tag{3.19}$$

and setting $s^k = \sum_{i=1}^n y_i^k$. If $\alpha_k = 0$, the above is a linear minimization over $\operatorname{conv} X_i$. Otherwise, $\alpha_k > 0$, and we can can rewrite (3.19) as

$$y_i^k \in \operatorname*{arg\,max}_{x_i \in \operatorname{conv} X_i} \left(-\frac{A_i^\top g^k}{\alpha_k} \right)^\top x_i - f_i^{**}(x_i) = \partial f_i^* \left(-\frac{A_i^\top g^k}{\alpha_k} \right).$$

The last few lines are central to our approach. They tell us that the key subproblem in conditional gradient methods, namely the linear minimization oracle (3.18), is as tractable as computing the conjugate of the functions f_i.

We summarize the full algorithm in Algorithm 13. Note that while there always exists an extreme point of $\mathcal{C}_i$ which solves (3.19), a non-extreme point might yield the same objective value. Without loss of generality, we assume that we always get an extreme point, and hence $y_i^k \in X_i$ and $f_i(y_i^k) = f_i^{**}(y_i^k)$. In the application we consider in section Section 3.5, this is never an issue.

At any iteration k of the algorithm, the suboptimality gap $\left\|z^k - z^*\right\|_+$ can be used as stopping criterion. Alternatively, the quality of the approximation can be bounded using classical convergence results on the Frank-Wolfe method.

Theorem 3.3.1. *Suppose we run Algorithm 13 for* K *iterations with* $\eta_k = \frac{2}{k+2}$. *Then the output satisfies*

$$\left\|z^K - z^*\right\|_+^2 \leq \frac{4D_{\mathcal{C}}^2}{K+1},$$

where $D_{\mathcal{C}} = diam(\mathcal{C}, \|\cdot\|) = \max_{z,z' \in \mathcal{C}} \|z - z'\|$.

Proof. This is a direct consequence of the analysis of Frank-Wolfe algorithm, see for example [Jag11, Theorem 2.3]. Note that the feasible set $\mathcal{C}$ we are considering is not necessarily convex, and hence it might be that $z^K \notin \mathcal{C}$. This is however not an issue for the presented bound to hold. □

After running Algorithm 13 for K iterations, the output z^K is a convex combination of the vectors $\{s_k\}_{k=0}^{K-1}$. In other words, for $\gamma_k \in \Delta_K$, we have

$$z^K = \sum_{k=0}^{K-1} \gamma_k s_k = \sum_{k=0}^{K-1} \gamma_k \sum_{i=1}^n \begin{pmatrix} f_i(y_i^k) \\ A_i y_i^k \end{pmatrix}. \tag{3.20}$$

Algorithm 13 Frank Wolfe algorithm to obtain primal points

Input: $z^* \in \mathbb{R}^{1+m}$, number of iterations K.
For all $i = 1, \dots, n$, choose $y_i^0 \in X_i$ such that $f_i(y_i^0) = f_i^{**}(y_i^0)$.
Set $z^0 = \sum_{i=1}^n \begin{pmatrix} f_i(y_i^0) \\ A_i y_i^0 \end{pmatrix}$
for $k = 0, \dots, K-1$ **do**
  Set $\alpha_k = \max(z_1^k - v^*, 0)$ and $g^k = \max(z_{2:m+1}^k - b, 0)$.
  for $i = 1, \dots, n$ **do**
    if $\alpha_k = 0$ then

$$y_i^k \in \underset{x_i \in \operatorname{conv} X_i}{\arg\min} (A_i^\top g^k)^\top x_i$$

    else

$$y_i^k \in \partial f_i^* \left(-\frac{A_i^\top g^k}{\alpha_k} \right)$$

  end for
  Set $s^k = \sum_{i=1}^n \begin{pmatrix} f_i(y_i^k) \\ A_i y_i^k \end{pmatrix}$.
  Pick some stepsize $\eta_k > 0$.
  $z^{k+1} = (1 - \eta_k) z_k + \eta_k s_k$.
end for
return z^K

3.3.2 Trimming the solution

Let us define $w^K \in \mathbb{R}^{1+m+n}$ and $\mathcal{A}_K \subset \mathbb{R}^{1+m+n}$ as

$$w^K = \begin{pmatrix} z^K \\ \mathbf{1}_n \end{pmatrix}, \quad \text{and} \quad \mathcal{A}_K = \left\{ \begin{pmatrix} f_i(y_i^k) \\ A_i y_i^k \\ e_i \end{pmatrix} \mid k = 0, \dots, K-1, i = 1, \dots n \right\}. \tag{3.21}$$

From Equation (3.20), we get that

$$w^K = \sum_{k=0}^{K-1} \gamma_k \sum_{i=1}^n \begin{pmatrix} f_i(y_i^k) \\ A_i y_i^k \\ e_i \end{pmatrix}, \tag{3.22}$$

where $\{e_i\}_{i=1}^n$ is the canonical basis of $\mathbb{R}^n$. In other words, $w^K \in \operatorname{Po}(\mathcal{A}_K)$.

We can thus apply Algorithm 11 with $w^* = w^K$. From Theorem 3.2.4, this yields a vector $\alpha \in \mathbb{R}_+^{1+m+n}$ such that

$$w^K = \sum_{l=1}^{n+m+1} \alpha_l \begin{pmatrix} f_{i_l}(y_{i_l}^{k_l}) \\ A_{i_l} y_{i_l}^{k_l} \\ e_{i_l} \end{pmatrix} = \sum_{i=1}^n \sum_{l \in I_i} \alpha_l \begin{pmatrix} f_i(y_i^{k_l}) \\ A_i y_i^{k_l} \\ e_i \end{pmatrix},$$

where we regrouped the terms in i by defining $I_i := \{l \mid i_l = i\}$ for each $i = 1, \dots, n$. This yields the following theorem.

Theorem 3.3.2. *Suppose we run Algorithm 11 with input $\mathcal{A}_K$ and w^K defined in (3.22). It outputs a vector $\alpha \in \mathbb{R}_+^{n+m+1}$ such that*

$$w^K = \sum_{i=1}^{n} \sum_{l \in I_i} \alpha_l \begin{pmatrix} f_i(y_i^{k_l}) \\ A_i y_i^{k_l} \\ e_i \end{pmatrix}.$$

In particular, for all i, we have $\sum_{l \in I_i} \alpha_l = 1$. Moreover, $|I_i| > 1$ for at most $m+1$ indices i.

Proof. The fact that $\sum_{l \in I_i} = 1$ follows from the last n coordinates of w^K being equal to 1. In particular we must have $|I_i| \geq 1$ for all i. Since $\sum_{i=1}^{n} |I_i| = n+m+1$, there can only be at most $m+1$ indices i such that $|I_i| > 1$. □

Remark 3.3.1. *If Algorithm 11 is applied with a dense QR decomposition and column addition/deletion as described in Section 3.2.2.1, Lemma 3.2.1 tells us that the complexity is $O(nK(n+m)^2)$. Alternatively, since the vectors in $\mathcal{A}_K$ are sparse (only one coordinate among the last n is nonzero), sparse solvers might be more efficient in practice.*

Another way to obtain a decomposition of w^K is through the approach of Section 3.2.2.2. Indeed, observe that $\sum_{k=0}^{K-1} \sum_{i=1}^{n} \gamma_k = n$, and thus $w^K/n \in \text{conv}(\mathcal{A}_K)$, so that Algorithm 12 can be applied with $w^* = w^K/n$. The output $\tilde{w}^{K,T}$ is a convex combination of vectors in $\mathcal{A}_K$, namely

$$\tilde{w}^{K,T} = \sum_{l=0}^{T} \beta_l \begin{pmatrix} f_{i_l}(y_{i_l}^{k_l}) \\ A_{i_l} y_{i_l}^{k_l} \\ e_{i_l} \end{pmatrix} = \sum_{i=1}^{n} \sum_{l \in I_i} \beta_l \begin{pmatrix} f_i(y_i^{k_l}) \\ A_i y_i^{k_l} \\ e_i \end{pmatrix}, \tag{3.23}$$

where again we regrouped the terms in i by defining $I_i := \{l \mid i_l = i\}$ for each $i = 1, \ldots, n$. Finally, for each l, define $\alpha_l = \frac{\beta_l}{\sum_{l' \in I_{i_l}} \beta_{l'}}$ and set

$$w^{K,T} = \sum_{i=1}^{n} \sum_{l \in I_i} \alpha_l \begin{pmatrix} f_i(y_i^{k_l}) \\ A_i y_i^{k_l} \\ e_i \end{pmatrix}. \tag{3.24}$$

The next theorem gives a bound on the quality of the approximation of w^K by $w^{K,T}$, and provides conditions under which $w^{K,T}$ is well-defined. For clarity, we separate the coordinates corresponding to function values to the coordinates corresponding to matrix vector multiplication.

Theorem 3.3.3. *Suppose we run Algorithm 11 for $T \geq n-1$ iterations with input $\mathcal{A}_K$ and vector w^K/n, which outputs $\tilde{w}^{K,T}$ defined in (3.23), and that we build the final solution $w^{K,T}$ as in (3.24). Let*

$$h_0 = \left\| w^0 - w^K/n \right\|,$$

where w^0 is the starting point of Algorithm 12. Let $D_{\mathcal{A}_K}$ be the diameter of $\mathcal{A}_K$ and let $\delta(\mathcal{A}_K, \frac{w^K}{n})$ be as defined in Theorem 3.2.5. Assume that T is large enough so that

$$\frac{1}{n} > h_0 \exp\left(-\frac{T}{8} \left(\frac{\delta(\mathcal{A}_K, \frac{w^K}{n})}{D_{\mathcal{A}_K}^2} \right) \right). \tag{3.25}$$

It then holds that $|I_i| \geq 1$ *and that* $\sum_{l \in I_i} \alpha_l = 1$ *for all* i*. Moreover,*

$$\left| \sum_{i=1}^{n} \sum_{l \in I_i} \alpha_l f_i(y_i^{k_l}) - w_1^K \right| \leq n \left(\sqrt{n} \max_{i,y \in X_i} |f_i(y)| + 1 \right) h_0 \exp\left(-\frac{T\delta(\mathcal{A}_K, \frac{w^K}{n})}{8D_{\mathcal{A}_K}^2} \right),$$

$$\left\| \sum_{i=1}^{n} \sum_{l \in I_i} \alpha_l A_i y_i^{k_l} - w_{2:m+1}^K \right\| \leq n \left(\sqrt{n} \max_{i,y \in X_i} \|A_i y\| + 1 \right) h_0 \exp\left(-\frac{T\delta(\mathcal{A}_K, \frac{w^K}{n})}{8D_{\mathcal{A}_K}^2} \right).$$

In particular, $|I_i| > 1$ *for at most* $T - (n-1)$ *indices* i*.*

Proof. See Section 3.A.6. □

Remark 3.3.2. *One needs* $|I_i| \geq 1$ *for all* i *for* $w^{K,T}$ *to be well-defined. Inequality (3.25) provides a sufficient condition for it to be true. It is however hard to check in practice, since both* $\delta(\mathcal{A}_K, \frac{w^K}{n})$ *and* $D_{\mathcal{A}_K}$ *are usually not available. Simple algebraic manipulations show that a sufficient condition for it to hold is*

$$n - 1 \geq \frac{8D_{\mathcal{A}_K}^2}{\delta(\mathcal{A}_K, \frac{w^K}{n})} \ln(nh_0).$$

Since both $D_{\mathcal{A}_K}$ *and* $\delta(\mathcal{A}_K, \frac{w^K}{n})$ *depend on* n*, we cannot be sure that this holds, even as* n *grows. In the considered practical application, we observed that one had* $|I_i| \geq 1$ *for all* i *for values of* T *smaller than* $n + m$*.*

Whether the trimming was performed using Algorithm 11 or Algorithm 12, we get an output of the form

$$\bar{w}^K = \sum_{i=1}^{n} \sum_{l \in I_i} \alpha_l \begin{pmatrix} f_i(y_i^{k_l}) \\ A_i y_i^{k_l} \\ e_i \end{pmatrix}, \tag{3.26}$$

where $\sum_{l \in I_i} \alpha_l = 1$ for all i. Let

$$q := |\{i \in \{1, \dots, n\} \mid |I_i| > 1\}|. \tag{3.27}$$

In other words, q is the number of indices i such that $|I_i| > 1$. In the case of Algorithm 11, Theorem 3.3.2 gives $q \leq m+1$ and $\bar{w}^K = w^K$. On the other hand, for Algorithm 12, $\bar{w}^K = w^{K,T}$ is an approximation of w^K (whose quality depends on T and is established in Theorem 3.3.3), and $q \leq T + 1 - n$.

3.3.3 Final reconstruction

Define the first $1 + m$ coordinates of $\bar{w}^K$ in (3.26) as $\bar{z}^K$, i.e.

$$\bar{z}^K = \sum_{i=1}^{n} \sum_{l \in I_i} \alpha_l \begin{pmatrix} f_i(y_i^{k_l}) \\ A_i y_i^{k_l} \end{pmatrix}, \tag{3.28}$$

In this section, we explore different ways to reconstruct a solution of the primal problem (P). In order to lighten notations, we define the error terms induced by trimming using Algorithm 12 as

$$\mathrm{Err}_f(n,T) := n \left(\sqrt{n} \max_{i,y \in X_i} |f_i(y)| + 1 \right) h_0 \exp\left(-\frac{T\delta(\mathcal{A}_K, \frac{w^K}{n})}{8D_{\mathcal{A}_K}^2} \right)$$

$$\mathrm{Err}_A(n,T) := n \left(\sqrt{n} \max_{i,y \in X_i} \|A_i y\| + 1 \right) h_0 \exp\left(-\frac{T\delta(\mathcal{A}_K, \frac{w^K}{n})}{8D_{\mathcal{A}_K}^2} \right),$$

where h_0, $\delta(\mathcal{A}_K, \frac{w^K}{n})$ and $D_{\mathcal{A}_K}$ are defined in Theorem 3.3.3. We start with the following useful technical lemma.

Lemma 3.3.4. *Let v^* be the optimal value of (D) and (CoP). Assume that we run Algorithm 13 for K iterations. If Algorithm 11 is used to trim the number of elements then*

$$\bar{z}_1^K \leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}},$$
$$\left\|\bar{z}_{2:m+1}^K - b\right\|_+ \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}}.$$

On the other hand, if Algorithm 12 for $T \geq n-1$ iterations is used to trim the number of elements, and under the condition of Theorem 3.3.3,

$$\bar{z}_1^K \leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_f(n,T),$$
$$\left\|\bar{z}_{2:m+1}^K - b\right\|_+ \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_A(n,T).$$

Proof. Observe that

$$\bar{z}_1^K - v^* = \bar{z}_1^K - z_1^K + z_1^K - v^*,$$
$$\left\|\bar{z}_{2:m+1}^K - b\right\|_+ \leq \left\|\bar{z}_{2:m+1}^K - z_{2:m+1}^K\right\|_+ + \left\|z_{2:m+1}^K - b\right\|_+.$$

From Theorem 3.3.1, we get

$$z_1^K - v^* \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}},$$
$$\left\|z_{2:m+1}^K - b\right\|_+ \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}}.$$

When trimming is performed using Algorithm 11, we have $\bar{z}^K = z^K$, and we are done. When trimming is performed using Algorithm 12, we can bound $\bar{z}_1^K - z_1^K$ and $\left\|\bar{z}_{2:m+1}^K - z_{2:m+1}^K\right\|_+$ using Theorem 3.3.3. □

3.3.3.1 Convex domains

When the function domains are convex, a natural candidate solution is $\bar{x} \in \mathbb{R}^d$ defined as

$$\bar{x}_i = \sum_{l \in I_i} \alpha_l y_i^{k_l}, \quad i = 1, \ldots, n. \tag{3.29}$$

The next result recovers the bound of Theorem 3.2.7 for the proposed solution (3.29), up to an additional term which decreases with K. It also shows that when trimming is performed using the approximate Carathéodory algorithm Algorithm 12, one can reduce the factor m in front of the nonconvexity measure, at the price of an additional term which decreases exponentially with n.

Theorem 3.3.4. *Suppose that X_i is convex for all $i = 1, \ldots, n$, and let v^* be the optimal value of (D) and (CoP). Assume that we run Algorithm 13 for K iterations. Let $\bar{x} \in \mathbb{R}^d$ be the final solution defined in (3.29).*

If Algorithm 11 is used to trim the number of elements, then $\bar{x}$ satisfies

$$\sum_{i=1}^n f_i(\bar{x}_i) \leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + (m+1)\max_i \rho(f_i),$$
$$\left\|\sum_{i=1}^n A_i \bar{x}_i - b\right\|_+ \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}}.$$

On the other hand, if Algorithm 12 for $T \geq n-1$ iterations is used to trim the number of elements, and under the condition of Theorem 3.3.3, $\bar{x}$ satisfies

$$\sum_{i=1}^{n} f_i(\bar{x}_i) \leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_f(n,T) + (T+1-n)\max_i \rho(f_i),$$
$$\left\| \sum_{i=1}^{n} A_i \bar{x}_i - b \right\|_+ \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_A(n,T).$$

Proof. Observe that

$$\begin{aligned} \sum_{i=1}^{n} f_i(\bar{x}_i) - v^* &= \left(\sum_{i=1}^{n} f_i(\bar{x}_i) - \bar{z}_1^K \right) + \left(\bar{z}_1^K - v^* \right) \\ \left\| \sum_{i=1}^{n} A_i \bar{x}_i - b \right\|_+ &\leq \left\| \sum_{i=1}^{n} A_i \bar{x}_i - \bar{z}_{2:m+1}^K \right\|_+ + \left\| \bar{z}_{2:m+1}^K - b \right\|_+ \end{aligned} \tag{3.30}$$

We can apply Lemma 3.3.4 to bound the last term of each equation. Moreover,

$$\begin{aligned} \sum_{i=1}^{n} f_i(\bar{x}_i) - \bar{z}_1^K &= \sum_{i=1}^{n} f_i(\bar{x}_i) - \sum_{i=1}^{n} \sum_{l \in I_i} \alpha_l f_i(y_i^{k_l}) \\ &= \sum_{i, |I_i|>1} \left(f_i \left(\sum_{l \in I_i} \alpha_l y_i^{k_l} \right) - \sum_{l \in I_i} \alpha_l f_i(y_i^{k_l}) \right) \\ &\leq \sum_{i, |I_i|>1} \rho(f_i) \\ &\leq q \max_i \rho(f_i), \end{aligned}$$

and

$$\sum_{i=1}^{n} A_i \bar{x}_i - \bar{z}_{2:m+1}^K = \sum_{i=1}^{n} A_i \sum_{l \in I_i} \alpha_l y_i^{k_l} - \sum_{i=1}^{n} \sum_{l \in I_i} \alpha_l A_i y_i^{k_l} = 0.$$

□

3.3.3.2 General domains

When the domains X_i are not necessarily convex, the solution $\bar{x}$ defined in (3.29) is in general not feasible, as for i such that $|I_i| > 1$, $\bar{x}_i$ is a non-trivial convex combination. In this section we suggest several approaches to construct feasible solutions.

3.3.3.2.1 Reconstruction via sampling Our first reconstruction scheme is a randomized approach. We suggest setting the solution $\bar{x}_i$ as $y_i^{k_l}$ with probability α_l. In other words, for each $i = 1, \ldots, n$, define L_i as the random variable taking values in I_i such that for all $l \in I_i$, $P(L_i = l) = \alpha_l$. The reconstructed solution is then

$$\begin{aligned} l_i &\sim L_i \\ \bar{x}_i &= y_i^{k_{l_i}} \end{aligned} \quad i = 1, \ldots, n. \tag{3.31}$$

Since $y_i^{k_l} \in X_i$ for all i and $l \in I_i$, we immediately get that $\bar{x}_i \in X_i$ for all i. We also get the following duality gap bound on the obtained solutions. In particular, the obtained solution satisfies the constraint in expectation (up to an arbitrarily small term).

Theorem 3.3.5. *Let v^* be the optimal value of (D) and (CoP). Assume that we run Algorithm 13 for K iterations. Let $\bar{x} \in \mathbb{R}^d$ be the final solution defined in (3.31).*

If Algorithm 11 is used to trim the number of elements, then $\bar{x}$ satisfies

$$\sum_{i=1}^{n} f_i(\bar{x}_i) \leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + (m+1)\max_i \gamma(f_i),$$
$$\mathbb{E}\left[\sum_{i=1}^{n} A_i\bar{x}_i - b\right] \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}}.$$

On the other hand, if Algorithm 12 for $T \geq n-1$ iterations is used to trim the number of elements, and under the condition of Theorem 3.3.3, $\bar{x}$ satisfies

$$\sum_{i=1}^{n} f_i(\bar{x}_i) \leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_f(n,T) + (T+1-n)\max_i \rho(f_i),$$
$$\mathbb{E}\left[\sum_{i=1}^{n} A_i\bar{x}_i - b\right] \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_A(n,T).$$

Proof. Observe that

$$\begin{aligned}
\sum_{i=1}^{n} f_i(\bar{x}_i) - v^* &= \left(\sum_{i=1}^{n} f_i(\bar{x}_i) - \bar{z}_1^K\right) + \left(\bar{z}_1^K - v^*\right) \\
\sum_{i=1}^{n} A_i\bar{x}_i - b &= \left(\sum_{i=1}^{n} A_i\bar{x}_i - \bar{z}_{2:m+1}^K\right) + \left(\bar{z}_{2:m+1}^K - b\right) \\
&\leq \left(\sum_{i=1}^{n} A_i\bar{x}_i - \bar{z}_{2:m+1}^K\right) + \left\|\bar{z}_{2:m+1}^K - b\right\|_+ .
\end{aligned} \tag{3.32}$$

Lemma 3.3.4 gives a bound on the last term of each equation. Moreover, we have

$$\begin{aligned}
\sum_{i=1}^{n} f_i(\bar{x}_i) - \bar{z}_1^K &= \sum_{i=1}^{n} f_i(\bar{x}_i) - \sum_{i=1}^{n}\sum_{l\in I_i} \alpha_l f_i(y_i^{k_l}) \\
&= \sum_{i,|I_i|>1} \left(f_i(\bar{x}_i) - \sum_{l\in I_i} \alpha_l f_i(y_i^{k_l})\right) \\
&\leq \sum_{i,|I_i|>1} \gamma(f_i) \\
&\leq q\max_i \gamma(f_i),
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{E}\left[\sum_{i=1}^{n} A_i\bar{x}_i - \bar{z}_{2:m+1}^K\right] &= \sum_{i=1}^{n} A_i\mathbb{E}\left[\bar{x}_i\right] - \bar{z}_{2:m+1}^K \\
&= \sum_{i=1}^{n} A_i \sum_{l\in I_i} \alpha_l y_i^{k_l} - \sum_{i=1}^{n}\sum_{l\in I_i} \alpha_l A_i y_i^{k_l} = 0.
\end{aligned}$$

□

3.3.3.2.2 Reconstruction under additional domain assumption The above reconstruction gives a solution which is feasible in expectation. In particular, one can build solutions of the form (3.31) by repeatedly sampling until a good enough solution is obtained. However

there is no guarantee that a fully-feasible solution will be obtained with that procedure. In order to construct such a solution, we require additional assumptions on the feasible set.

One such assumption states that for each domain, there is a vector for which the value of the affine constraint is smaller than for all other vectors.

Assumption 3.3.1. *For all $i = 1, \dots, n$ and all $x_i \in \operatorname{conv} X_i$, there exists $\hat{x}_i \in X_i$ such that $A_i \hat{x}_i \leq A_i x_i$.*

This assumption holds, for example, when $A_i x_i \geq 0$ for all $x_i \in X_i$ and $0 \in X_i$. Although quite restrictive, this is known to hold for some practical applications, such as supply chain managament [VEGM14]. Under Assumption 3.3.1, we can construct a final solution as

$$\bar{x}_i = \begin{cases} \sum_{l \in I_i} \alpha_l y_i^{k_l} & \text{if } |I_i| = 1, \\ \hat{x}_i & \text{if } |I_i| > 1. \end{cases} \tag{3.33}$$

The next theorem states that the resulting solution recovers the duality gap bound of Theorem 3.2.8, up to an additional decreasing term. The proof is similar to the one of Theorem 3.3.4, so we leave it to the appendix.

Theorem 3.3.6. *Let v^* be the optimal value of (D) and (CoP) and suppose that Assumption 3.3.1 holds. Assume that we run Algorithm 13 for K iterations. Let $\bar{x} \in \mathbb{R}^d$ be the final solution defined in (3.33).*

If Algorithm 11 is used to trim the number of elements, then $\bar{x}$ satisfies

$$\sum_{i=1}^{n} f_i(\bar{x}_i) \leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + (m+1) \max_i \gamma(f_i),$$
$$\left\| \sum_{i=1}^{n} A_i \bar{x}_i - b \right\|_+ \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}}.$$

On the other hand, if Algorithm 12 for $T \geq n-1$ iterations is used to trim the number of elements, and under the condition of Theorem 3.3.3, $\bar{x}$ satisfies

$$\sum_{i=1}^{n} f_i(\bar{x}_i) \leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_f(n,T) + (T+1-n) \max_i \gamma(f_i),$$
$$\left\| \sum_{i=1}^{n} A_i \bar{x}_i - b \right\|_+ \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_A(n,T).$$

Proof. See Section 3.A.7. □

3.3.3.2.3 Reconstruction after solving perturbed problem Assumption 3.3.1 can sometimes be restrictive. Alternatively, following [VEG+16], one can consider a perturbed version of (CoP), defined in (CoP$_\theta$) with $\theta \in \mathbb{R}^m$ defined as

$$\theta_k := (m+1) \max_i \left(\max_{x_i \in X_i} ((A_i)_{k,:})^\top x_i - \min_{x_i \in X_i} ((A_i)_{k,:})^\top x_i \right). \tag{3.34}$$

This will ensure that the error made when constructing a final solution is small enough so that the solution is still primal feasible. We also need the following technical assumption, which is a Slater-type condition on the perturbed problem (CoP$_\theta$).

Assumption 3.3.2. *For all $i = 1, \dots, n$, there exists $\xi > 0$ and $\hat{x}_i \in \operatorname{conv} X_i$ such that*

$$\sum_{i=1}^{n} A_i \hat{x}_i \leq b - \theta - n\xi \mathbf{1}_m.$$

Finally, we suggest the following reconstruction scheme

$$\begin{aligned} l_i &= \arg\max_{l \in I_i} \alpha_l, \\ \bar{x}_i &= y_i^{k_{l_i}}, \end{aligned} \qquad i = 1, \dots, n. \tag{3.35}$$

Taking the maximum over all coefficients in the convex combinations allows us to prove a stronger bound than in Theorem 3.2.9.

Theorem 3.3.7. *Let v^* be the optimal value of (D) and (CoP) and suppose that Assumption 3.3.2 holds. Suppose that we apply the method of Sections 3.3.1 and 3.3.2 to problem (CoP_θ) with θ defined in (3.34). Assume that we run Algorithm 13 for K iterations. Let $\bar{x} \in \mathbb{R}^d$ be the final solution defined in (3.35). Let q be the number of non-trivial convex combinations in the output of the Carathéodory algorithm, as defined in (3.27).*

If Algorithm 11 is used to trim the number of elements, then $\bar{x}$ satisfies

$$\begin{aligned} \sum_{i=1}^n f_i(\bar{x}_i) &\leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + \left(\frac{q(m+1)}{q+(m+1)} + \frac{\|\theta\|_\infty}{\xi} \right) \max_i \gamma(f_i), \\ \left\| \sum_{i=1}^n A_i \bar{x}_i - b \right\|_+ &\leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}}. \end{aligned}$$

On the other hand, if Algorithm 12 for $T \geq n-1$ iterations (and $T \leq n+m$) is used to trim the number of elements, and under the condition of Theorem 3.3.3, $\bar{x}$ satisfies

$$\begin{aligned} \sum_{i=1}^n f_i(\bar{x}_i) &\leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_f(n,T) + \left(\frac{q(m+1)}{q+(m+1)} + \frac{\|\theta\|_\infty}{\xi} \right) \max_i \gamma(f_i), \\ \left\| \sum_{i=1}^n A_i \bar{x}_i - b \right\|_+ &\leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_A(n,T). \end{aligned}$$

Proof. See Section 3.A.8. □

In the above theorem, plugging in $q = (m+1)$ gives a term $\frac{m+1}{2}$ in the bound, which improves the term in Theorem 3.2.9 by a factor 2.

3.3.3.2.4 Reconstruction after approximate Carathéodory To finish this section, observe that the approximate Carathéodory approach of Algorithm 12 might yield a primal feasible solution after $T = n-1$ iterations, provided that at each iteration, it picks up a different index i. This holds under the following assumption, which we have already mentioned in Remark 3.3.2.

Assumption 3.3.3. *The following inequality holds*

$$n - 1 \geq \frac{8D^2_{\mathcal{A}_K}}{\delta(\mathcal{A}_K, \frac{w^K}{n})} \ln(nh_0),$$

where h_0, $\delta(\mathcal{A}_K, \frac{w^K}{n})$ and $D_{\mathcal{A}_K}$ are defined in Theorem 3.3.3.

In that case, Theorem 3.3.3 shows that $|I_i| \geq 1$ for all i. Since $T = n-1$, we must actually have $|I_i| = 1$ for all i. Let l_i such that $I_i = \{l_i\}$. One can then build a final solution $\bar{x} \in \mathbb{R}^d$ as

$$\bar{x}_i = y_i^{k_{l_i}}, \quad i = 1, \dots, n. \tag{3.36}$$

Under the above assumption and reconstruction, we can recover duality gap bounds which do not depend on the nonconvexity of the objective functions.

Theorem 3.3.8. *Suppose that Assumption 3.3.3 holds and let v^* be the optimal value of (D) and (CoP). Assume that we run Algorithm 13 for K iterations and Algorithm 12 for $T = n - 1$ iterations to trim the number of elements. Let $\bar{x} \in \mathbb{R}^d$ be the final solution defined in (3.36). Then $\bar{x}$ satisfies*

$$\sum_{i=1}^{n} f_i(\bar{x}_i) \leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_f(n,T)$$
$$\left\| \sum_{i=1}^{n} A_i \bar{x}_i - b \right\|_+ \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_A(n,T).$$

Proof. This is an immediate consequence of Lemma 3.3.4 and the discussion above. □

3.4 The special case of mixed integer linear programs

In this section, we show that in the case of MILPs, we can obtain linear rates of convergence in K, instead of sublinear rates, by replacing Algorithm 13 with a conditional gradient methods with linear rates on strongly convex objectives.

For MILPs, we have $f_i(x_i) = c_i^\top x_i$, and some of the variables are constrained to be integral. In other words, the domains X_i can be written as

$$X_i = \{x_i \in \mathbb{Z}^{d_i^1} \times \mathbb{R}^{d_i^2} \mid C_i x_i \leq g_i\}, \tag{3.37}$$

where $d_i^1 + d_i^2 = d_i$. In constrast with the approach of Section 3.3.1, this time we require access to the solution $x^* \in \mathbb{R}^d$ of (CoP), and not only of the optimal value. Define $z^* \in \mathbb{R}^{1+m}$ as

$$z^* = \begin{pmatrix} v^* \\ Ax^* \end{pmatrix}, \tag{3.38}$$

and thus

$$z^* \in \sum_{i=1}^{n} \left\{ \begin{pmatrix} c_i^\top x_i \\ A_i x_i \end{pmatrix} \mid x_i \in \operatorname{conv} X_i \right\}. \tag{3.39}$$

Just like in Section 3.3.1, we set $\mathcal{C}_i = \left\{ \begin{pmatrix} c_i^\top x_i \\ A_i x_i \end{pmatrix} \mid x_i \in \operatorname{conv} X_i \right\}$ and $\mathcal{C} = \sum_{i=1}^n \mathcal{C}_i$. Again, we aim to approximate z^* with extreme points of $\mathcal{C}_i$, with a conditional gradient method. This time, however, we minimize the strongly convex program

$$\text{minimize } \frac{1}{2} \|z - z^*\|^2 \quad \text{subject to } z \in \sum_{i=1}^{n} \mathcal{C}_i. \tag{3.40}$$

At each iteration k, the gradient of the objective function with respect to z^k is given by (α_k, g^k) with $\alpha_k = z_1^k - v^*$ and $g^k = z_{2:m+1}^k - Ax^*$. The linear minimization steps then reads

$$s^k \in \operatorname*{arg\,min}_{z \in \sum_{i=1}^n \mathcal{C}_i} z^\top (\alpha_k, g^k). \tag{3.41}$$

This amounts to solving, for each $i \in \{1, \ldots, n\}$,

$$y_i^k \in \operatorname*{arg\,min}_{x_i \in \operatorname{conv} X_i} \alpha_k (c_i^\top x_i) + (g^k)^\top A_i x_i. \tag{3.42}$$

This is a linear minimization over a convex polyhedra, which can be computed efficiently in many applications.

Because the objective function is smooth and strongly convex, we can apply variants of Frank Wolfe with linear convergence rates on strongly convex objectives. We suggest applying the fully-corrective variant, since in that case it is tractable, but one could also use the away-step variant and get similar convergence guarantees [LJJ15, PR19]. We summarize the method in Algorithm 14. The procedure to perform the minimization over the convex hull on line 12 is the same as in Section 3.2.2.2, so we omit it here.

Algorithm 14 Fully corrective Frank Wolfe algorithm to obtain primal points for MILPs

1: Input: $x^* \in \mathbb{R}^{1+m}$ solution of (CoP), number of iterations K.
2: For all $i = 1, \dots, n$, choose $y_i^0 \in X_i$.
3: Set $z^0 = \sum_{i=1}^n \begin{pmatrix} f_i(y_i^0) \\ A_i y_i^0 \end{pmatrix}$
4: $\mathcal{S}_0 = \{z_0\}$
5: **for** $k = 0, \dots, K-1$ **do**
6: Set $\alpha_k = z_1^k - v^*$ and $g^k = z_{2:m+1}^k - Ax^*$.
7: **for** $i = 1, \dots, n$ **do**
8: Solve

$$y_i^k \in \underset{x_i \in \operatorname{conv} X_i}{\arg\min}\ \alpha_k (c_i^\top x_i) + (g^k)^\top A_i x_i$$

9: **end for**
10: Set $s^k = \sum_{i=1}^n \begin{pmatrix} f_i(y_i^k) \\ A_i y_i^k \end{pmatrix}$.
11: $\mathcal{S}_{k+1} = \mathcal{S}_k \cup \{s^k\}$
12: $z^{k+1} = \arg\min_{s \in \operatorname{conv} \mathcal{S}_{k+1}} \frac{1}{2} \|s - z^*\|^2$.
13: **end for**
14: **return** z^K

Theorem 3.4.1. *Suppose we run Algorithm 14 for K iterations. Then the output satisfies*

$$\left\| z^K - z^* \right\|^2 \leq D_{\mathcal{C}}^2 \exp\left(-\frac{K \omega_{\mathcal{C}}^2}{4 D_{\mathcal{C}}^2} \right)$$

where $D_{\mathcal{C}} = diam(\mathcal{C}, \|\cdot\|) = \max_{z,z' \in \mathcal{C}} \|z - z'\|$, and $\omega_{\mathcal{C}}$ is the facial distance of $\mathcal{C}$.

Proof. This is the result from [LJJ15, PR19]. □

Remark 3.4.1. *We are able to use a strongly convex objective in this approach because it does not matter whether α_k is positive or negative for (3.42) to be tractable. On the contrary, in Section 3.3.1, if $\alpha_k < 0$, then the linear minimization oracle (3.19) requires minimizing a concave function, i.e. is not tractable.*

The remainder of the approach to obtain primal solutions is the same as in Section 3.3, and the theorems derived in Section 3.3.3 still hold, except that one can replace the term $\frac{2D_{\mathcal{C}}}{\sqrt{K+1}}$ by $D_{\mathcal{C}} \exp\left(-\frac{K\omega_{\mathcal{C}}^2}{8D_{\mathcal{C}}^2} \right)$ in all of them.

3.5 Numerical results

We apply our method to a unit commitment (UC) problem aimed at finding a scheduling of power-generating units with minimum cost, subject to a demand constraint. We first describe

the problem. For an in-depth survey of different UC problems and solvers, see [Pad04, MS14]. We are given a network of n power-generating units, and a time-horizon N. At each time step $t \in \{1, \dots, N\}$, unit i can either be turned on or off, which is denoted by a binary variable $u_t^i \in \{0, 1\}$. Turning on unit i has cost c_{01}^i, and turning it off has cost c_{10}^i. The power output of unit i at time step t is written $g_i^t \in \mathbb{R}$. When $u_t^i = 0$, unit i is turned off and $g_t^i = 0$. When $u_t^i = 1$, the power generation g_t^i lies between a minimum value $\underline{g}^i$ and a maximum value $\overline{g}^i$. As is standard for UC problems [FALA17], we assume the cost of production to be a quadratic function of g_t^i. We assume that all units are turned off at time step 0. The cost functions can then be written

$$f_i(u^i, g^i) = \sum_{t=1}^{N} \ell_i(u_t^i, g_t^i), \quad i = 1, \dots, n, \tag{3.43}$$

where

$$\ell_i(u_t^i, g_t^i) = \begin{cases} c_{10} u_{t-1}^i & \text{if } u_t^i = 0, \\ c_{01}^i (1 - u_{t-1}^i) + \beta_i (g_t^i)^2 + \gamma_i g_t^i + \omega_i & \text{if } u_t^i = 1, \end{cases}$$

where $u_0^i = 0$ by assumption. The domain of the functions is

$$\operatorname{dom}(f_i) = \left\{ (u^i, g^i) \in \{0,1\}^N \times \mathbb{R}^N \;\middle|\; \begin{array}{l} (u_t^i, g_t^i) = (0, 0) \text{ or} \\ (u_t^i, g_t^i) \in (1, [\underline{g}^i, \overline{g}^i]), \end{array} \text{ for all } t = 1, \dots, N \right\}. \tag{3.44}$$

Finally, the network must meet a demand $D_t \in \mathbb{R}$ at each time step t. The optimization of production costs can then be written as

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} f_i(u^i, g^i) \\ \text{subject to} \quad & \sum_{i=1}^{n} g_t^i \geq D_t, \quad t = 1, \dots, N \\ & (u_i, g_i) \in \operatorname{dom}(f_i), \quad i = 1, \dots, n. \end{aligned} \tag{3.45}$$

This fits exactly withing the setting of (P) with $m = N$. We show in the next subsection how to compute the linear minimization oracle (3.19), and in particular the Fenchel conjugate of f_i.

3.5.1 Computing the conjugate

In this section we show how to compute the conjugate of the function f_i defined in (3.43). For ease of notation, we drop all indices i and superscripts i. Suppose we wish to compute $f^*(v, p)$ for $v \in \mathbb{R}^N$ and $p \in \mathbb{R}^N$. Let us define

$$\Delta_{k,0} = \max_{\substack{u_1,\dots,u_k, \\ g_1,\dots,g_k, \\ u_k = 0}} \left\{ \sum_{t=1}^{k} (u_t v_t + g_t p_t) - \sum_{t=1}^{k} \ell(u_t, g_t) \right\},$$

$$\Delta_{k,1} = \max_{\substack{u_1,\dots,u_k, \\ g_1,\dots,g_k, \\ u_k = 1}} \left\{ \sum_{t=1}^{k} (u_t v_t + g_t p_t) - \sum_{t=1}^{k} \ell(u_t, g_t) \right\}.$$

One then notices that $f^*(v, p) = \max(\Delta_{N,0}, \Delta_{N,1})$. Moreover, we have

$$\Delta_{1,0} = 0, \quad \Delta_{1,1} = v_1 - c_{01} + \max_{\underline{g} \leq g \leq \overline{g}} \left\{ p_1 g - \beta g^2 - \gamma g - \omega \right\},$$

and for $k > 1$,

$$\Delta_{k,0} = \max\left(\Delta_{k-1,0}, \Delta_{k-1,1} - c_{10}\right),$$
$$\Delta_{k,1} = \max\left(\Delta_{k-1,0} - c_{01}, \Delta_{k-1,1}\right) + v_k + \max_{\underline{g} \leq g \leq \overline{g}} \left\{p_k g - \beta g^2 - \gamma g - \omega\right\}.$$

The objective in the maximization steps is a one-dimensional quadratic over a bounded closed interval, so obtaining the optimal value is straightforward. Thus the value $f^*(v, p)$ can be obtained recursively, and the complexity is $O(N)$. Moreover, the gradient can be obtained by keeping track of the arguments yielding the maximum in the recursion.

3.5.2 Linear minimization over the domain

When $\alpha_k = 0$ in (3.19), one needs to compute a linear minimization over $\operatorname{dom} f_i$. We show how to do so now. As in the computation of the conjugate above, we drop all indices i. Given $(v, p) \in \mathbb{R}^N \times \mathbb{R}^N$, one must solve

$$\min_{(u,g) \in \operatorname{dom} f} \sum_{t=1}^{N} (u_t v_t + g_t p_t).$$

The structure of $\operatorname{dom} f$ tells us that the problem is separable, and the subproblems can be rewritten as

$$\begin{aligned} \min_{u_t, g_t} \quad & u_t v_t + g_t p_t \\ \text{subject to } & (u_t, g_t) = (0, 0) \text{ or } (u_t, g_t) \in (1, [\underline{g}, \overline{g}]). \end{aligned}$$

Since the objective is linear, the minimum is necessarily attained at an extreme point, i.e. at one of $(0, 0)$, $(1, \underline{g})$ or $(1, \overline{g})$.

3.5.3 Results

We generate random instances of the UC problem with $n = 50$ units over $N = 10$ time steps. Details on the data generation process can be found in Section 3.D. To compute the optimal dual value v^*, we solve the dual (D) using projected gradient descent with a decreasing stepsize, which we stop when no more significant progress on the dual objective value is achieved. We run Algorithm 13 for $K = 10^4$ iterations. We then run either Algorithm 11 or Algorithm 12 to trim the result. Moreover, we suggest the following simple heuristic to improve the performance of Algorithm 12. After the QP on line 8 is solved, we remove from the active set the vectors whose coefficient in the newly computed convex combination on line 9 is smaller than some tolerance value. This has the benefit of limiting the impact of a bad choice of initialization of the active set. We set the tolerance value to $1e^{-9}$. For Algorithm 12, we solve the QP using ProxQP [BSEK+23]. The code to reproduce the experiments is available at: `https://github.com/bpauld/NonConvexOpt`.

The bounds provided in Section 3.3.3 give solutions that are feasible up to an arbitrarily small term. In addition, the domains of the functions are nonconvex, so we have to resort to one of the strategies in Section 3.3.3.2 to construct a solution in the domain of f_i. Early experiments showed that solving a slightly perturbed version of the problem yielded fully feasible primal solutions, no matter which of the reconstruction strategies of Section 3.3.3.2 was used. In the spirit of the approach described in paragraph 3.3.3.2.3, we thus apply our method to problem (3.45) perturbed by a vector $\theta \in \mathbb{R}^N$. Although the discussion in 3.3.3.2.3 suggests setting θ to $(N+1)\max_i \overline{g}^i$, this turned out to be unnecessarily large in our experiments. Instead, we set $\theta = \zeta \max_i \overline{g}^i$, with initial value $\zeta = 1$. If the solution satisfies the demand, we stop there.

Table 3.1: Results for different random instances of the unit commitment problem with $n = 50$ and $m = 10$. $x^{\max}$ corresponds to the reconstructed solution built by taking the maximum coefficient in every convex combination as in Equation (3.35), while x^F corresponds to the reconstructed solution whose feasibility is better than the obtained convex combination, as in Equation (3.33). Exact Carathéodory corresponds to the case where the trimming is performed using Algorithm 11, while approximate corresponds to the trimming performed with Algorithm 12. We see that the recovered solutions have duality gap much below the theoretical value of $(m+1)\max_i \gamma(f_i)$.

			Exact Carathéodory		Approx. Carathéodory	
v^*	ζ	$\max_i \gamma(f_i)$	$f(x^{\max})$	$f(x^F)$	$f(x^{\max})$	$f(x^F)$
87 772	1	28 395	94 843	95 989	95 505	95 303
62 412	2	23 881	76 873	79 193	76 205	78 356
62 259	1	25 079	70 089	69 817	69 376	70 895
85 452	1	23 231	92 910	94 679	93 555	93 271
113 128	1	25 301	123 348	122 856	122 011	121 831
90 860	2	23 802	109 098	109 592	109 780	109 284
78 967	1	21 190	88 230	86 315	87 493	86 745
90 168	1	23 364	98 132	99 154	99 702	98 958
108 709	2	27 126	126 610	129 285	126 492	129 304
44 832	1	22 921	50 247	51 078	50 417	52 286

Otherwise, we increase ζ by 1 and solve the new perturbed problem. We repeat this until the obtained solution meets the demand.

Table 3.1 presents the results for 10 randomly generated instances. The first observation is that it is always enough to set ζ to either 1 or 2 to obtain solutions that meet the power demand. Second, while the theoretical duality gap is $(N+1)\max_i \gamma(f_i)$, all of our experiments yield solutions whose duality gap is smaller than $\max_i \gamma(f_i)$, a considerable improvement. Finally, the quality of the solutions is not greatly impacted by the Carathéodory approach (exact as in Algorithm 11 or approximate as in Algorithm 12) and by the final reconstruction scheme (Equation (3.35) or Equation (3.33)).

3.A Additional proofs

3.A.1 Proof of Theorem 3.2.3

Theorem 3.2.3 (Conic Carathéodory)**.** *Let $S \subset \mathbb{R}^p$ and $w \in \operatorname{Po} S$. Then there exists $\{\lambda_j\}_{j=1}^p \subset \mathbb{R}_+$, $\{w_j\}_{j=1}^p \subset S$, such that $x = \sum_{j=1}^p \lambda_j w_j$.*

Proof. Since $w \in \operatorname{Po}(S)$, there exists $k \in \mathbb{N}$, $\lambda \in \mathbb{R}_+^k$ and $w_1, \dots, w_k \in S$ such that $w = \sum_{j=1}^k \lambda_j w_j$. We assume $k > p$ (if $k \leq p$, we are already done). We claim that we can rearrange the coefficients so that one of the w_j's has coefficient 0. To do so, suppose $\lambda_j > 0$ for all $j = 1, \dots, k$ (otherwise we are done). Since $k > p$, the w_j's are linearly dependent, so there exists $\delta_1, \dots, \delta_k$ not all zero such that

$$\sum_{j=1}^k \delta_j w_j = 0.$$

Let $t^* = \max\{t \geq 0 \mid \lambda_j - t\delta_j \geq 0 \text{ for all } j = 1, \dots, k\}$ and set $\lambda'_j = \lambda_j - t^*\delta_j$ for all j. Then $\lambda'_j \geq 0$ for all j. Moreover, there exists j_0 such that $\lambda'_{j_0} = 0$ and

$$\sum_{j=1}^k \lambda'_j w_j = \sum_{j=1}^k \lambda_j w_j - t^* \sum_{j=1}^k \delta_j w_j = x,$$

which proves that we can indeed reduce the number of strictly positive coefficients to $k-1$. This argument can be repeated for r iterations as long as $k - r > p$, which proves the theorem. □

3.A.2 Proof of Theorem 3.2.7

Theorem 3.2.7. *Let v^* be the dual optimal value defined in (3.4) and $\rho(f_i)$ be as defined in (3.2). Assume that X_i is convex for all i. There exists $x^* \in \mathbb{R}^d$ such that $Ax^* \leq d$ and*

$$v^* \leq \sum_{i=1}^n f_i(x_i^*) \leq v^* + (m+1) \max_i \rho(f_i).$$

Proof. By Lemma 3.B.1, there exists $\bar{x} \in \mathbb{R}^d$ and set $S \subset \{1, \dots, n\}$ of cardinality at most $m+1$ such that $A\bar{x} \leq b$, $\bar{x}_i \in X_i$ for $i \notin S$ and $\bar{x}_i \in \operatorname{conv} X_i = X_i$ (since X_i is convex by assumption) such that

$$v^* = \sum_{i \notin S} f_i(\bar{x}_i) + \sum_{i \in S} f_i^{**}(\bar{x}_i)$$

Setting $x^* = \bar{x}$, observe that x^* is primal feasible. Moreover,

$$\begin{aligned} v^* &\leq \sum_{i=1}^n f_i(x_i^*) \\ &= \sum_{i \notin S} f_i(x_i^*) + \sum_{i \in S} f_i(x_i^*) \\ &= v^* + \sum_{i \in S} (f_i(x_i^*) - f_i^{**}(x_i^*)) \\ &\leq v^* + (m+1) \max_i \rho(f_i). \end{aligned}$$

□

3.A.3 Proof of Theorem 3.2.8

Theorem 3.2.8. *Let v^* be the dual optimal value defined in (3.4) and $\gamma(f_i)$ be defined in (3.13). Suppose that there exists $\hat{x}_i \in X_i$ such that $A_i\hat{x}_i \leq A_i x_i$ for all $x_i \in \operatorname{conv} X_i$. There exists $x^* \in \mathbb{R}^d$ such that $x_i^* \in X_i$ for all i, $Ax^* \leq d$ and*

$$v^* \leq \sum_{i=1}^{n} f_i(x_i^*) \leq v^* + (m+1)\max_i \gamma(f_i).$$

Proof. By Lemma 3.B.1, there exists $\bar{x} \in \mathbb{R}^d$ and set $S \subset \{1, \ldots, n\}$ of cardinality at most $m+1$ such that $A\bar{x} \leq b$, $\bar{x}_i \in X_i$ for $i \notin S$ and $\bar{x}_i \in \operatorname{conv} X_i$ such that

$$v^* = \sum_{i \notin S} f_i(\bar{x}_i) + \sum_{i \in S} f_i^{**}(\bar{x}_i)$$

In contrast with the proof of Theorem 3.2.7, $\bar{x}$ is not necessarily primal feasible since X_i is not convex. For $i \in S$, let $x_i^* = \hat{x}_i \in X_i$, i.e. the vector such that $A_i\hat{x}_i \leq A_i x_i$ for all $x_i \in \operatorname{conv} X_i$. For $i \notin S$, let $x_i^* = \bar{x}_i$. We then have

$$Ax^* = \sum_{i \in S} Ax_i^* + \sum_{ii\not\in S} Ax_i^* \leq \sum_{i \in S} A\bar{x}_i + \sum_{ii\not\in S} A\bar{x}_i \leq b,$$

so that x^* is primal feasible, and

$$\begin{aligned}
v^* &\leq \sum_{i=1}^{n} f_i(x_i^*) \\
&= \sum_{i \notin S} f_i(x_i^*) + \sum_{i \in S} f_i(x_i^*) \\
&= v^* + \sum_{i \in S} (f_i(x_i^*) - f_i^{**}(\bar{x}_i)) \\
&\leq v^* + \sum_{i \in S} \sup_{x_i \in X_i} f_i(x_i) - \inf_{x_i \in X_i} f_i(x) \\
&\leq v^* + (m+1)\max_i \gamma(f_i).
\end{aligned}$$

The fact that $\inf_{x_i \in X_i} f_i(x_i) \leq f_i^{**}(\bar{x}_i)$ follows from the fact that the function $g(x_i) = \inf_{y \in X_i} f_i(y)$ is a convex minorant of f_i on X_i. □

3.A.4 Proof of Theorem 3.2.9

Theorem 3.2.9. *Let v^* be the dual optimal value as defined in (3.4) and $\gamma(f_i)$ be as defined in (3.13). Let $\theta \in \mathbb{R}^m$ be defined as*

$$\theta_k = (m+1)\max_i \left(\max_{x_i \in X_i} ((A_i)_{k,:})^\top x_i - \min_{x_i \in X_i} ((A_i)_{k,:})^\top x_i \right),$$

and suppose that there exists $\xi > 0$ and $\hat{x}_i \in \operatorname{conv} X_i$ for all i such that

$$\sum_{i=1}^{n} A_i\hat{x}_i \leq b - \theta - n\xi \mathbf{1}_m.$$

Then, there exists a solution $x^* \in \mathbb{R}^d$, such that $x_i^* \in X_i$, $Ax^* \leq b$, and

$$v^* \leq \sum_{i=1}^{n} f_i(x_i^*) \leq v^* + \left((m+1) + \frac{\|\theta\|_\infty}{\xi} \right) \max_i \gamma(f_i).$$

Proof. Let v_θ^* be the optimal value of (CoP_θ). Applying Lemma 3.B.1 to v_θ^*, there exist a set $S \subset \{1, \dots, n\}$ of cardinality at most $m+1$ and $\tilde{x} \in \mathbb{R}^d$ such that $A\tilde{x} \leq b - \theta$, $\tilde{x}_i \in X_i$ for $i \notin S$ and $\tilde{x}_i \in \text{conv}(X_i)$ for $i \in S$ such that

$$v_\theta^* = \sum_{i \in S} f_i^{**}(\tilde{x}_i) + \sum_{i \notin S} f_i(\tilde{x}_i)$$

Define $x^* \in \mathbb{R}^d$ as $x_i^* = \tilde{x}_i$ for $i \notin S$ and x_i^* be any point in X_i for $i \in S$. We then have

$$\begin{aligned} Ax^* &= \sum_{i \in S} A_i x_i^* + \sum_{i \notin S} A_i x_i^* = \sum_{i \in S} A_i(x_i^* - \tilde{x}_i) + \sum_{i \in S} A_i \tilde{x}_i + \sum_{i \notin S} A_i \tilde{x}_i \\ &= \sum_{i \in S} A_i(x_i^* - \tilde{x}_i) + \sum_{i=1}^n A_i \tilde{x}_i \\ &\leq \theta + (b - \theta) = b \end{aligned}$$

Thus x^* is primal feasible. As in Theorem 3.2.8, we can then show that

$$\sum_{i=1}^n f_i(x_i^*) \leq v_\theta^* + (m+1) \max_i \gamma(f_i).$$

It remains to show that

$$v_\theta^* \leq v^* + \frac{\|\theta\|_\infty}{\xi} \max_i \gamma(f_i).$$

We mirror the proof of [VEG+16, Theorem 3.3], except that we make no assumption on the uniqueness on the solution, or on the linearity of the objective functions. Let Ψ_θ be defined as Ψ in 3.3, except for the perturbed problem, and let $\lambda_\theta^* = \arg\max_{\lambda \geq 0} \Psi_\theta(\lambda)$. By [NO09, Lemma 1], $\hat{x}$ satisfies, for all $\lambda \geq 0$

$$\begin{aligned} \|\lambda_\theta^*\|_1 &\leq \frac{1}{n\xi} \left(\sum_{i=1}^n f_i^{**}(\hat{x}_i) - \Psi_\theta(\lambda) \right) \\ &\leq \frac{1}{n\xi} \left(\sum_{i=1}^n f_i^{**}(\hat{x}_i) - \sum_{i=1}^n \inf_{x_i \in X_i} (f_i(x) + \lambda^\top A_i x_i) - \lambda^\top (b - \theta) \right) \end{aligned}$$

Plugging $\lambda = 0$, and because $f_i^{**} \leq f_i$, this gives

$$\|\lambda_\theta^*\|_1 \leq \frac{1}{\xi} \max_i \gamma(f_i).$$

Now consider (CoP_θ) as an unperturbed program, and (CoP) as its perturbed version (perturbed by $+\theta$). Sensitivity analysis as in [BV04, Section 5.6.2] tells us that since strong duality holds,

$$v_\theta^* \leq v^* + (\lambda_\theta^*)^\top \theta \leq v^* + \|\lambda_\theta^*\|_1 \|\theta\|_\infty \leq v^* + \frac{\|\theta\|_\infty}{\xi} \max_i \gamma(f_i)$$

□

3.A.5 Proof of Lemma 3.3.3

Lemma 3.3.3. *Suppose* $\tilde{z} = \begin{pmatrix} \sum_{i=1}^n f_i^{**}(\tilde{x}_i) \\ A\tilde{x} \end{pmatrix}$ *is a solution of* (3.17). *Then* $\|\tilde{z} - z^*\|_+ = 0$ *and* $\sum_{i=1}^n f_i^{**}(\tilde{x}_i) = v^*$ *and* $A\tilde{x} \leq b$.

Proof. Let x^* be a solution to the bi-dual problem (CoP). We then have

$$z^* = \begin{pmatrix} v^* \\ b \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^n f_i^{**}(x_i^*) \\ Ax^* \end{pmatrix} + \begin{pmatrix} 0 \\ b - Ax^* \end{pmatrix} \in \sum_{i=1}^n \mathcal{C}_i + \mathbb{R}_+^{m+1}.$$

In particular, we have

$$\frac{1}{2}\left\| \begin{pmatrix} \sum_{i=1}^n f_i^{**}(x_i^*) \\ Ax^* \end{pmatrix} - z^* \right\|_+^2 = \frac{1}{2}\left\| \begin{pmatrix} 0 \\ Ax^* - b \end{pmatrix} \right\|_+^2 = 0,$$

since $Ax^* - b \leq 0$ by assumption. This implies that the minimum value of (3.17) is 0. In particular, if $\begin{pmatrix} \sum_{i=1}^n f_i^{**}(\tilde{x}_i) \\ A\tilde{x} \end{pmatrix}$ is a solution of (3.17), then $A\tilde{x} \leq b$ and $\sum_{i=1}^n f_i^{**}(\tilde{x}_i) \leq v^*$. Since v^* is optimal value of (CoP), we actually have $\sum_{i=1}^n f_i^{**}(\tilde{x}_i) = v^*$. □

3.A.6 Proof of Theorem 3.3.3

Theorem 3.3.3. *Suppose we run Algorithm 11 for $T \geq n-1$ iterations with input $\mathcal{A}_K$ and vector w^K/n, which outputs $\tilde{w}^{K,T}$ defined in (3.23), and that we build the final solution $w^{K,T}$ as in (3.24). Let*

$$h_0 = \left\| w^0 - w^K/n \right\|,$$

where w^0 is the starting point of Algorithm 12. Let $D_{\mathcal{A}_K}$ be the diameter of $\mathcal{A}_K$ and let $\delta(\mathcal{A}_K, \frac{w^K}{n})$ be as defined in Theorem 3.2.5. Assume that T is large enough so that

$$\frac{1}{n} > h_0 \exp\left(-\frac{T}{8} \left(\frac{\delta(\mathcal{A}_K, \frac{w^K}{n})}{D_{\mathcal{A}_K}^2} \right) \right). \tag{3.25}$$

It then holds that $|I_i| \geq 1$ and that $\sum_{l \in I_i} \alpha_l = 1$ for all i. Moreover,

$$\left| \sum_{i=1}^n \sum_{l \in I_i} \alpha_l f_i(y_i^{k_l}) - w_1^K \right| \leq n \left(\sqrt{n} \max_{i, y \in X_i} |f_i(y)| + 1 \right) h_0 \exp\left(-\frac{T\delta(\mathcal{A}_K, \frac{w^K}{n})}{8D_{\mathcal{A}_K}^2} \right),$$

$$\left\| \sum_{i=1}^n \sum_{l \in I_i} \alpha_l A_i y_i^{k_l} - w_{2:m+1}^K \right\| \leq n \left(\sqrt{n} \max_{i, y \in X_i} \|A_i y\| + 1 \right) h_0 \exp\left(-\frac{T\delta(\mathcal{A}_K, \frac{w^K}{n})}{8D_{\mathcal{A}_K}^2} \right).$$

In particular, $|I_i| > 1$ for at most $T - (n-1)$ indices i.

Proof. We start by showing that $|I_i| \geq 1$ for all i, which implies that α_l is well defined for all l. Indeed, the bound of Theorem 3.2.5 can be applied in particular to the last n coordinates of $\tilde{w}^{K,T} - \frac{w^K}{n}$. This gives, for all $i = 1, \ldots, n$,

$$\left(\sum_{l \in I_i} \beta_l - \frac{1}{n} \right)^2 \leq h_0^2 \left(1 - \frac{1}{4} \left(\frac{\delta(\mathcal{A}_K, \frac{w^K}{n})}{D_{\mathcal{A}_K}^2} \right) \right)^T \leq h_0^2 \exp\left(-\frac{T}{4} \left(\frac{\delta(\mathcal{A}_K, \frac{w^K}{n})}{D_{\mathcal{A}_K}^2} \right) \right).$$

If, for some i, I_i is empty, this implies that

$$\frac{1}{n} \leq h_0 \exp\left(-\frac{T}{8} \left(\frac{\delta(\mathcal{A}_K, \frac{w^K}{n})}{D_{\mathcal{A}_K}^2} \right) \right),$$

which contradicts our assumption. We now prove the inequalities. We start with the first one. The second one follows with the same argument. By the triangle inequality,

$$\begin{aligned}\left|\sum_{i=1}^{n}\sum_{l\in I_i}\alpha_l f_i(y_i^{i_l}) - w_1^K\right| \leq& \left|\sum_{i=1}^{n}\sum_{l\in I_i}\alpha_l f_i(y_i^{i_l}) - n\tilde{w}_1^{K,T}\right| \\ &+ n\left|\tilde{w}_1^{K,T} - \frac{w_1^K}{n}\right| \\ =& \left|\sum_{i=1}^{n}\sum_{l\in I_i}\alpha_l f_i(y_i^{i_l}) - n\sum_{i=1}^{n}\sum_{l\in I_i}\beta_l f_i(y_i^{i_l})\right| \\ &+ n\left|\tilde{w}_1^{K,T} - \frac{w_1^K}{n}\right|\end{aligned}$$

The second term in the sum can be bounded using the rate from Theorem 3.2.5, which gives

$$n\left|\tilde{w}_1^{K,T} - \frac{w_1^K}{n}\right| \leq nh_0 \exp\left(-\frac{T\delta(\mathcal{A}_K, \frac{w^K}{n})}{8D^2_{\mathcal{A}_K}}\right).$$

For the first term, we have

$$\begin{aligned}\left|\sum_{i=1}^{n}\sum_{l\in I_i}\alpha_l f_i(y_i^{i_l}) - n\sum_{i=1}^{n}\sum_{l\in I_i}\beta_l f_i(y_i^{i_l})\right| &= \left|\sum_{i=1}^{n}\left(\sum_{l\in I_i}\alpha_l f_i(y_i^{i_l}) - \sum_{l\in I_i} n\beta_l f_i(y_i^{i_l})\right)\right| \\ &= \left|\sum_{i=1}^{n}\sum_{l\in I_i}(\alpha_l - n\beta_l) f_i(y_i^{i_l})\right| \\ &\leq \max_{i,y\in X_i}|f_i(y)| \sum_{i=1}^{n}\sum_{l\in I_i}|\alpha_l - n\beta_l| \\ &= \max_{i,y\in X_i}|f_i(y)| \sum_{i=1}^{n}\sum_{l\in I_i}\left|\beta_l\left(\frac{1}{\sum_{l'\in I_i}\beta_{l'}} - n\right)\right| \\ &= \max_{i,y\in X_i}|f_i(y)| \sum_{i=1}^{n}\sum_{l\in I_i}\beta_l\left|\frac{1}{\sum_{l'\in I_i}\beta_{l'}} - n\right|\end{aligned}$$

Now, since $\sum_{l\in I_i}\beta_l > 0$ and the term in absolute value does not depend on $l \in I_i$,

$$\begin{aligned}\left|\sum_{i=1}^{n}\sum_{l\in I_i}\alpha_l f_i(y_i^{i_l}) - n\sum_{i=1}^{n}\sum_{l\in I_i}\beta_l f_i(y_i^{i_l})\right| &\leq \max_{i,y\in X_i}|f_i(y)| \sum_{i=1}^{n}\left|\sum_{l\in I_i}\beta_l\left(\frac{1}{\sum_{l'\in I_i}\beta_{l'}} - n\right)\right| \\ &= \max_{i,y\in X_i}|f_i(y)| \sum_{i=1}^{n} n\left|\frac{1}{n} - \sum_{l\in I_i}\beta_l\right| \\ &\leq \max_{i,y\in X_i}|f_i(y)| n\sqrt{n}\sqrt{\sum_{i=1}^{n}\left(\frac{1}{n} - \sum_{l\in I_i}\beta_l\right)^2} \\ &\leq n\sqrt{n}\max_{i,y\in X_i}|f_i(y)| h_0 \exp\left(-\frac{T\delta(\mathcal{A}_K, \frac{w^K}{n})}{8D^2_{\mathcal{A}_K}}\right).\end{aligned}$$

The second to last inequality comes from the fact that $\|x\|_1 \leq \sqrt{n}\|x\|_2$ for any $x \in \mathbb{R}^n$, while the last one comes from the rate given in Theorem 3.2.5. □

3.A.7 Proof of Theorem 3.3.6

Theorem 3.3.6. *Let v^* be the optimal value of (D) and (CoP) and suppose that Assumption 3.3.1 holds. Assume that we run Algorithm 13 for K iterations. Let $\bar{x} \in \mathbb{R}^d$ be the final solution defined in (3.33).*

If Algorithm 11 is used to trim the number of elements, then $\bar{x}$ satisfies

$$\sum_{i=1}^{n} f_i(\bar{x}_i) \leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + (m+1)\max_i \gamma(f_i),$$
$$\left\| \sum_{i=1}^{n} A_i \bar{x}_i - b \right\|_+ \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}}.$$

On the other hand, if Algorithm 12 for $T \geq n-1$ iterations is used to trim the number of elements, and under the condition of Theorem 3.3.3, $\bar{x}$ satisfies

$$\sum_{i=1}^{n} f_i(\bar{x}_i) \leq v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_f(n,T) + (T+1-n)\max_i \gamma(f_i),$$
$$\left\| \sum_{i=1}^{n} A_i \bar{x}_i - b \right\|_+ \leq \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_A(n,T).$$

Proof. Observe that

$$\begin{aligned} \sum_{i=1}^{n} f_i(\bar{x}_i) - v^* &= \left(\sum_{i=1}^{n} f_i(\bar{x}_i) - \bar{z}_1^K \right) + \left(\bar{z}_1^K - v^* \right) \\ \left\| \sum_{i=1}^{n} A_i \bar{x}_i - b \right\|_+ &\leq \left\| \sum_{i=1}^{n} A_i \bar{x}_i - \bar{z}_{2:m+1}^K \right\|_+ + \left\| \bar{z}_{2:m+1}^K - b \right\|_+ \end{aligned} \tag{3.46}$$

We can apply Lemma 3.3.4 to bound the last term of each equation. Moreover, we have

$$\begin{aligned} \sum_{i=1}^{n} f_i(\bar{x}_i) - \bar{z}_1^K &= \sum_{i=1}^{n} f_i(\bar{x}_i) - \sum_{i=1}^{n} \sum_{l \in I_i} \alpha_l f_i(y_i^{k_l}) \\ &= \sum_{i, |I_i|>1} \left(f_i(\bar{x}_i) - \sum_{l \in I_i} \alpha_l f_i(y_i^{k_l}) \right) \\ &\leq \sum_{i, |I_i|>1} \gamma(f_i) \\ &\leq q \max_i \gamma(f_i), \end{aligned}$$

and

$$\begin{aligned} \left\| \sum_{i=1}^{n} A_i \bar{x}_i - \bar{z}_{2:m+1}^K \right\|_+ &= \left\| \sum_{i=1}^{n} A_i \bar{x}_i - \sum_{i=1}^{n} \sum_{l \in I_i} \alpha_l A_i y_i^{k_l} \right\|_+ \\ &= \left\| \sum_{i, |I_i|>1} \left(A_i \hat{x}_i - \sum_{l \in I_i} \alpha_l A_i y_i^{k_l} \right) \right\|_+ \end{aligned}$$

By Assumption 3.3.1, $A_i \hat{x}_i \leq A_i x_i$ for all $x_i \in \operatorname{conv} X_i$. Thus, for all i such that $|I_i| > 1$,

$$A_i \hat{x}_i - \sum_{l \in I_i} \alpha_l A_i y_i^{k_l} \leq 0,$$

and hence

$$\left\| \sum_{i,|I_i|>1} \left(A_i \hat{x}_i - \sum_{l \in I_i} \alpha_l A_i y_i^{k_l} \right) \right\|_+ = 0.$$

□

3.A.8 Proof of Theorem 3.3.7

Theorem 3.3.7. *Let v^* be the optimal value of (D) and (CoP) and suppose that Assumption 3.3.2 holds. Suppose that we apply the method of Sections 3.3.1 and 3.3.2 to problem (CoP_θ) with θ defined in (3.34). Assume that we run Algorithm 13 for K iterations. Let $\bar{x} \in \mathbb{R}^d$ be the final solution defined in (3.35). Let q be the number of non-trivial convex combinations in the output of the Carathéodory algorithm, as defined in (3.27).*

If Algorithm 11 is used to trim the number of elements, then $\bar{x}$ satisfies

$$\sum_{i=1}^n f_i(\bar{x}_i) \le v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + \left(\frac{q(m+1)}{q+(m+1)} + \frac{\|\theta\|_\infty}{\xi} \right) \max_i \gamma(f_i),$$
$$\left\| \sum_{i=1}^n A_i \bar{x}_i - b \right\|_+ \le \frac{2D_{\mathcal{C}}}{\sqrt{K+1}}.$$

On the other hand, if Algorithm 12 for $T \ge n-1$ iterations (and $T \le n+m$) is used to trim the number of elements, and under the condition of Theorem 3.3.3, $\bar{x}$ satisfies

$$\sum_{i=1}^n f_i(\bar{x}_i) \le v^* + \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_f(n,T) + \left(\frac{q(m+1)}{q+(m+1)} + \frac{\|\theta\|_\infty}{\xi} \right) \max_i \gamma(f_i),$$
$$\left\| \sum_{i=1}^n A_i \bar{x}_i - b \right\|_+ \le \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + Err_A(n,T).$$

Proof. Let v_θ^* be the optimal value of (CoP_θ). Observe that

$$\begin{aligned} \sum_{i=1}^n f_i(\bar{x}_i) - v^* &= \left(\sum_{i=1}^n f_i(\bar{x}_i) - \bar{z}_1^K \right) + \left(\bar{z}_1^K - v_\theta^* \right) + (v_\theta^* - v^*) \\ \sum_{i=1}^n A_i \bar{x}_i - b &\le \left(\sum_{i=1}^n A_i \bar{x}_i - \bar{z}_{2:m+1}^K \right) - \theta + \left(\bar{z}_{2:m+1}^K - (b-\theta) \right) \\ &\le \left(\sum_{i=1}^n A_i \bar{x}_i - \bar{z}_{2:m+1}^K \right) - \theta + \mathbf{1}_m \left\| \bar{z}_{2:m+1}^K - (b-\theta) \right\|_+ \end{aligned} \tag{3.47}$$

Since we assumed that we applied the method to the perturbed problem (CoP_θ), we can apply Lemma 3.3.4 to get (in the case of Algorithm 11)

$$\left(\bar{z}_1^K - v_\theta^* \right) \le \frac{2D_{\mathcal{C}}}{\sqrt{K+1}},$$
$$\left\| \bar{z}_{2:m+1}^K - (b-\theta) \right\|_+ \le \frac{2D_{\mathcal{C}}}{\sqrt{K+1}},$$

or (in the case of Algorithm 12 with T iterations),

$$\left(\bar{z}_1^K - v_\theta^* \right) \le \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + \text{Err}_f(n,T),$$
$$\left\| \bar{z}_{2:m+1}^K - (b-\theta) \right\|_+ \le \frac{2D_{\mathcal{C}}}{\sqrt{K+1}} + \text{Err}_A(n,T).$$

Now,

$$\begin{aligned}\sum_{i=1}^{n} A_i\bar{x}_i - \bar{z}^K_{2:m+1} &= \sum_{i=1}^{n} A_i\bar{x}_i - \sum_{i=1}^{n}\sum_{l\in I_i} \alpha_l A_i y_i^{k_l} \\ &= \sum_{i,|I_i|>1} \left(A_i\bar{x}_i - \sum_{l\in I_i} \alpha_l A_i y_i^{k_l} \right) \\ &\leq \theta.\end{aligned}$$

This proves the inequalities on the feasibility. Moreover, we showed in Theorem 3.2.9 that

$$v^*_\theta \leq v^* + \frac{\|\theta\|_\infty}{\xi} \max_i \gamma(f_i).$$

Therefore, it only remains to show that

$$\sum_{i=1}^{n} f_i(\bar{x}_i) - \bar{z}^K_1 \leq \frac{q(m+1)}{q+(m+1)} \max_i \gamma(f_i).$$

Indeed, we have

$$\begin{aligned}\sum_{i=1}^{n} f_i(\bar{x}_i) - \bar{z}^K_1 &= \sum_{i=1}^{n} f_i(\bar{x}_i) - \sum_{i=1}^{n}\sum_{l\in I_i} \alpha_l f_i(y_i^{k_l}) \\ &= \sum_{i,|I_i|>1} \left(f_i(y_i^{k_{l_i}}) - \sum_{l\in I_i} \alpha_l f_i(y_i^{k_l}) \right) \\ &= \sum_{i,|I_i|>1} \left((1-\alpha_{l_i}) f_i(y_i^{k_{l_i}}) - \sum_{l\in I_i, l\neq l_i} \alpha_l f_i(y_i^{k_l}) \right) \\ &= \sum_{i,|I_i|>1} (1-\alpha_{l_i}) \left(f_i(y_i^{k_{l_i}}) - \sum_{l\in I_i, l\neq l_i} \frac{\alpha_l}{1-\alpha_{l_i}} f_i(y_i^{k_l}) \right) \\ &\leq \max_i \gamma(f_i) \sum_{i,|I_i|>1} (1-\alpha_{l_i}).\end{aligned}$$

We now show that

$$\sum_{i,|I_i|>1} (1-\alpha_{l_i}) \leq \frac{q(m+1)}{q+(m+1)}.$$

First observe that since $\alpha_{l_i} = \max_{l\in I_i} \alpha_l$ and $\sum_{l\in I_i} \alpha_l = 1$, we must have $\alpha_{l_i} \geq \frac{1}{|I_i|}$. Moreover we have $\sum_{i,|I_i|>1}(|I_i|-1) \leq (m+1)$. We therefore look for an upper bound on the program:

$$\begin{aligned}\max_{u_i\geq 2} \quad & \sum_{i=1}^{q} (1-\frac{1}{u_i}) \\ \text{s.t.} \quad & \sum_{i=1}^{q} (u_i - 1) \leq m+1.\end{aligned}$$

Since the program is concave and invariant under permutation, the solution must have all u_i's equal so that it can be rewritten

$$\begin{aligned}\max_{u\geq 2} \quad & q(1-\frac{1}{u}) \\ \text{s.t.} \quad & q(u-1) \leq m+1.\end{aligned}$$

The Lagrangian reads

$$L(u,\lambda,\mu) = q(1-\frac{1}{u}) + \lambda(u-2) + \mu\left(m+1-q(u-1)\right).$$

Maximization with respect to u gives

$$\frac{q}{u^2} + \lambda - \mu q = 0 \Rightarrow \frac{1}{u} = \sqrt{\mu - \lambda/q},$$

and the dual then reads

$$\min_{\mu\geq\lambda/q\geq 0} \quad q - q\sqrt{\mu-\lambda/q} - 2\lambda + \mu(m+1) + \mu q - q\sqrt{\mu-\lambda/q}.$$

By the change of variable $\gamma = \sqrt{\mu-\lambda/q}$, we get

$$\min_{\gamma,\lambda\geq 0} \quad q - 2q\gamma - 2\lambda + (\gamma^2 + \frac{\lambda}{q})(m+1) + (\gamma^2+\frac{\lambda}{q})q$$

First order optimality gives $\lambda = 0$ (since $q \leq m+1$) and $\gamma = \frac{q}{q+(m+1)}$. The corresponding optimal value is then $\frac{q(m+1)}{q+(m+1)}$. □

3.B Auxiliary lemma

Lemma 3.B.1. *Let v^* be the dual optimal value as defined in (3.4). There exists a set $S \subset \{1,\dots,n\}$ of cardinality at most $m+1$ and a vector $\bar{x} \in \mathbb{R}^d$, such that*

$$\begin{aligned}
&\bar{x}_i \in X_i \quad \textit{for } i \notin S,\\
&\bar{x}_i \in \operatorname{conv} X_i \quad \textit{for } i \in S,\\
&A\bar{x} \leq b,
\end{aligned}$$

and

$$v^* = \sum_{i\in S} f_i^{**}(\bar{x}_i) + \sum_{i\notin S} f_i(\bar{x}_i)$$

Proof. Since v^* is the optimal value of (CoP), there exists $\tilde{x} \in \mathbb{R}^d$ such that $A\tilde{x} \leq b$ and $v^* = \sum_{i=1}^n f_i^{**}(\tilde{x}_i)$. Let

$$\tilde{z} = \begin{pmatrix} \sum_{i=1}^n f_i^{**}(\tilde{x}_i) \\ b \end{pmatrix}$$

we have

$$\begin{aligned}
\tilde{z} &\in \sum_{i=1}^n \left\{ \begin{pmatrix} f_i^{**}(x_i) \\ A_i x_i \end{pmatrix} \mid x_i \in \operatorname{conv} X_i \right\} + \mathbb{R}_+^{m+1}\\
&= \sum_{i=1}^n \operatorname{conv}\left\{ \begin{pmatrix} f_i(x_i) \\ A_i x_i \end{pmatrix} \mid x_i \in X_i \right\} + \mathbb{R}_+^{m+1} \qquad \text{(Lemma 3.C.2).}
\end{aligned}$$

We can now apply the Shapley-Folkman theorem to show that there exists a set S of cardinality $m+1$ such that

$$\begin{aligned}
\tilde{z} &\in \sum_{i\notin S} \left\{ \begin{pmatrix} f_i(x_i) \\ A_i x_i \end{pmatrix} \mid x_i \in X_i \right\} + \sum_{i\in S} \operatorname{conv}\left\{ \begin{pmatrix} f_i(x_i) \\ A_i x_i \end{pmatrix} \mid x_i \in X_i \right\} + \mathbb{R}_+^{m+1}\\
&= \sum_{i\notin S} \left\{ \begin{pmatrix} f_i(x_i) \\ A_i x_i \end{pmatrix} \mid x_i \in X_i \right\} + \sum_{i\in S} \left\{ \begin{pmatrix} f_i^{**}(x_i) \\ A_i x_i \end{pmatrix} \mid x_i \in \operatorname{conv} X_i \right\} + \mathbb{R}_+^{m+1},
\end{aligned}$$

where we applied Lemma 3.C.2 again. In other words, there exists $\bar{x} \in \mathbb{R}^d$, $\bar{x}_i \in X_i$ for $i \notin S$ and $\bar{x}_i \in \operatorname{conv} X_i$ for $i \in S$, and $\delta_1 \geq 0, \delta_2 \in \mathbb{R}^m_+$ such that

$$v^* = \sum_{i=1}^n f_i^{**}(\tilde{x}_i) = \sum_{i \notin S} f_i(\bar{x}_i) + \sum_{i \in S} f_i^{**}(\bar{x}_i) + \delta_1,$$
$$b = A\bar{x} + \delta_2.$$

In particular, $\bar{x}$ is feasible, and since $f_i^{**} \leq f_i$, we must actually have $\delta_1 = 0$ and

$$\sum_{i=1}^n f_i^{**}(\tilde{x}_i) = \sum_{i \notin S} f_i(\bar{x}_i) + \sum_{i \in S} f_i^{**}(\bar{x}_i).$$

□

3.C Technical lemmas

Lemma 3.3.1. *Let $\tilde{x}_i \in \operatorname{conv} X_i$ such that $(f_i^{**}(\tilde{x}_i), A_i\tilde{x}_i)$ is an extreme point of $\mathcal{C}_i$. Then $\tilde{x}_i \in X_i$ and $f_i(\tilde{x}_i) = f_i^{**}(\tilde{x}_i)$.*

To prove the above lemma, we prove the following, stronger result. Recall that a function f is 1-coercive if

$$\lim_{\|x\| \to \infty} \frac{f(x)}{\|x\|} = \infty.$$

In particular, a function with compact domain is 1-coercive.

Lemma 3.C.1. *Suppose f is proper, closed and 1-coercive. Then*

$$\{(Ax, f^{**}(x)) \mid x \in \operatorname{dom} f^{**}\} \subset \operatorname{conv}\{(Ax, f(x)) \mid x \in \operatorname{dom} f\}.$$

Moreover, if for some $\tilde{x}$,

$$(A\tilde{x}, f^{**}(\tilde{x})) \in \operatorname{ext}\{(Ax, f^{**}(x)) \mid x \in \operatorname{dom} f^{**}\},$$

then $\tilde{x} \in \operatorname{dom} f$ and $f^{**}(\tilde{x}) = f(\tilde{x})$.

Proof. Let $z = (A\tilde{x}, f^{**}(\tilde{x})) \in \{(Ax, f^{**}(x)) \mid x \in \operatorname{dom} f^{**}\}$.

We have $(\tilde{x}, f^{**}(\tilde{x})) \in \operatorname{epi} f^{**} = \operatorname{conv} \operatorname{epi} f$ [HUL96, Theorem X.1.5.3]. Therefore, for some $K \in \mathbb{N}$, $\alpha \in \Delta_K$ and some $x_j \in \operatorname{dom} f$,

$$(\tilde{x}, f^{**}(\tilde{x})) = \sum_{j=1}^K \alpha_i(x_j, t_j) \quad \text{with} \quad f(x_j) \leq t_j \tag{3.48}$$

and thus

$$z = (A\tilde{x}, f^{**}(\tilde{x})) = \sum_{j=1}^K \alpha_j(Ax_j, t_j) \quad \text{with} \quad f(x_j) \leq t_j.$$

In particular, we have

$$f^{**}(\tilde{x}) = f^{**}(\sum_{j=1}^K \alpha_j x_j) \leq \sum_{j=1}^K \alpha_j f^{**}(x_j) \leq \sum_{j=1}^K \alpha_j f(x_j) \leq \sum_{j=1}^K \alpha_j t_j = f^{**}(\tilde{x}). \tag{3.49}$$

All the above inequalities are therefore equalities, and in particular,

$$(A\tilde{x}, f^{**}(\tilde{x})) = \sum_{j=1}^{K} \alpha_j (Ax_j, f(x_j)) \tag{3.50}$$

so that $z \in \operatorname{conv}\{(Ax, f(x)) \mid x \in \operatorname{dom} f\}$ which proves the first statement. Up to regrouping, we can assume that $(Ax_{j_1}, f^{**}(x_{j_1})) \neq (Ax_{j_2}, f^{**}(x_{j_2}))$ for all $j_1 \neq j_2$ in the sum.

Now suppose that z is an extreme point of the set $\{(Ax, f^{**}(x)) \mid x \in \operatorname{dom} f^{**}\}$. Recall that $z = (A\tilde{x}, f^{**}(\tilde{x}))$ and observe that (3.48) and (3.49) tell us that

$$z = (A\tilde{x}, f^{**}(\tilde{x})) = \sum_{j=1}^{K} \alpha_j (Ax_j, f^{**}(x_j)).$$

Since z is an extreme point, the above convex combination must be trivial, i.e. we must have $\alpha_j = 0$ for all but one index. Suppose without loss of generality that $\alpha_1 = 1$ and that $\alpha_j = 0$ for $j > 1$. We then have, by (3.48), $\tilde{x} = x_1 \in \operatorname{dom} f$. We also have, by (3.50), $f^{**}(\tilde{x}) = f(x_1) = f(\tilde{x})$. □

Lemma 3.C.2. *Suppose f is proper, closed, and 1-coercive. Then*

$$\left\{ \begin{pmatrix} f^{**}(x) \\ Ax \end{pmatrix} \mid x \in \operatorname{dom} f^{**} \right\} + \mathbb{R}^{m+1}_{+} = \operatorname{conv} \left\{ \begin{pmatrix} f(x) \\ Ax \end{pmatrix} \mid x \in \operatorname{dom} f \right\} + \mathbb{R}^{m+1}_{+}$$

Proof. Let $z \in \left\{ \begin{pmatrix} f^{**}(x) \\ Ax \end{pmatrix} \mid x \in \operatorname{dom} f^{**} \right\} + \mathbb{R}^{m+1}_{+}$ so that

$$z = \begin{pmatrix} f^{**}(\tilde{x}) \\ A\tilde{x} \end{pmatrix} + \begin{pmatrix} \delta_1 \\ \delta_2 \end{pmatrix},$$

with $\delta_1 \in \mathbb{R}_+, \delta_2 \in \mathbb{R}^+_m$. It is then clear that

$$\begin{pmatrix} \tilde{x} \\ z \end{pmatrix} \in \{(x, r_0, r) \mid f^{**}(x) \leq r_0, Ax \leq r\}.$$

Using [LR01, Corollary A.6], this implies actually that

$$\begin{aligned} \begin{pmatrix} \tilde{x} \\ z \end{pmatrix} &\in \operatorname{cl} \operatorname{conv} \{(x, r_0, r) \mid f(x) \leq r_0, Ax \leq r\} \\ &= \operatorname{conv} \{(x, r_0, r) \mid f(x) \leq r_0, Ax \leq r\} \end{aligned}$$

where the equality is due to Lemma 3.C.3. In particular, we have

$$\begin{aligned} z &\in \operatorname{conv} \{(r_0, r) \mid f(x) \leq r_0, Ax \leq r \text{ for some } x\} \\ &= \operatorname{conv} \left\{ \left\{ \begin{pmatrix} f(x) \\ Ax \end{pmatrix} \mid x \in \operatorname{dom} f \right\} + \mathbb{R}^{m+1}_{+} \right\} \\ &= \operatorname{conv} \left\{ \begin{pmatrix} f(x) \\ Ax \end{pmatrix} \mid x \in \operatorname{dom} f \right\} + \mathbb{R}^{m+1}_{+}. \end{aligned}$$

Now suppose

$$\begin{aligned} z = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} &\in \operatorname{conv} \left\{ \begin{pmatrix} f(x) \\ Ax \end{pmatrix} \mid x \in \operatorname{dom} f \right\} + \mathbb{R}^{m+1}_{+} \\ &= \operatorname{conv} \{(r_0, r) \mid f(x) \leq r_0, Ax \leq r \text{ for some } x\}. \end{aligned}$$

Then there exists $\alpha \geq 0$ such that $1^\top \alpha = 1$ and

$$(z_1, z_2) = \sum_k \alpha_k (f(x_k), Ax_k) + (\delta_1, \delta_2)$$

for some $\delta_1 \in \mathbb{R}_+, \delta_2 \in \mathbb{R}^m_+$ and some $x_k \in \operatorname{dom} f$. Let us set $y = \sum_k \alpha_k x_k$. Then

$$\begin{aligned}\begin{pmatrix} y \\ z \end{pmatrix} &\in \operatorname{conv} \{(x, r_0, r) \mid f(x) \leq r_0, Ax \leq r\} \\ &= \operatorname{cl} \operatorname{conv} \{(x, r_0, r) \mid f(x) \leq r_0, Ax \leq r\} \\ &= \{(x, r_0, r \mid f^{**}(x) \leq r_0, Ax \leq r\}\end{aligned}$$

where the first equality is by Lemma 3.C.3 and the second is again from [LR01, Corollary A.6]. This ends the proof. □

Lemma 3.C.3. *Suppose f is proper, closed and 1-coercive. Then*

$$\operatorname{conv} \{(x, r_0, r) \mid f(x) \leq r_0, Ax \leq r\}$$

is closed.

Proof. Suppose $(x_k, r_{0,k}, r_k) \in \operatorname{conv} \{(x, r_0, r) \mid f(x) \leq r_0, Ax \leq r\}$ converges to $(\tilde{x}, \tilde{r}_0, \tilde{r})$. We want to show that $(\tilde{x}, \tilde{r}_0, \tilde{r}) \in \operatorname{conv} \{(x, r_0, r) \mid f(x) \leq r_0, Ax \leq r\}$. For all k, there exists $\alpha_k \geq 0$ such that $1^\top \alpha_k = 1$ and

$$(x_k, r_{0,k}, r_k) = \sum_j \alpha_{kj} (x_{kj}, r_{0,kj}, r_{kj})$$

with $(x_{kj}, r_{0,kj}) \in \operatorname{epi} f$ and $Ax_{kj} \leq r_{kj}$. In particular, this implies that $(x_k, r_{0,k}) \in \operatorname{conv} \operatorname{epi} f$ for all k. Since $\operatorname{conv} \operatorname{epi} f$ is closed [HUL96, Lemma X.1.5.3], this implies that $(\tilde{x}, \tilde{r}_0) \in \operatorname{conv} \operatorname{epi} f$. Therefore there exists $(x_p, r_{0,p}) \in \operatorname{epi} f$ such that

$$(\tilde{x}, \tilde{r}_0) = \sum_p \beta_p (x_p, r_{0,p})$$

with $\beta \geq 0$, $1^\top \beta = 1$. Moreover, we have

$$Ax_k = \sum_k \alpha_{kj} Ax_{kj} \leq \sum_k \alpha_{kj} r_{kj} = r_k.$$

Taking limits, this implies that $A\tilde{x} \leq \tilde{r}$. Define $r_p = Ax_p + (\tilde{r} - A\tilde{x})$. Then $Ax_p \leq r_p$ and $\sum_p \beta_p r_p = A \sum_p \beta_p x_p + (\tilde{r} - A\tilde{x}) = \tilde{r}$. Thus

$$(\tilde{x}, \tilde{r}_0, \tilde{r}) \in \operatorname{conv} \{(x, r_0, r) \mid f(x) \leq r_0, Ax \leq r\}$$

. □

3.D Experimental details

In this section we give details on the data generation of Section 3.5. We write $U(c, d)$ as the uniform distribution in interval $[c, d]$. At each time step t, the demand D_t is generated according to $D_t \sim U(100, 300)$. For each unit i, we then set the maximum and minimum generation parameter $\overline{g}^i$ and $\underline{g}^i$ as

$$\begin{aligned}p^i &\sim \frac{1}{n} \times U(100, 300) \\ \overline{g}^i &= 2p^i \\ \underline{g}^i &= 0.5p^i.\end{aligned}$$

We then set, for all unit i,

$$\beta_i \sim U(1,20), \quad \gamma_i \sim U(3,5), \quad \omega_i \sim U(30,50),$$
$$c_{01}^i = \frac{\sum_j \beta_j (p^j)^2 + \gamma_j p^j + \omega_j}{2n}, \quad c_{10}^i = \frac{c_{01}^i}{4}.$$

We found that choosing the parameters this way allows for interesting settings where the choice of when to turn on/off units of production, and of how much power should be generated, is non-trivial.

Chapter

4 Iteratively Reweighted Least Squares for Phase Unwrapping

Chapter content

4.1 Introduction . . . 83
4.1.1 Related Work . . . 84
4.1.2 Contributions . . . 85
4.2 Notations and definitions . . . 86
4.3 Model . . . 87
4.4 Algorithm . . . 88
4.4.1 Iteratively reweighted least squares . . . 88
4.4.2 Conjugate gradient . . . 93
4.4.3 Choosing the preconditioner . . . 94
4.5 Experiments . . . 99
4.5.1 Real Data Acquisition and Synthetic Data Generation . . . 100
4.5.2 Weight computation . . . 101
4.5.3 Number of CG iterations and stopping criterion . . . 101
4.5.4 Code details . . . 102
4.5.5 Results . . . 104
Appendix 4.A Matrix visualization . . . 107
Appendix 4.B Proof of Proposition 4.4.3 . . . 107
Appendix 4.C Additional experiments . . . 109
4.C.1 Simulated images . . . 109
4.C.2 Real images with Goldstein filtering . . . 114

4.1 Introduction

We focus on the problem of estimating the phase of an input signal in scenarios where, because of the system's physical limitations, the phase is only available modulo 2π. This is the case, for instance, in interferometric synthetic aperture radar (InSAR) imaging [Gra74, ZG86, GP98, RHJ+00], in magnetic resonance imaging (MRI) [Lau73, HR92] or in optical interferometry [PJJ94]. For example, in SAR imaging, one measures the phase shift between a reference wave and a reflected wave, in order to compute terrain elevation. Since phase is periodic, the resulting signal is contained between 0 and 2π. Whenever the true phase exceeds 2π, the measured signal 'wraps' to 0, resulting in a discontinuous observation. Phase unwrapping describes the process of recovering the original signal from its observation modulo 2π.

Formally, 2D phase unwrapping is the problem of recovering the true phase $\boldsymbol{U}$, given an observed wrapped phase $\boldsymbol{X}$, such that

$$\boldsymbol{U} = \boldsymbol{X} + 2\pi \boldsymbol{K},$$

where $\boldsymbol{K}$ is an integer-valued matrix. The problem is ill-posed, as any shift of 2π in the matrix $\boldsymbol{U}$ results in the same wrapped matrix $\boldsymbol{X}$. Most phase unwrapping algorithms are based on the so-called Itoh condition [Ito82], which states that the absolute difference between neighboring pixels (i.e. elements of the matrix $\boldsymbol{U}$) is no more than π. When that condition is satisfied, unwrapping can be easily determined (up to a constant) by a simple integration procedure. However, in practice Itoh's condition is often violated. This can be due for example to the presence of noise in the input images, or to large discontinuities in the input signal. Phase unwrapping then becomes a much more complex task, which has been the focus of a long line of research [YLY+19].

4.1.1 Related Work

Most phase unwrapping algorithms fall within one of the following categories: path-following or optimization-based methods. We next give an overview of these two approaches. We refer the reader to the literature review in [YLY+19] for a more complete description of modern phase unwrapping techniques.

4.1.1.1 Path following algorithms.

As previously mentioned, when the Itoh condition is satisfied, unwrapping can be exactly performed via a simple integration procedure, and the obtained result does not depend on the integration path. Integration can still be performed when the Itoh condition is not satisfied, but in that case the choice of the integration path significantly impacts the quality of the unwrapping. Path following methods are methods aimed at choosing good integration path. Among them are quality-guided methods, which choose the path based on a quality map of the input image, using this map to minimize phase unwrapping error in regions where the quality is high [Fly96, ZTZC10, ZHZ+11, Jia16]. Popular examples of quality maps include correlation maps, phase derivative variance maps, or priority maps.

Another important class of path following algorithms are methods based on balancing residues in the image. Residues are the results of the loop integration of every 2×2 neighboring pixel block in the input image and are a key concept for many phase unwrapping techniques. One popular example of a residue-based method is the branch cut algorithm, which first connects nearby residues of opposite polarities, and then integrates without crossing the connections [GZW88, AG18].

4.1.1.2 L^p-norm minimization methods.

L^p-norm minimization methods take a global approach to the phase unwrapping problem. Based on Itoh's condition, those methods seek to minimize the number of times the unwrapped phase differences fail to match the wrapped differences of the input image. In other words, the residual image formed from observed and reconstructed gradients should be sparse. This leads to minimization objectives of the form

$$\min_{U} \sum_{i,j} |\boldsymbol{U}_{i+1,j} - \boldsymbol{U}_{i,j} - \boldsymbol{G}^{\boldsymbol{v}}_{i,j}|^p + \sum_{i,j} |\boldsymbol{U}_{i,j+1} - \boldsymbol{U}_{i,j} - \boldsymbol{G}^{\boldsymbol{h}}_{i,j}|^p,$$

where $\boldsymbol{G}^{\boldsymbol{v}}_{ij}$ and $\boldsymbol{G}^{\boldsymbol{h}}_{ij}$ denote the vertical and horizontal phase differences, respectively, of the input image $\boldsymbol{X}$.

When $p = 2$, the objective is quadratic, and efficient algorithms like preconditioned conjugate methods can be designed [GR94, GP98]. Unfortunately, the output of the L^2-norm problem is

much worse compared to the ones of L^p-norm problems with $p \leq 1$, as it tends to smooth out areas of large discontinuities [GR96, FMGP+07].

It is widely accepted that $p = 0$ yields the best solution, since the problem is then exactly minimizing the number of pixels where the gradients do not match [YLB11]. Unfortunately, the objective function is not convex and the minimization problem is NP-hard [CZ00]. Yet several methods have been proposed to approximately solve the non-convex problem [GR96, GP98, CZ00, BDV07, YLB11]. In particular, [GR96] propose a general algorithm to solve the L^p-norm problem for any $p < 2$ by iteratively solving weighted linear systems. The proposed algorithm resembles an iteratively reweighted least squares approach when $p = 1$, although the minimization is not exact with respect to the introduced weights, and thus no convergence proof is provided.

On the other hand, when $p = 1$ the resulting problem is known to be a good approximation of the L^0-norm problem, and in that case the objective function is convex, so efficient algorithms can be developed [Fly97]. In particular, the L^1-norm problem can be cast as a minimum cost flow (MCF) which can be solved using graph solvers [Cos98].

Finally, statistics-based methods model the phase unwrapping problem as a maximum a posteriori (MAP) estimation problem. Different assumptions on the underlying probability distributions have been proposed, and those directly impact the hardness of the resulting optimization problem [NPD00, CZ01, DL02, BDV07]. The SNAPHU method [CZ01] is one such approach, and is one of the methods of choice for phase unwrapping, as evidenced by its use in many InSAR software packages [SNA, HBSA12]. Unfortunately, statistics-based methods must solve complicated optimization problems in large dimension, and thus often exhibit long running times.

4.1.2 Contributions

The main motivation behind this chapter is that despite this long line of research, the running time of existing methods can be prohibitive on very large images. For example, it is common in satellite imagery to treat images of a few thousand pixels by a few thousand pixels and as we will see in the experimental part of this chapter, existing methods take at least a few minutes on such images. Our work significantly improves this running time, developing an algorithm with strong theoretical guarantees, which can be implemented on GPUs.

We focus on the L^1-norm problem. Our proposed method rewrites the L^1-norm as a weighted L^2-norm, introducing data-dependent weights [BR96, DDDM04, DDFG10, BJM+12, MBP+14, FPRW16]. This leads to the so-called iteratively reweighted least squares (IRLS) algorithm, where the minimization is performed alternatively with respect to the weights and the objective image $\boldsymbol{U}$. The minimization with respect to the weights has a closed form solution, while the minimization with respect to $\boldsymbol{U}$ comes down to solving a linear system. This linear system is high-dimensional but the matrix-vector product associated is cheap to compute in practice, suggesting the use of iterative solvers. We thus implement the conjugate gradient (CG) method [NW06] to approximately solve the system, and we provide a carefully designed preconditioner to accelerate the CG iterations. We prove that despite the approximate solutions obtained through the CG algorithm, the iterates of the resulting IRLS algorithm satisfy a sublinear rate of convergence in function value.

We implement our fully GPU-compatible algorithm in Python. We show over extensive experiments on both simulated and real data that our method is competitive with state of the art phase unwrapping techniques in terms of image quality, and reduces the average running time by about one order of magnitude.

The chapter is organized as follows. We first introduce useful mathematical notations and properties. We detail the L^1-norm problem in Section 4.3. In Section 4.4 we describe the IRLS algorithm and detail the conjugate gradient method to solve the linear system that appears in

IRLS iterations, as this is the main bottleneck of our method. We focus in particular on an efficient preconditioning of the linear system. We also prove sublinear convergence of the IRLS algorithm in terms of function value. In Section 4.5 we compare our method against other phase unwrapping methods for different image sizes, both in terms of computing time and output quality.

4.2 Notations and definitions

We write $\|\cdot\|$ and $\|\cdot\|_{\ell_1}$ for the Euclidean and ℓ_1 norm of a vector, respectively. For integers $n, m \in \mathbb{N}$, we consider $\mathbb{R}^{n\times m}$, the set of real-valued matrices of size n by m, equipped with the standard inner product $\langle \cdot, \cdot \rangle : \mathbb{R}^{n\times m} \times \mathbb{R}^{n\times m} \to \mathbb{R}$, as well as the Frobenius norm $\|\cdot\|_F$ and the ℓ_1 norm $\|\cdot\|_{\ell_1}$. Formally, for $X, Y \in \mathbb{R}^{n\times m}$, we have

$$\begin{aligned}
\langle X, Y \rangle &= \mathbf{Tr}(X^\top Y) = \sum_{\substack{1\le i\le n \\ 1\le j\le m}} X_{i,j} Y_{i,j} \\
\|X\|_F &= \sqrt{\langle X, X\rangle} = \sqrt{\sum_{\substack{1\le i\le n \\ 1\le j\le m}} (X_{i,j})^2} \\
\|X\|_{\ell_1} &= \sum_{\substack{1\le i\le n \\ 1\le j\le m}} |X_{i,j}|.
\end{aligned}$$

We simply use $\|\cdot\|$ for the Frobenius norm. We write $\mathbf{I}_n$ as the identity matrix of dimension n, $\mathbf{1}_n$ and $\mathbf{1}_{n\times m}$ as the vector and matrix whose entries are all equal to 1. We define $\mathbf{0}_n$ and $\mathbf{0}_{n\times m}$ similarly. We drop the integer index when the dimension is clear from context.

The Hadamard (or component-wise) product of $\boldsymbol{X}$ and $\boldsymbol{Y}$, written $\boldsymbol{X} \odot \boldsymbol{Y}$, is the matrix of the same dimension as $\boldsymbol{X}$ and $\boldsymbol{Y}$ with elements given by

$$(\boldsymbol{X} \odot \boldsymbol{Y})_{i,j} = \boldsymbol{X}_{i,j}\boldsymbol{Y}_{i,j}, \quad 1 \le i \le n, \quad 1 \le j \le m.$$

We also write $\frac{1}{\boldsymbol{X}}$ as the matrix whose entries are equal to the inverse of the entries in $\boldsymbol{X}$, provided none of those are equal to zero.

The Kronecker product of two matrices $\boldsymbol{X} \in \mathbb{R}^{n_1\times m_1}$ and $\boldsymbol{Y} \in \mathbb{R}^{n_2\times m_2}$, denoted as $\boldsymbol{X} \otimes \boldsymbol{Y}$, is the matrix of size $n_1 n_2 \times m_1 m_2$ given by

$$\boldsymbol{X} \otimes \boldsymbol{Y} = \begin{pmatrix} \boldsymbol{X}_{1,1}\boldsymbol{Y} & \cdots & \boldsymbol{X}_{1,m_1}\boldsymbol{Y} \\ \vdots & \ddots & \vdots \\ \boldsymbol{X}_{n_1,1}\boldsymbol{Y} & \cdots & \boldsymbol{X}_{n_1,m_1}\boldsymbol{Y} \end{pmatrix}$$

For a matrix $\boldsymbol{X} \in \mathbb{R}^{n\times m}$, the vectorization of $\boldsymbol{X}$ is the vector $\mathbf{vec}(\boldsymbol{X}) \in \mathbb{R}^{nm}$ defined as

$$\mathbf{vec}(\boldsymbol{X}) = (\boldsymbol{X}_{1,1}, \boldsymbol{X}_{2,1}, \ldots, \boldsymbol{X}_{n,1}, \boldsymbol{X}_{1,2}, \ldots, \boldsymbol{X}_{n,2}, \ldots, \boldsymbol{X}_{1,m}, \ldots \boldsymbol{X}_{n,m})^\top.$$

In particular, we will make use of the following property relating matrix-matrix products to matrix-vector products,

$$\mathbf{vec}(\boldsymbol{X}\boldsymbol{Y}\boldsymbol{Z}) = (\boldsymbol{Z}^\top \otimes \boldsymbol{X})\mathbf{vec}(\boldsymbol{Y}). \tag{4.1}$$

For a vector $x \in \mathbb{R}^n$, we define $\mathbf{diag}(x)$ as the diagonal matrix of dimension $n \times n$ whose diagonal is equal to the vector x. For a matrix $\boldsymbol{X} \in \mathbb{R}^{n\times n}$, we define $\mathbf{diag}(\boldsymbol{X})$ as the vector of size n composed of the diagonal entries of $\boldsymbol{X}$. We also define weighted norms. Formally, for a matrix

$\boldsymbol{W} \in \mathbb{R}^{n\times m}$ with strictly positive entries, the weighted Frobenius norm and weighted ℓ_1 norm are respectively written

$$\|\boldsymbol{X}\|_{F,\boldsymbol{W}} := \sqrt{\sum_{\substack{1\le i\le n\\ 1\le j\le m}} \boldsymbol{W}_{i,j}(\boldsymbol{X}_{i,j})^2},$$
$$\|\boldsymbol{X}\|_{\ell_1(\boldsymbol{W})} := ||\boldsymbol{W}\odot\boldsymbol{X}||_{\ell_1}.$$

4.3 Model

In this section we define the phase unwrapping problem, together with some useful notation. We assume that the input is a wrapped phase image $\boldsymbol{X} \in \mathbb{R}^{N\times M}$, namely that $\boldsymbol{X}_{i,j} \in [0, 2\pi)$ for all $1 \le i \le N$ and all $1 \le j \le M$. The wrapped phase gradients are the matrices $\boldsymbol{G^v} \in \mathbb{R}^{(N-1)\times M}$ and $\boldsymbol{G^h} \in \mathbb{R}^{N\times(M-1)}$ defined as

$$\begin{aligned}
\boldsymbol{G}^{\boldsymbol{v}}_{i,j} &= \boldsymbol{X}_{i+1,j} - \boldsymbol{X}_{i,j} \mod 2\pi \quad 1 \le i \le N-1, \quad 1 \le j \le M,\\
\boldsymbol{G}^{\boldsymbol{h}}_{i,j} &= \boldsymbol{X}_{i,j+1} - \boldsymbol{X}_{i,j} \mod 2\pi \quad 1 \le i \le N, \quad 1 \le j \le M-1.
\end{aligned}$$

Following [Cos98], we solve the L^1-norm phase unwrapping problem defined as

$$\min_{\boldsymbol{U}\in\mathbb{R}^{N\times M}} \left\{ \begin{array}{l} \displaystyle\sum_{\substack{1\le i\le N-1\\ 1\le j\le M}} \boldsymbol{C}^{\boldsymbol{v}}_{i,j} \left|\boldsymbol{U}_{i+1,j} - \boldsymbol{U}_{i,j} - \boldsymbol{G}^{\boldsymbol{v}}_{i,j}\right| \\ + \displaystyle\sum_{\substack{1\le i\le N\\ 1\le j\le M-1}} \boldsymbol{C}^{\boldsymbol{h}}_{i,j} \left|\boldsymbol{U}_{i,j+1} - \boldsymbol{U}_{i,j} - \boldsymbol{G}^{\boldsymbol{h}}_{i,j}\right| \end{array} \right\}, \tag{4.2}$$

in the variable $\boldsymbol{U} \in \mathbb{R}^{N\times M}$, where $\boldsymbol{C^v} \in \mathbb{R}^{(N-1)\times M}, \boldsymbol{C^h} \in \mathbb{R}^{N\times(M-1)}$ are positive, user-defined weights. When problem (4.2) is solved using linear programming techniques, the resulting unwrapped image will be congruent to the input image, even if congruency is not explicity enforced. This is due to the total unimodularity of the constraint matrix introduced when casting (4.2) as a linear program [AMO88, Cos98]. However, because of our algorithm and the modifications we will make to problem (4.2), there is no guarantee that our solution will be congruent to the input image. We argue that this is an acceptable approach for several reasons. First, most input images $\boldsymbol{X}$ are noisy, and even when a denoising step is applied prior to the unwrapping process, unwrapped solutions might be more accurate when no congruency is enforced. Second, as we will see in the experimental section, our proposed algorithm yields close-to congruent outputs when the input image is noiseless. Finally, in terms of numerical efficiency, removing the congruency assumption allows for much faster algorithms in practice.

Problem (4.2) is ambiguous, in the sense that any solution yields the same objective function value under additive shift. We therefore define the following constraint set

$$\mathbf{C_0} = \left\{ \boldsymbol{U} \in \mathbb{R}^{N\times M} \mid \sum_{i,j} \boldsymbol{U}_{i,j} = 0 \right\}$$

and consider the problem

$$\min_{\boldsymbol{U}\in\mathbf{C_0}} \left\{ \sum_{\substack{1\le i\le N-1\\ 1\le j\le M}} \boldsymbol{C}^{\boldsymbol{v}}_{i,j} \left|\boldsymbol{U}_{i+1,j} - \boldsymbol{U}_{i,j} - \boldsymbol{G}^{\boldsymbol{v}}_{i,j}\right| + \sum_{\substack{1\le i\le N\\ 1\le j\le M-1}} \boldsymbol{C}^{\boldsymbol{h}}_{i,j} \left|\boldsymbol{U}_{i,j+1} - \boldsymbol{U}_{i,j} - \boldsymbol{G}^{\boldsymbol{h}}_{i,j}\right| \right\}, \tag{4.3}$$

We rewrite problem (4.3) in matrix form, defining matrices $\boldsymbol{S} \in \mathbb{R}^{(N-1)\times N}$ and $\boldsymbol{T} \in \mathbb{R}^{M\times(M-1)}$ as

$$\boldsymbol{S}_{i,j} = \begin{cases} 1 & \text{if } j = i+1 \\ -1 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}, \quad 1 \le i \le N-1, \quad 1 \le j \le N,$$

$$\boldsymbol{T}_{i,j} = \begin{cases} 1 & \text{if } i = j+1 \\ -1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}, \quad 1 \le i \le M, \quad 1 \le j \le M-1.$$

Problem (4.3) can then be cast as

$$\min_{\boldsymbol{U}\in\mathbf{C_0}} \left\{ \|\boldsymbol{C^v} \odot (\boldsymbol{SU} - \boldsymbol{G^v})\|_{\ell_1} + \left\|\boldsymbol{C^h} \odot (\boldsymbol{UT} - \boldsymbol{G^h})\right\|_{\ell_1} \right\}. \tag{4.4}$$

4.4 Algorithm

This section details the algorithmic framework for solving (4.4). We first rewrite (4.4) and introduce quadratic penalties. We cast the resulting problem as a sequence of weighted least squares problems whose solutions converge to the solution of the original problem. Each least squares problem is solved using the conjugate gradient method with an appropriate preconditioner. We detail each of these steps below, following the algorithmic framework in [FPRW16]. Note that reweighting techniques similar in spirit (but not exactly the same) to the ones presented in this chapter have been proposed for the phase unwrapping problem [GR96], although no connection to the iteratively reweighted least squares approach was made and no convergence theory was provided.

4.4.1 Iteratively reweighted least squares

We let $\boldsymbol{V^v} \in \mathbb{R}^{(N-1)\times M}$ and $\boldsymbol{V^h} \in \mathbb{R}^{N\times(M-1)}$ be slack variables and rewrite problem (4.4) as

$$\min_{\substack{\boldsymbol{U}\in\mathbf{C_0},\\ \boldsymbol{V^v},\boldsymbol{V^h}}} F(\boldsymbol{U},\boldsymbol{V^v},\boldsymbol{V^h}) := \left\{ \begin{array}{l} \|\boldsymbol{V^v}\|_{\ell_1(\boldsymbol{C^v})} + \left\|\boldsymbol{V^h}\right\|_{\ell_1(\boldsymbol{C^h})} \\ + \dfrac{1}{2\tau}\|\boldsymbol{SU} - \boldsymbol{G^v} - \boldsymbol{V^v}\|^2 + \dfrac{1}{2\tau}\left\|\boldsymbol{UT} - \boldsymbol{G^h} - \boldsymbol{V^h}\right\|^2 \end{array} \right\}, \tag{4.5}$$

for some penalization term $\tau > 0$. Problem (4.5) is a quadratic penalty method for (4.4), and iteratively solving (4.5) for decreasing values of τ converging to 0 yields a sequence of solutions whose limit points are exact minimizers of (4.4) [Ber14]. We will however keep the value of τ constant to reduce the computing time. This is also motivated by the fact that images encountered in phase unwrapping problems are often noisy, and quadratic penalties might be beneficial in that regard. Moreover, substituting hard constraints for quadratic penalties has numerous numerical advantages, as pointed out in [FPRW16], and we shall explore them in Section 4.4.2. Before detailing the iteratively reweighted least squares (IRLS) method, we prove existence and uniqueness of the solution of (4.5).

Theorem 4.4.1. *Problem (4.5) has a unique solution.*

Proof. Define the vectorized function f as

$$f(u, v^v, v^h) := \|v^v\|_{\ell_1(c^v)} + \left\|v^h\right\|_{\ell_1(c^h)} + \frac{1}{2\tau}\left\|\boldsymbol{B}\begin{pmatrix} u \\ v^v \\ v^h \end{pmatrix} - \begin{pmatrix} g^v \\ g^h \end{pmatrix}\right\|^2.$$

with $g^v = \mathbf{vec}(\boldsymbol{G^v})$, $g^h = \mathbf{vec}(\boldsymbol{G^h})$, and

$$\boldsymbol{B} = \begin{pmatrix} \mathbf{I}_M \otimes \boldsymbol{S} & -\mathbf{I}_{(N-1)M} & \mathbf{0} \\ \boldsymbol{T}^\top \otimes \mathbf{I}_N & \mathbf{0} & -\mathbf{I}_{N(M-1)} \end{pmatrix}$$

It follows that $F(\boldsymbol{U}, \boldsymbol{V^v}, \boldsymbol{V^h}) = f(\mathbf{vec}(\boldsymbol{U}), \mathbf{vec}(\boldsymbol{V^v}), \mathbf{vec}(\boldsymbol{V^h}))$. Moreover define

$$\mathbf{c_0} = \{x \in \mathbb{R}^{NM} \mid \sum_i x_i = 0\}.$$

It is clear that $\boldsymbol{U} \in \mathbf{C_0} \iff \mathbf{vec}(\boldsymbol{U}) \in \mathbf{c_0}$. Therefore problem (4.5) is equivalent to

$$\min_{u, v^v, v^h} f(u, v^v, v^h) + \delta_{\mathbf{c_0}}(u)$$

where $\delta_{\mathbf{c_0}}$ is the indicator function of the closed convex set $\mathbf{c_0}$. Now, the nullspace of the matrix $\boldsymbol{B}$ is one dimensional and spanned by $\begin{pmatrix} \mathbf{1}_{NM}^\top & \mathbf{0}_{(N-1)M}^\top & \mathbf{0}_{N(M-1)}^\top \end{pmatrix}^\top$. This implies that the feasible set is precisely the orthogonal complement of the nullspace of the matrix $\boldsymbol{B}$. This in turn implies in particular that the quadratic function

$$\frac{1}{2\tau} \left\| \boldsymbol{B} \begin{pmatrix} u \\ v^v \\ v^h \end{pmatrix} - \begin{pmatrix} g^v \\ g^h \end{pmatrix} \right\|^2$$

is strongly convex on the feasible set. Since $\|v^v\|_{\ell_1(c^v)} + \left\|v^h\right\|_{\ell_1(c^h)}$ is convex, the function f is strongly convex on the feasible set, hence the minimizer is unique. □

Problem (4.5) is a convex optimization problem, composed of a smooth part and non-smooth part. It could be tackled using well-established methods based on proximal steps such as FISTA [BT09] or IHT [BD09], but here we follow the IRLS approach [Law61, HW77, Osb85, DDDM04, DDFG10, FPRW16]. This is motivated by empirical evidence provided in [FPRW16] showing the superiority of the IRLS approach in high dimension. We start by rewriting F as

$$\begin{aligned} F(\boldsymbol{U}, \boldsymbol{V^v}, \boldsymbol{V^h}) = & \sum_{\substack{1 \le i \le N-1 \\ 1 \le j \le M}} \sqrt{(\boldsymbol{C}^{\boldsymbol{v}}_{i,j})^2 (\boldsymbol{V}^{\boldsymbol{v}}_{i,j})^2} + \sum_{\substack{1 \le i \le N \\ 1 \le j \le M-1}} \sqrt{(\boldsymbol{C}^{\boldsymbol{h}}_{i,j})^2 (\boldsymbol{V}^{\boldsymbol{h}}_{i,j})^2} \\ & + \frac{1}{2\tau} \left\| \boldsymbol{SU} - \boldsymbol{G^v} - \boldsymbol{V^v} \right\|^2 + \frac{1}{2\tau} \left\| \boldsymbol{UT} - \boldsymbol{G^h} - \boldsymbol{V^h} \right\|^2. \end{aligned} \tag{4.6}$$

The issue in optimizing the above is the non-smooth square root part of the objective. For $x \in \mathbb{R}$, we have

$$|x| = \sqrt{x^2} = \frac{1}{2} \inf_{w > 0} \left\{ \frac{x^2}{w} + w \right\}. \tag{4.7}$$

This simple equation is a classical way of rewriting sparsity-inducing norms in terms of quadratic objectives [BR96, DDDM04, DDFG10, BJM+12, MBP+14, FPRW16]. Whenever $x \neq 0$, the above infimum is attained for $w = |x|$. Unfortunately the infimum is not attained when $x = 0$. In our case this is an issue since our ultimate goal is to perform minimization with respect to this variable, so we introduce a fixed parameter $\delta > 0$ and observe that

$$\sqrt{x^2 + \delta^2} = \frac{1}{2} \inf_{w \ge \frac{\delta}{2}} \left\{ (x^2 + \delta^2) \frac{1}{w} + w \right\}. \tag{4.8}$$

The above infimum is always attained for $w = \sqrt{x^2+\delta^2}$. We therefore introduce the function F_δ defined as

$$\begin{aligned} F_\delta(\boldsymbol{U},\boldsymbol{V^v},\boldsymbol{V^h}) := & \sum_{\substack{1\le i\le N-1\\ 1\le j\le M}} \sqrt{(\boldsymbol{C^v_{i,j}})^2(\boldsymbol{V^v_{i,j}})^2+\delta^2} + \sum_{\substack{1\le i\le N\\ 1\le j\le M-1}} \sqrt{(\boldsymbol{C^h_{i,j}})^2(\boldsymbol{V^h_{i,j}})^2+\delta^2} \\ & + \frac{1}{2\tau}\left\|\boldsymbol{SU}-\boldsymbol{G^v}-\boldsymbol{V^v}\right\|^2 + \frac{1}{2\tau}\left\|\boldsymbol{UT}-\boldsymbol{G^h}-\boldsymbol{V^h}\right\|^2. \end{aligned} \tag{4.9}$$

We then show the following approximation result, as a function of δ.

Theorem 4.4.2. *For any $\boldsymbol{U}\in\mathbb{R}^{N\times M}, \boldsymbol{V^v}\in\mathbb{R}^{(N-1)\times M}$ and $\boldsymbol{V^h}\in\mathbb{R}^{N\times(M-1)}$,*

$$F(\boldsymbol{U},\boldsymbol{V^v},\boldsymbol{V^h}) \le F_\delta(\boldsymbol{U},\boldsymbol{V^v},\boldsymbol{V^h}) \le F(\boldsymbol{U},\boldsymbol{V^v},\boldsymbol{V^h}) + \delta\left((N-1)M+N(M-1)\right).$$

Proof. It is clear that $F \le F_\delta$ since $\delta > 0$. The second inequality comes from the fact that $\sqrt{a^2+b^2} \le a+b$ for $a,b\ge 0$ hence

$$\begin{aligned} F_\delta(\boldsymbol{U},\boldsymbol{V^v},\boldsymbol{V^h}) &\le F(\boldsymbol{U},\boldsymbol{V^v},\boldsymbol{V^h}) + \sum_{\substack{1\le i\le N-1\\ 1\le j\le M}}\delta + \sum_{\substack{1\le i\le N\\ 1\le j\le M-1}}\delta \\ &= F(\boldsymbol{U},\boldsymbol{V^v},\boldsymbol{V^h}) + \delta(N-1)M + \delta N(M-1) \end{aligned}$$

which is the desired result. □

Using the above remark, we get that minimizing F_δ is equivalent to solving

$$\inf_{\substack{\boldsymbol{U}\in\mathbf{C_0},\\ \boldsymbol{V^v},\boldsymbol{V^h},\\ \boldsymbol{W^v}\ge\delta/2,\boldsymbol{W^h}\ge\delta/2}} \left\{ \begin{array}{l} \frac{1}{2}\displaystyle\sum_{\substack{1\le i\le N-1\\ 1\le j\le M}} ((\boldsymbol{C^v_{i,j}})^2(\boldsymbol{V^v_{i,j}})^2+\delta^2)\frac{1}{\boldsymbol{W^v_{i,j}}} + \boldsymbol{W^v_{i,j}} \\ +\frac{1}{2}\displaystyle\sum_{\substack{1\le i\le N\\ 1\le j\le M-1}} ((\boldsymbol{C^h_{i,j}})^2(\boldsymbol{V^h_{i,j}})^2+\delta^2)\frac{1}{\boldsymbol{W^h_{i,j}}} + \boldsymbol{W^h_{i,j}} \\ +\frac{1}{2\tau}\left\|\boldsymbol{SU}-\boldsymbol{G^v}-\boldsymbol{V^v}\right\|^2 + \frac{1}{2\tau}\left\|\boldsymbol{UT}-\boldsymbol{G^h}-\boldsymbol{V^h}\right\|^2 \end{array} \right\}, \tag{4.10}$$

where $\boldsymbol{W^v}\in\mathbb{R}^{(N-1)\times M}$ and $\boldsymbol{W^h}\in\mathbb{R}^{M\times(N-1)}$. We solve (4.10) by alternatively minimizing with respect to $(\boldsymbol{U},\boldsymbol{V^v},\boldsymbol{V^h})$ and $(\boldsymbol{W^v},\boldsymbol{W^h})$. The objective is a simple convex function with respect to $\boldsymbol{W^v}$ and $\boldsymbol{W^h}$ and setting the gradients to 0 yields the following closed-form updates for $\boldsymbol{W^v}$ and $\boldsymbol{W^h}$:

$$\begin{aligned} \boldsymbol{W^v_{i,j}} &= \sqrt{(\boldsymbol{C^v_{i,j}})^2(\boldsymbol{V^v_{i,j}})^2+\delta^2}, \quad 1\le i\le N-1, \quad 1\le j\le M, \\ \boldsymbol{W^h_{i,j}} &= \sqrt{(\boldsymbol{C^h_{i,j}})^2(\boldsymbol{V^h_{i,j}})^2+\delta^2}, \quad 1\le i\le N, \quad 1\le j\le M-1. \end{aligned} \tag{4.11}$$

On the other hand, objective (4.10) is a quadratic function of $(\boldsymbol{U},\boldsymbol{V^v},\boldsymbol{V^h})$, and the minimization can be performed using the following simple proposition.

Proposition 4.4.1. *Minimizing (4.10) with respect to $(\boldsymbol{U},\boldsymbol{V^v},\boldsymbol{V^h})\in\mathbf{C_0}\times\mathbb{R}^{(N-1)\times M}\times\mathbb{R}^{N\times(M-1)}$ is equivalent to solving the system*

$$\begin{pmatrix} \frac{1}{\tau}\left(\boldsymbol{S}^\top\boldsymbol{SU}+\boldsymbol{UTT}^\top-\boldsymbol{S}^\top\boldsymbol{V^v}-\boldsymbol{V^h}\boldsymbol{T}^\top\right) \\ \boldsymbol{C^v}\odot\boldsymbol{C^v}\odot\frac{1}{\boldsymbol{W^v}}\odot\boldsymbol{V^v}+\frac{1}{\tau}(\boldsymbol{V^v}-\boldsymbol{SU}) \\ \boldsymbol{C^h}\odot\boldsymbol{C^h}\odot\frac{1}{\boldsymbol{W^h}}\odot\boldsymbol{V^h}+\frac{1}{\tau}(\boldsymbol{V^h}-\boldsymbol{UT}) \end{pmatrix} = \begin{pmatrix} \frac{1}{\tau}(\boldsymbol{S}^\top\boldsymbol{G^v}+\boldsymbol{G^h}\boldsymbol{T}^\top) \\ -\frac{1}{\tau}\boldsymbol{G^v} \\ -\frac{1}{\tau}\boldsymbol{G^h} \end{pmatrix}, \tag{4.12}$$

for $(\boldsymbol{U},\boldsymbol{V^v},\boldsymbol{V^h})\in\mathbb{R}^{N\times M}\times\mathbb{R}^{(N-1)\times M}\times\mathbb{R}^{N\times(M-1)}$, and then setting $\boldsymbol{U}=\boldsymbol{U}-\frac{\sum_{i,j}\boldsymbol{U}_{i,j}}{NM}$.

Proof. (4.10) is a quadratic function of $(\boldsymbol{U}, \boldsymbol{V^v}, \boldsymbol{V^h})$, and setting its gradient with respect to those variables yields the linear system (4.12). Once we get a solution, and since the system is invariant by additive shift on $\boldsymbol{U}$, we can simply rescale it by its mean, i.e. set $\boldsymbol{U}$ to $\boldsymbol{U} - \frac{\sum_{i,j} \boldsymbol{U}_{i,j}}{NM}$ to ensure $\boldsymbol{U} \in \mathbf{C_0}$. □

Solving (4.12) exactly is too costly in practice. Instead, we suggest solving it approximately using the conjugate gradient method, which we detail in the next section. For now we summarize the above steps in Algorithm 15.

Algorithm 15 IRLS algorithm

Input: $\boldsymbol{G^v} \in \mathbb{R}^{(N-1)\times M}, \boldsymbol{G^h} \in \mathbb{R}^{N\times(M-1)}, \boldsymbol{C^v} \in \mathbb{R}^{(N-1)\times M}, \boldsymbol{C^h} \in \mathbb{R}^{N\times(M-1)}, \delta > 0.$

$\boldsymbol{U}^0 = \mathbf{0}_{N\times M}$

$\boldsymbol{V}^{\boldsymbol{v},0} = \boldsymbol{SU} - \boldsymbol{G^v}$, $\boldsymbol{V}^{\boldsymbol{h},0} = \boldsymbol{UT} - \boldsymbol{G^h}$

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

$$\boldsymbol{W}^{\boldsymbol{v},k}_{i,j} = \sqrt{(\boldsymbol{C}^{\boldsymbol{v}}_{i,j})^2((\boldsymbol{V}^{\boldsymbol{v},k})_{i,j})^2 + \delta^2}, \quad 1 \le i \le N-1, \quad 1 \le j \le M$$

$$\boldsymbol{W}^{\boldsymbol{h},k}_{i,j} = \sqrt{(\boldsymbol{C}^{\boldsymbol{h}}_{i,j})^2((\boldsymbol{V}^{\boldsymbol{h},k})_{i,j})^2 + \delta^2}, \quad 1 \le i \le N, \quad 1 \le j \le M-1.$$

For $(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{\boldsymbol{v},k+1}, \boldsymbol{V}^{\boldsymbol{h},k+1})$, use Algorithm 16 to approximately solve

$$\begin{pmatrix} \boldsymbol{S}^\top\boldsymbol{SU}^{k+1} + \boldsymbol{U}^{k+1}\boldsymbol{TT}^\top - \boldsymbol{S}^\top\boldsymbol{V}^{\boldsymbol{v},k+1} - \boldsymbol{V}^{\boldsymbol{h},k+1}\boldsymbol{T}^\top \\ \boldsymbol{C^v} \odot \boldsymbol{C^v} \odot \frac{1}{\boldsymbol{W}^{\boldsymbol{v},k}} \odot \boldsymbol{V}^{\boldsymbol{v},k+1} + \frac{1}{\tau}(\boldsymbol{V}^{\boldsymbol{v},k+1} - \boldsymbol{SU}^{k+1}) \\ \boldsymbol{C^h} \odot \boldsymbol{C^h} \odot \frac{1}{\boldsymbol{W}^{\boldsymbol{h},k}} \odot \boldsymbol{V}^{\boldsymbol{h},k+1} + \frac{1}{\tau}(\boldsymbol{V}^{\boldsymbol{h},k+1} - \boldsymbol{U}^{k+1}\boldsymbol{T}) \end{pmatrix} = \begin{pmatrix} \boldsymbol{S}^\top\boldsymbol{G^v} + \boldsymbol{G^h}\boldsymbol{T}^\top \\ -\frac{1}{\tau}\boldsymbol{G^v} \\ -\frac{1}{\tau}\boldsymbol{G^h} \end{pmatrix}.$$

$\boldsymbol{U}^{k+1} = \boldsymbol{U}^{k+1} - \frac{\sum_{ij}(\boldsymbol{U}^{k+1})_{ij}}{NM}$.

end for

The next theorem gives a sublinear convergence rate for Algorithm 15, provided the linear system (4.12) is solved with enough accuracy. More precisely, it states that the algorithm converges as long as the function value at the iterates $(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{\boldsymbol{v},k+1}, \boldsymbol{V}^{\boldsymbol{h},k+1})$ is smaller than the function value that would have been obtained if a simple gradient step had been taken on the previous iterates $(\boldsymbol{U}^k, \boldsymbol{V}^{\boldsymbol{v},k}, \boldsymbol{V}^{\boldsymbol{h},k})$.

Theorem 4.4.3. *Define $H_\delta(\boldsymbol{U}, \boldsymbol{V^v}, \boldsymbol{V^h}, \boldsymbol{W^v}, \boldsymbol{W^h})$ as the function minimized in (4.10) and*

$$C_{max} = \max(\max_{ij}(\boldsymbol{C}^{\boldsymbol{v}}_{ij}),\ \max_{ij}(\boldsymbol{C}^{\boldsymbol{h}}_{ij})), \quad L = \frac{12}{\tau} + \frac{1}{\delta}C^2_{max}.$$

Let the candidate solutions $\tilde{\boldsymbol{U}}^k, \tilde{\boldsymbol{V}}^{\boldsymbol{v},k}, \tilde{\boldsymbol{V}}^{\boldsymbol{h},k}$ at iteration k be

$$\begin{aligned} \tilde{\boldsymbol{U}}^k &= \boldsymbol{U}^k - \frac{1}{\tau L}\left(\boldsymbol{S}^\top\left(\boldsymbol{SU}^k - \boldsymbol{G^v} - \boldsymbol{V}^{\boldsymbol{v},k}\right) + \left(\boldsymbol{U}^k\boldsymbol{T} - \boldsymbol{G^h} - \boldsymbol{V}^{\boldsymbol{h},k}\right)\boldsymbol{T}^\top\right) \\ \tilde{\boldsymbol{V}}^{\boldsymbol{v},k} &= \boldsymbol{V}^{\boldsymbol{v},k} - \frac{1}{L}\left(\boldsymbol{C^v} \odot \boldsymbol{C^v} \odot \frac{1}{\boldsymbol{W}^{\boldsymbol{v},k}} \odot \boldsymbol{V}^{\boldsymbol{v},k} + \frac{1}{\tau}(\boldsymbol{V}^{\boldsymbol{v},k} - \boldsymbol{SU}^k + \boldsymbol{G^v})\right) \\ \tilde{\boldsymbol{V}}^{\boldsymbol{h},k} &= \boldsymbol{V}^{\boldsymbol{h},k} - \frac{1}{L}\left(\boldsymbol{C^h} \odot \boldsymbol{C^h} \odot \frac{1}{\boldsymbol{W}^{\boldsymbol{h},k}} \odot \boldsymbol{V}^{\boldsymbol{h},k} + \frac{1}{\tau}(\boldsymbol{V}^{\boldsymbol{h},k} - \boldsymbol{U}^k\boldsymbol{T} + \boldsymbol{G^h})\right) \end{aligned} \tag{4.13}$$

Suppose that at each iteration of the IRLS algorithm 15, the approximate solutions $(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{\boldsymbol{v},k+1}, \boldsymbol{V}^{\boldsymbol{h},k+1})$ of the linear system (4.12) satisfy

$$H_\delta(\boldsymbol{U}^{k+1}, \boldsymbol{V}^{\boldsymbol{v},k+1}, \boldsymbol{V}^{\boldsymbol{h},k+1}, \boldsymbol{W}^{\boldsymbol{v},k}, \boldsymbol{W}^{\boldsymbol{h},k}) \le H_\delta(\tilde{\boldsymbol{U}}^k, \tilde{\boldsymbol{V}}^{\boldsymbol{v},k}, \tilde{\boldsymbol{V}}^{\boldsymbol{h},k}, \boldsymbol{W}^{\boldsymbol{v},k}, \boldsymbol{W}^{\boldsymbol{h},k}) \tag{4.14}$$

Then, after K iterations, we have

$$\boxed{F_\delta(\boldsymbol{U}^K, \boldsymbol{V}^{\boldsymbol{v},K}, \boldsymbol{V}^{\boldsymbol{h},K}) - F_\delta^* \leq \max\left(\left(\frac{1}{2}\right)^{\frac{K-1}{2}}, \frac{8LR^2}{K-1}\right),} \tag{4.15}$$

where F_δ^* is the optimal value of F and the constant R is defined as

$$R = \max_{\boldsymbol{Y} \in \Gamma} \{\|\boldsymbol{Y} - \boldsymbol{Y}^*\| \mid H_\delta(\boldsymbol{Y}) \leq H_\delta(\boldsymbol{Y}_0)\}, \tag{4.16}$$

with H_δ being the function to be minimized in problem (4.10), Γ the feasible set of problem (4.10),

$$\boldsymbol{Y}^* = \begin{pmatrix} \boldsymbol{U}^* \\ \boldsymbol{V}^{\boldsymbol{v},*} \\ \boldsymbol{V}^{\boldsymbol{h},*} \\ \boldsymbol{W}^{\boldsymbol{v},*} \\ \boldsymbol{W}^{\boldsymbol{h},*} \end{pmatrix} = \begin{pmatrix} \boldsymbol{U}^* \\ \boldsymbol{V}^{\boldsymbol{v},*} \\ \boldsymbol{V}^{\boldsymbol{h},*} \\ \sqrt{\boldsymbol{C}^{\boldsymbol{v}} \odot \boldsymbol{V}^{\boldsymbol{v},*} + \delta^2} \\ \sqrt{\boldsymbol{C}^{\boldsymbol{h}} \odot \boldsymbol{V}^{\boldsymbol{h},*} + \delta^2} \end{pmatrix}, \quad \boldsymbol{Y}_0 = \begin{pmatrix} \boldsymbol{U}^0 \\ \boldsymbol{V}^{\boldsymbol{v},0} \\ \boldsymbol{V}^{\boldsymbol{h},0} \\ \boldsymbol{W}^{\boldsymbol{v},0} \\ \boldsymbol{W}^{\boldsymbol{h},0} \end{pmatrix}, \tag{4.17}$$

and $\left((\boldsymbol{U}^*)^\top, (\boldsymbol{V}^{\boldsymbol{v},*})^\top, (\boldsymbol{V}^{\boldsymbol{h},*})^\top\right)^\top$ is the minimizer of F_δ.

Proof. This is a direct consequence of [Bec15, Theorem 4.1]. Indeed, defining

$$s(u, v^v, v^h) = \frac{1}{2\tau} \left\| \begin{pmatrix} (\mathbf{I}_M \otimes \boldsymbol{S}) & -\mathbf{I}_{(N-1)M} & \mathbf{0} \\ (\boldsymbol{T}^\top \otimes \mathbf{I}_N) & \mathbf{0} & -\mathbf{I}_{N(M-1)} \end{pmatrix} \begin{pmatrix} u \\ v^v \\ v^h \end{pmatrix} - \begin{pmatrix} g^v \\ g^h \end{pmatrix} \right\|^2,$$

this result directly implies (4.15), where L is given by

$$L = L_s + \frac{1}{\delta} \lambda_{\max} \left(\sum_{k=NM+1}^{NM+(N-1)M} (c_k^v)^2 e_k e_k^\top + \sum_{k=NM+(N-1)M+1}^{MM+(N-1)M+N(M-1)} (c_k^h)^2 e_k e_k^\top \right), \tag{4.18}$$

where $\{e_k\}_{k=1}^{NM+(N-1)M+N(M-1)}$ are the vectors in the canonical basis of $\mathbb{R}^{NM+(N-1)M+N(M-1)}$, L_s is the Lipschitz constant of the gradient of s, which is equal to the maximum eigenvalue of the Hessian of s, and $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue of the matrix in parentheses. We now bound L_s. The Hessian of s is

$$\begin{aligned} \nabla^2 s(u, v^v, v^h) &= \frac{1}{\tau} \begin{pmatrix} (\mathbf{I} \otimes \boldsymbol{S}) & -\mathbf{I} & \mathbf{0} \\ (\boldsymbol{T}^\top \otimes \mathbf{I}) & \mathbf{0} & -\mathbf{I} \end{pmatrix}^\top \begin{pmatrix} (\mathbf{I} \otimes \boldsymbol{S}) & -\mathbf{I} & \mathbf{0} \\ (\boldsymbol{T}^\top \otimes \mathbf{I}) & \mathbf{0} & -\mathbf{I} \end{pmatrix} \\ &= \frac{1}{\tau} \begin{pmatrix} (\mathbf{I} \otimes \boldsymbol{S}^\top \boldsymbol{S}) + (\boldsymbol{T}\boldsymbol{T}^\top \otimes \mathbf{I}) & -(\mathbf{I} \otimes \boldsymbol{S})^\top & -(\boldsymbol{T} \otimes \mathbf{I}) \\ -(\mathbf{I} \otimes \boldsymbol{S}) & \mathbf{I} & \mathbf{0} \\ -(\boldsymbol{T}^\top \otimes \mathbf{I}) & \mathbf{0} & \mathbf{I} \end{pmatrix} \end{aligned}$$

We will make use of Gershgorin circle theorem to prove a bound on the largest eigenvalue of $\nabla^2 s$ [GER31]. The theorem states that all the eigenvalues of $\nabla^2 s$ lie within the union of the discs

$$D_i = \left\{ \lambda \in \mathbb{R} \mid |\lambda - (\nabla^2 s)_{ii}| \leq \sum_{j \neq i} |(\nabla^2 s)_{ij}| \right\}, \tag{4.19}$$

for $1 \leq i \leq NM + (N-1)M + N(M-1)$.

Writing out the definitions of $\mathbf{I}_M \otimes \boldsymbol{S}$, $\mathbf{I}_M \otimes \boldsymbol{S}^\top \boldsymbol{S}$, $\boldsymbol{T}\boldsymbol{T}^\top \otimes \mathbf{I}_N$ and $\boldsymbol{T} \otimes \mathbf{I}_N$ (see Appendix 4.A), one can see that all diagonal elements of $\nabla^2 s$ are positive (which can also be deduced from the fact that $\nabla^2 s$ is the Hessian of a convex quadratic) and bounded above by $\frac{4}{\tau}$, and that

$$\max_i \sum_{j \neq i} |(\nabla^2 s)_{ij}| = \frac{8}{\tau}.$$

This value of $\frac{8}{\tau}$ is in particular attained for lines where the diagonal element is maximal, i.e. equal to $\frac{4}{\tau}$. This is the case for example at line $N+2$. Going back to the discs defined in (4.19), this implies that the maximum eigenvalue of ∇_s^2, and therefore L_s, is such that

$$L_s - \frac{4}{\tau} \leq \frac{8}{\tau},$$

and thus $L_s \leq \frac{12}{\tau}$.

We now bound the right part of (4.18), i.e. the maximum eigenvalue of

$$\left(\sum_{k=NM+1}^{NM+(N-1)M} (c_k^v)^2 e_k e_k^\top + \sum_{k=NM+(N-1)M+1}^{MM+(N-1)M+N(M-1)} (c_k^h)^2 e_k e_k^\top \right).$$

This is a diagonal matrix, whose first NM elements are zero, and the remaining are the squares of the coefficient vectors c^v and c^h. Therefore its maximum eigenvalue is given by

$$\max\left(\max_k \left\{ (c_k^v)^2 \right\}, \ \max_k \left\{ (c_k^h)^2 \right\} \right) = \max\left(\max_{ij} \left\{ (\boldsymbol{C}_{ij}^{\boldsymbol{v}})^2 \right\}, \ \max_{ij} \left\{ (\boldsymbol{C}_{ij}^{\boldsymbol{h}})^2 \right\} \right)$$

hence the desired result. □

Remark 4.4.1. *The result in [Bec15, Theorem 4.1] is actually stated for an exact minimization step, i.e. when the linear system (4.12) is solved exactly. However, upon closer inspection of the proof, one notices that the result holds as long as a sufficient decrease property holds. In our case this decrease property translates to condition (4.14).*

Remark 4.4.2. *In our experiments, we always checked that condition (4.14) was satisfied after running CG. In practice this was always the case, even if CG was run for just a few iterations. We have also tried to implement Algorithm 15 using only the candidate updates (4.13) instead of running CG, but this did not yield competitive performance.*

4.4.2 Conjugate gradient

Since the updates for $\boldsymbol{W}^{\boldsymbol{v}}$ and $\boldsymbol{W}^{\boldsymbol{h}}$ have simple closed form solutions, the main numerical difficulty of Algorithm 15 relies in solving the linear system (4.12). In large dimensions such as the ones encountered in InSAR imagery, direct linear solvers will be too slow and we instead decide to use a carefully preconditioned conjugate gradient (CG) method [NW06]. We detail the method in the specific case of problem (4.12). First let us simplify the notation by defining $n = NM + (N-1)M + N(M-1)$ and

$$\boldsymbol{B}_1 := \frac{1}{\tau}(\boldsymbol{S}^\top \boldsymbol{G}^{\boldsymbol{v}} + \boldsymbol{G}^{\boldsymbol{h}} \boldsymbol{T}^\top), \ \boldsymbol{B}_2 := -\frac{1}{\tau}\boldsymbol{G}^{\boldsymbol{v}}, \ \boldsymbol{B}_3 := -\frac{1}{\tau}\boldsymbol{G}^{\boldsymbol{h}}. \tag{4.20}$$

In order to make the algorithm more readable, we will vectorize the linear system using the following proposition.

Proposition 4.4.2. *Defining the matrix* $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ *as*

$$\boldsymbol{A} := \begin{pmatrix} \frac{1}{\tau}\left((\mathbf{I}_M \otimes \boldsymbol{S}^\top \boldsymbol{S}) + (\boldsymbol{T}\boldsymbol{T}^\top \otimes \mathbf{I}_N)\right) & -\frac{1}{\tau}(\mathbf{I}_M \otimes \boldsymbol{S}^\top) & -\frac{1}{\tau}(\boldsymbol{T} \otimes \mathbf{I}_N) \\ -\frac{1}{\tau}(\mathbf{I}_M \otimes \boldsymbol{S}) & \boldsymbol{D}^{\boldsymbol{v}} + \frac{1}{\tau}\mathbf{I} & \mathbf{0} \\ -\frac{1}{\tau}(\boldsymbol{T}^\top \otimes \mathbf{I}_N) & \mathbf{0} & \boldsymbol{D}^{\boldsymbol{h}} + \frac{1}{\tau}\mathbf{I} \end{pmatrix}, \tag{4.21}$$

the linear system (4.12) *is equivalent to the following linear system*

$$\boldsymbol{A}\begin{pmatrix} u \\ v^v \\ v^h \end{pmatrix} = b, \tag{4.22}$$

where

$$\begin{aligned} \boldsymbol{D}^{\boldsymbol{v}} &:= \mathbf{diag}(\mathbf{vec}(\boldsymbol{C}^{\boldsymbol{v}} \odot \boldsymbol{C}^{\boldsymbol{v}} \odot \frac{1}{\boldsymbol{W}^{\boldsymbol{v}}})), \\ \boldsymbol{D}^{\boldsymbol{h}} &:= \mathbf{diag}(\mathbf{vec}(\boldsymbol{C}^{\boldsymbol{h}} \odot \boldsymbol{C}^{\boldsymbol{h}} \odot \frac{1}{\boldsymbol{W}^{\boldsymbol{h}}})), \\ b &:= \begin{pmatrix} \mathbf{vec}(\boldsymbol{B}_1) \\ \mathbf{vec}(\boldsymbol{B}_2) \\ \mathbf{vec}(\boldsymbol{B}_3) \end{pmatrix}. \end{aligned} \tag{4.23}$$

Indeed, $(\boldsymbol{U}, \boldsymbol{V}^{\boldsymbol{v}}, \boldsymbol{V}^{\boldsymbol{h}})$ *solves* (4.12) *if and only if* (u, v^v, v^h) *solves* (4.22) *with* $u = \mathbf{vec}(\boldsymbol{U})$, $v^v = \mathbf{vec}(\boldsymbol{V}^{\boldsymbol{v}})$, $v^h = \mathbf{vec}(\boldsymbol{V}^{\boldsymbol{h}})$.

Proof. This is a direct consequence of the vectorization property (4.1) and the fact that

$$\begin{aligned} \mathbf{vec}(\boldsymbol{C}^{\boldsymbol{v}} \odot \boldsymbol{C}^{\boldsymbol{v}} \odot \frac{1}{\boldsymbol{W}^{\boldsymbol{v}}} \odot \boldsymbol{V}^{\boldsymbol{v}}) &= \mathbf{diag}(\mathbf{vec}(\boldsymbol{C}^{\boldsymbol{v}} \odot \boldsymbol{C}^{\boldsymbol{v}} \odot \frac{1}{\boldsymbol{W}^{\boldsymbol{v}}}))\mathbf{vec}(\boldsymbol{V}^{\boldsymbol{v}}) \\ \mathbf{vec}(\boldsymbol{C}^{\boldsymbol{h}} \odot \boldsymbol{C}^{\boldsymbol{h}} \odot \frac{1}{\boldsymbol{W}^{\boldsymbol{h}}} \odot \boldsymbol{V}^{\boldsymbol{h}}) &= \mathbf{diag}(\mathbf{vec}(\boldsymbol{C}^{\boldsymbol{h}} \odot \boldsymbol{C}^{\boldsymbol{h}} \odot \frac{1}{\boldsymbol{W}^{\boldsymbol{h}}}))\mathbf{vec}(\boldsymbol{V}^{\boldsymbol{h}}) \end{aligned}$$

□

For completeness, we detail the conjugate gradient method for solving problem (4.22) in Algorithm 16. We emphasize that in practice the system is never vectorized and the matrix $\boldsymbol{A}$ is never constructed. Instead, we keep all variables in matrix form and matrix vector products are computed using the linear map defined in (4.12). The reason we decided to introduce $\boldsymbol{A}$ in this section is two-fold. First it allows to write out the conjugate gradient method of Algorithm 16 in what we believe to be a more reader-friendly way. Second, and more importantly, this way of writing will shed some light on how to pick the preconditioner, which is the key to fast convergence of the conjugate gradient method. This is the topic of the next section.

Remark 4.4.3. *The matrix* $\boldsymbol{A}$ *is singular. Its nullspace is one-dimensional, consisting of the vectors that are constant in the first* NM *entries and* 0 *everywhere else (this is due to the block* $(\mathbf{I} \otimes \boldsymbol{S}^\top \boldsymbol{S}) + (\boldsymbol{T}\boldsymbol{T}^\top \otimes \mathbf{I})$ *being invariant under additive scaling). The conjugate gradient method still converges for singular matrices [Kaa88, Hay18].*

4.4.3 Choosing the preconditioner

The convergence of CG directly depends on the conditioning of the matrix $\boldsymbol{A}$. In later iterations of IRLS when the weight matrices $\boldsymbol{W}^{\boldsymbol{v}}$ and $\boldsymbol{W}^{\boldsymbol{h}}$ have many small entries, the matrix $\boldsymbol{A}$ becomes very ill-conditioned. To overcome this issue, the most common trick is to use a preconditioner, i.e. a matrix $\boldsymbol{D} = \boldsymbol{C}^\top \boldsymbol{C}$ such that $\boldsymbol{C}^{-\top}\boldsymbol{A}\boldsymbol{C}^{-1}$ is better-conditioned than $\boldsymbol{A}$, and such that linear systems of the form $\boldsymbol{D}z = r$ are very cheap to solve in practice. For more on the topic of

Algorithm 16 Conjugate gradient (CG)

Input: $b \in \mathbb{R}^n$, matrix $A \in \mathbb{R}^{n \times n}$ or a function to compute the linear map $\boldsymbol{A}p$ for vectors p, preconditioner $\boldsymbol{D} \succeq 0$
Initialize $u^0, v^{v,0}$ and $v^{h,0}$.
Set

$$r_l = b - \boldsymbol{A} \begin{pmatrix} u^0 \\ v^{v,0} \\ v^{h,0} \end{pmatrix}$$

Solve $\boldsymbol{D}z_l = r_l$
$p_l = z_l$
$\rho_l = r_l^\top z_l$
for $l = 0, 1, \ldots$ **do**
    $\alpha_l = \rho_l / (p_l^\top \boldsymbol{A} p_l)$
    Set

$$\begin{pmatrix} u^{l+1} \\ v^{v,l+1} \\ v^{h,l+1} \end{pmatrix} = \begin{pmatrix} u^l \\ v^{v,l} \\ v^{h,l} \end{pmatrix} + \alpha_l p_l$$

    $r_{l+1} = r_l - \alpha_l \boldsymbol{A} p_l$
    Solve $\boldsymbol{D}z_{l+1} = r_{l+1}$
    $\rho_{l+1} = (r_{l+1})^\top z_{l+1}$
    $\beta = \rho_{l+1} / \rho_l$
    $p_{l+1} = z_{l+1} + \beta p_l$
end for

preconditioning, we refer the interested reader to [Kel95, NW06]. It is interesting to note that in [GR94], the authors suggest using a preconditioned conjugate gradient method to solve the L^2-norm phase unwrapping problem, whose structure is close to the linear system we solve at each IRLS iteration. Their preconditioner is however different from the one we present next and proved to be less efficient in practice.

A common choice of preconditioner is the so-called Jacobi preconditioner, where $\boldsymbol{D}$ is set to a diagonal matrix whose diagonal matches the diagonal of $\boldsymbol{A}$. In our case we choose the more efficient block diagonal preconditioner, i.e. we define $\boldsymbol{D}$ as

$$\boldsymbol{D} := \begin{pmatrix} \frac{1}{\tau}\left((\mathbf{I} \otimes \boldsymbol{S}^\top \boldsymbol{S}) + (\boldsymbol{T}\boldsymbol{T}^\top \otimes \mathbf{I})\right) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{D}^{\boldsymbol{v}} + \frac{1}{\tau}\mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{D}^{\boldsymbol{h}} + \frac{1}{\tau}\mathbf{I} \end{pmatrix}. \tag{4.24}$$

For solving $\boldsymbol{D}z = r$, we proceed as follows. Writing

$$z = \begin{pmatrix} z_a^\top & z_b^\top & z_c^\top \end{pmatrix}^\top \text{ and } r = \begin{pmatrix} r_a^\top & r_b^\top & r_c^\top \end{pmatrix}^\top$$

the problem comes down to solving the following three smaller linear systems for z_a, z_b and z_c

$$\begin{aligned} \left((\mathbf{I} \otimes \boldsymbol{S}^\top \boldsymbol{S}) + (\boldsymbol{T}\boldsymbol{T}^\top \otimes \mathbf{I})\right) z_a &= \tau r_a \\ (\boldsymbol{D}^{\boldsymbol{v}} + \frac{1}{\tau}\mathbf{I}) z_b &= r_b \\ (\boldsymbol{D}^{\boldsymbol{h}} + \frac{1}{\tau}\mathbf{I}) z_c &= r_c \end{aligned} \tag{4.25}$$

Since $\boldsymbol{D}^{\boldsymbol{v}}$ and $\boldsymbol{D}^{\boldsymbol{h}}$ are diagonal matrices, the last two subproblems are numerically fast to solve. Problem (4.25) can be recast as

$$\boldsymbol{S}^\top \boldsymbol{S} \boldsymbol{Z}_a + \boldsymbol{Z}_a \boldsymbol{T}\boldsymbol{T}^\top = \tau \boldsymbol{R}_a, \tag{4.26}$$

where $z_a = \mathbf{vec}(\boldsymbol{Z}_a)$ and $r_a = \mathbf{vec}(\boldsymbol{R}_a)$. This equation is commonly known as a Sylvester equation (in the case where $N = M$, then $\boldsymbol{T} = \boldsymbol{S}^\top$, and the equation is known as a continuous-time Lyapunov equation). The Bartels-Stewart algorithm [BS72] is a common way to solve such equations by first computing the Schur decompositions of $\boldsymbol{S}^\top \boldsymbol{S}$ and $\boldsymbol{T}\boldsymbol{T}^\top$ and then solving for $\boldsymbol{Z}_c$ after some careful algebraic manipulations. In our case, since $\boldsymbol{S}^\top \boldsymbol{S}$ and $\boldsymbol{T}\boldsymbol{T}^\top$ are positive semidefinite matrices, the Schur decompositions are eigendecompositions

$$\boldsymbol{S}^\top \boldsymbol{S} = \boldsymbol{P}_S \boldsymbol{\Lambda}_S \boldsymbol{P}_S^\top, \quad \boldsymbol{T}\boldsymbol{T}^\top = \boldsymbol{P}_T \boldsymbol{\Lambda}_T \boldsymbol{P}_T^\top, \tag{4.27}$$

where $\boldsymbol{\Lambda}_S, \boldsymbol{\Lambda}_T$ are diagonal matrices, and $\boldsymbol{P}_S$, $\boldsymbol{P}_T$ are orthogonal matrices. Plugging this into (4.26) gives

$$\begin{aligned} & \boldsymbol{P}_S \boldsymbol{\Lambda}_S \boldsymbol{P}_S^\top \boldsymbol{Z}_a + \boldsymbol{Z}_a \boldsymbol{P}_T \boldsymbol{\Lambda}_T \boldsymbol{P}_T^\top = \tau \boldsymbol{R}_a && (4.28) \\ \iff & \boldsymbol{\Lambda}_S \boldsymbol{P}_S^\top \boldsymbol{Z}_a \boldsymbol{P}_T + \boldsymbol{P}_S^\top \boldsymbol{Z}_a \boldsymbol{P}_T \boldsymbol{\Lambda}_T = \tau \boldsymbol{P}_S^\top \boldsymbol{R}_a \boldsymbol{P}_T && (4.29) \\ \iff & ((\mathbf{I} \otimes \boldsymbol{\Lambda}_S) + (\boldsymbol{\Lambda}_T \otimes \mathbf{I}))\, \mathbf{vec}(\boldsymbol{P}_S^\top \boldsymbol{Z}_a \boldsymbol{P}_T) = \tau \mathbf{vec}(\boldsymbol{P}_S^\top \boldsymbol{R}_a \boldsymbol{P}_T) && (4.30) \\ \iff & ((\mathbf{I} \otimes \boldsymbol{\Lambda}_S) + (\boldsymbol{\Lambda}_T \otimes \mathbf{I}))\, \mathbf{vec}(\boldsymbol{Z}') = \tau \mathbf{vec}(\boldsymbol{P}_S^\top \boldsymbol{R}_a \boldsymbol{P}_T) \text{ and } \boldsymbol{Z}_a = \boldsymbol{P}_S \boldsymbol{Z}' \boldsymbol{P}_T^\top && (4.31) \end{aligned}$$

Since $\boldsymbol{\Lambda}_S$ and $\boldsymbol{\Lambda}_T$ are diagonal, so is $(\mathbf{I} \otimes \boldsymbol{\Lambda}_S) + (\boldsymbol{\Lambda}_T \otimes \mathbf{I})$, and system (4.31) can be solved efficiently.

Observe that throughout the CG iterations, the linear system (4.26) always has the same structure, and only the right-hand side $\boldsymbol{R}_a$ changes. This implies that we can precompute the matrices $\boldsymbol{P}_S, \boldsymbol{P}_T, \boldsymbol{\Lambda}_S, \boldsymbol{\Lambda}_T$ before the start of the IRLS algorithm 15 and keep them in memory. The main numerical bottleneck then lies in the computation of the matrix products $\boldsymbol{P}_S^\top \boldsymbol{R}_a \boldsymbol{P}_T$ and $\boldsymbol{P}_S \boldsymbol{Z}' \boldsymbol{P}_T^\top$.

4.4.3.1 Equivalence of preconditioned CG with better-conditioned linear system

We point out that while using CG for singular systems is a well studied task (see [Kaa88, Hay18] and remark 4.4.3), in our case the preconditioner $\boldsymbol{D}$ is also singular, but we claim that this is not an issue with the current choice of the preconditioner. The main reason is that both $\boldsymbol{A}$ and $\boldsymbol{D}$ share the same one-dimensional nullspace, and that r_k is always in the range of $\boldsymbol{A}$, and thus of $\boldsymbol{D}$, so that the linear system on line 4 of Algorithm 16 is always well-defined.

More precisely, let us write the eigendecompositions of $\boldsymbol{A}$ and $\boldsymbol{D}$ as

$$\boldsymbol{A} = \sum_{i=1}^{n} \lambda_i q_i q_i^\top, \ \boldsymbol{D} = \sum_{i=1}^{n} \gamma_i d_i d_i^\top,$$

where $\{q_i\}_{i=1}^n$ and $\{d_i\}_{i=1}^n$ are orthonormal bases of $\mathbb{R}^n$. Since $\boldsymbol{A}$ and $\boldsymbol{D}$ have the same one-dimensional nullspace, we assume without loss of generality that $\lambda_1 = \gamma_1 = 0$ (namely that $\boldsymbol{A}q_1 = 0$ and $\boldsymbol{D}d_1 = 0$), and that $q_1 = d_1$. We then get that $\lambda_i > 0$ and $\gamma_i > 0$ for all $i \geq 2$. Let us then define

$$\boldsymbol{C} = \sum_{i=2}^{n} \sqrt{\gamma_i} d_i d_i^\top, \ \boldsymbol{C}^* = \sum_{i=2}^{n} \frac{1}{\sqrt{\gamma_i}} d_i d_i^\top.$$

For simplicity, we will write $x_l = \begin{pmatrix} u^l \\ v^{v,l} \\ v^{h,l} \end{pmatrix} \in \mathbb{R}^n$.

Proposition 4.4.3. *Solving*

$$\boldsymbol{A}x = b \tag{4.32}$$

with conjugate gradient with preconditioner $\boldsymbol{D}$ starting from x_0 is equivalent to solving the linear system

$$\boldsymbol{C}^*\boldsymbol{A}\boldsymbol{C}^*\tilde{x} = \boldsymbol{C}^* b \tag{4.33}$$

with conjugate gradient without a preconditioner starting from $\tilde{x}_0 = \boldsymbol{C}x_0$

This is a known result when the matrix $\boldsymbol{D}$ is non-singular, and our proof is basically the same. We provide it for completeness in Appendix 4.B.

4.4.3.2 Numerical benefits of the preconditioner

To conclude this section, we focus on showing the numerical benefits of the preconditioner defined above. Before we do so, let us recall two important properties about the convergence of CG. Define

$$\rho(\boldsymbol{A}) := \frac{\sqrt{\kappa(\boldsymbol{A})} - 1}{\sqrt{\kappa(\boldsymbol{A})} + 1}$$

where $\kappa(\boldsymbol{A})$ is the condition number of $\boldsymbol{A}$, namely the ratio of the largest eigenvalue over the smallest strictly positive eigenvalue. Then the residual $r_l = \boldsymbol{A}x_l - b$ converges to 0 at the rate $\rho(\boldsymbol{A})^l$ [Hay18]. Now, for illustrative purposes, assume $\boldsymbol{A}$ is invertible. Then, for any polynomial Q_l of degree l such that $Q_l(0) = 1$, we have

$$\frac{\|\boldsymbol{A}x_l - b\|_{\boldsymbol{A}^{-1}}}{\|\boldsymbol{A}x_0 - b\|_{\boldsymbol{A}^{-1}}} \leq \max_{z \in \sigma(\boldsymbol{A})} |Q_l(z)|, \tag{4.34}$$

where $\sigma(\boldsymbol{A})$ is the set of eigenvalues of $\boldsymbol{A}$ [Kel95, Corollary 2.2.1]. The same results hold for the convergence of the preconditioned system $\boldsymbol{C}^*\boldsymbol{A}\boldsymbol{C}^*\tilde{x} = \boldsymbol{C}^*b$, in particular by defining $\rho(\boldsymbol{C}^*\boldsymbol{A}\boldsymbol{C}^*)$ similarly.

The convergence bound (4.34) highlights the fact that clustered eigenvalues can significantly help the convergence of CG. Indeed, the more clustered the eigenvalues are, the easier it is to find a polynomial Q_l which has low absolute value for all of them.

Although our matrices $\boldsymbol{A}$ and $\boldsymbol{C}^*\boldsymbol{A}\boldsymbol{C}^*$ are not invertible, we hypothesize that a result similar to (4.34) holds for singular matrices (just like it is true that the convergence rate in $\rho(\boldsymbol{A})^l$ holds for singular matrices $\boldsymbol{A}$ [Hay18]). We shift our focus to the comparison of the distribution of the (strictly positive) eigenvalues and of the condition number of $\boldsymbol{A}$ and $\boldsymbol{C}^*\boldsymbol{A}\boldsymbol{C}^*$. To do so, we generate matrices $\boldsymbol{D}^{\boldsymbol{v}}$ and $\boldsymbol{D}^{\boldsymbol{h}}$ with entries uniformly distributed between 0 and $1/\delta$ (since, assuming $\boldsymbol{C}^{\boldsymbol{v}}$ and $\boldsymbol{C}^{\boldsymbol{h}}$ are unity, this is the interval in which the entries of $\boldsymbol{D}^{\boldsymbol{v}}$ and $\boldsymbol{D}^{\boldsymbol{h}}$ lie). With those, we construct $\boldsymbol{A}$ and $\boldsymbol{D}$ from (4.21) and (4.24) respectively. We set $\delta = 10^{-6}$ as in the experimental section 4.5.

We plot in Figure 4.1 the spectrum of the matrix $\boldsymbol{A}$ and the spectrum of the matrix $\boldsymbol{C}^*\boldsymbol{A}\boldsymbol{C}^*$ for $N = M = 16$. We observe a nice clustering of the eigenvalues of $\boldsymbol{C}^*\boldsymbol{A}\boldsymbol{C}^*$ around 1. On the other hand, the distribution of the eigenvalues of $\boldsymbol{A}$ seem to be almost continuous from about 1 to over 10^5.

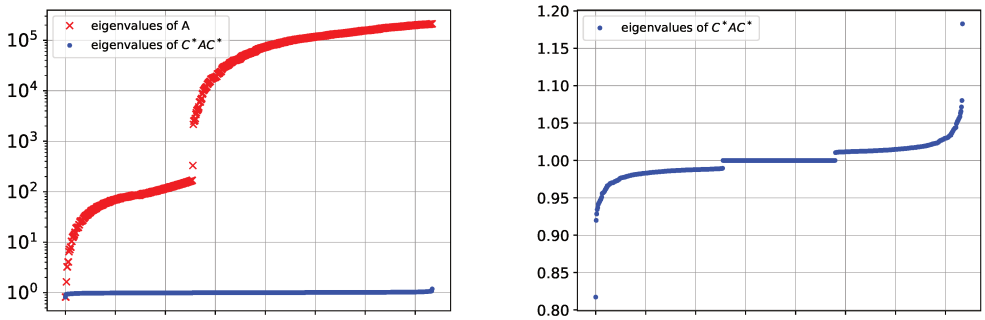

Figure 4.1: Left: Distribution of the eigenvalues of matrices $\boldsymbol{A}$ and $\boldsymbol{C}^*\boldsymbol{A}\boldsymbol{C}^*$ for $N = M = 16$ in log scale. Right: distribution of the eigenvalues of matrix $\boldsymbol{C}^*\boldsymbol{A}\boldsymbol{C}^*$ only. We observe that the eigenvalues of the preconditioned matrix are significantly more clustered.

Figure 4.2 plots both the condition number and the values of ρ for $\boldsymbol{A}$ and $\boldsymbol{C}^*A\boldsymbol{C}$, for different values of N. We observe that as the dimension increases, the condition number of $\boldsymbol{A}$ explodes and the value of ρ is very close to 1. For the matrix $\boldsymbol{C}^*\boldsymbol{A}\boldsymbol{C}^*$, the conditioning is much better.

Remark 4.4.4. *Note that the values of N used in the above plots are relatively small compared to the ones encountered in practical applications. We could not go to higher values because of numerical bottlenecks. Indeed, computing $\boldsymbol{C}^*$ first involves computing the eigendecomposition of $\boldsymbol{D}$. To get the above plots, one then needs to compute the eigenvalues of the dense matrix $\boldsymbol{C}^*\boldsymbol{A}\boldsymbol{C}^*$. Even for $N = M = 100$, the size of the matrices $\boldsymbol{A}$, $\boldsymbol{D}$ and $\boldsymbol{C}^*$ is 29800×29800. This proved to be too slow in practice, and we believe that this section still illustrates well the benefits of our preconditioner.*

Remark 4.4.5. *We are finally able to explain why it is numerically interesting to use quadratic penalties instead of hard constraints in problem (4.5). Indeed, it would be possible to derive the IRLS and CG algorithms of sections 4.4.1 and 4.4.2 starting from problem (4.4). The resulting vectorized linear system to be solved in the IRLS algorithm would then have the form*

$$((\mathbf{I} \otimes \boldsymbol{S})^\top \mathbf{diag}(\mathbf{vec}(\frac{1}{\boldsymbol{W}^{\boldsymbol{v}}}))(\mathbf{I} \otimes \boldsymbol{S}) + (\boldsymbol{T} \otimes \mathbf{I})\mathbf{diag}(\mathbf{vec}(\frac{1}{\boldsymbol{W}^{\boldsymbol{h}}}))(\boldsymbol{T} \otimes \mathbf{I})^\top)u = b',$$

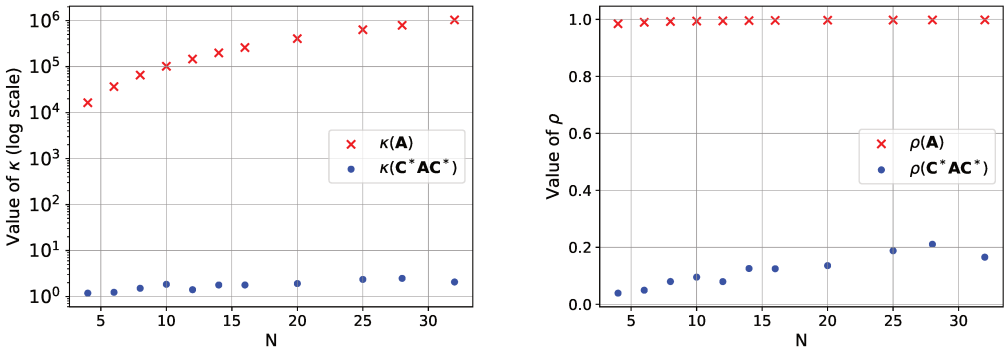

Figure 4.2: Left: Value of the condition numbers of the original and preconditioned linear systems (in semilog scale), for different values of N. Right: Value of CG convergence rate ρ for both original and preconditioned linear system, for different values of N. As we can see, there is a significant improvement in the value of the condition number for the preconditioned system, and this translates into an improved value of ρ.

with $b' \in \mathbb{R}^n$ (we assume for simplicity that the weights $\boldsymbol{C^v}$ and $\boldsymbol{C^h}$ are unit weights). This results in the matrices $\boldsymbol{W^v}$ and $\boldsymbol{W^h}$ being 'sandwiched' between two matrices, namely $(\mathbf{I} \otimes \boldsymbol{S})$ and its transpose, and $(\boldsymbol{T} \otimes \mathbf{I})$ and its transpose, respectively. As the IRLS algorithm makes progress, the linear system becomes more and more ill-conditioned, and we were not able to find good preconditioners to speed up the conjugate gradient iterates. In contrast, in matrix $\boldsymbol{A}$ defined in (4.21), the blocks in $\boldsymbol{W^v}$ and $\boldsymbol{W^h}$ and the block of the form $(\mathbf{I} \otimes \boldsymbol{S}^\top \boldsymbol{S}) + (\boldsymbol{T}\boldsymbol{T}^\top \otimes \mathbf{I})$ are separate and form the diagonal, making the preconditioning much easier and efficient. A similar observation was made in [FPRW16], and we refer the interested reader to this work for more details on the numerical benefits of the quadratic penalties.

4.5 Experiments

We test our algorithm on both synthetic and real topographic data sets. The synthetic data allows us to verify that the output of our algorithm is accurate. For both synthetic and real data, we compare running time and solution quality of our method with the output of SNAPHU [CZ00, CZ01, CZ02]. SNAPHU is a widely used open-source software for phase unwrapping. The default algorithm is the statistical-cost, network-flow algorithm, which poses the phase unwrapping as a maximum a posteriori probability problem. SNAPHU also implements solvers for L^p-minimization phase unwrapping problem. In particular, the L^1-norm problem is solved using a modified network-simplex solver.

Whether SNAPHU solves the statistical-cost problems or L^p-norm problems, it is initialized with either a minimum cost flow (MCF) algorithm [Cos98] or a minimum spanning tree (MST) algorithm [CZ00]. It also offers the option to return the solution after only running the initialization.

Note that MCF is another way of solving the L^1-norm problem. As noted in the SNAPHU documentation, the results of MCF and of the SNAPHU L^1-norm solver may be different, although in theory both should be L^1 optimal.

We shall therefore compare the IRLS algorithm against the five following SNAPHU options: MCF, MST, L^1-norm minimization with MCF initialization, L^1-norm minimization with MST initialization, statistical-cost with MCF initialization and statistical-cost with MST initialization, the latter being the default SNAPHU option.

IRLS experiments are ran on a Nvidia RTX8000 48 GB GPU, and the SNAPHU experiments on a Intel(R) Xeon(R) CPU E5-2650.

4.5.1 Real Data Acquisition and Synthetic Data Generation

We now describe the Sentinel-1 InSAR dataset used in our experiments. For clarity, recall that the main geometric effects present in the interferogram between images 1 and 2 are the orbital o_{12} and ξ_{12} topographic components for which their unwrapped phase value can be approximated by [Han02]:

$$o_{12} = \frac{-2m\pi}{\lambda} B_{12}^{\parallel} \text{ (rad)}, \tag{4.35}$$

$$\xi_{12} = \frac{-2m\pi}{\lambda} \frac{B_{12}^{\perp} h}{R_{st} \sin\theta} \text{ (rad)}, \tag{4.36}$$

where λ is the radar carrier wavelength, $B_{12}^{\parallel}$ and $B_{12}^{\perp}$ correspond to the parallel and orthogonal baseline respectively (related to the separation of the two orbit passes in the direction parallel/orthogonal to the line of sight), R_{st} is the satellite-target distance also known as the range, θ is the incidence angle and m is equal to 2 to account for the two-way wave propagation in Sentinel-1. A real Sentinel-1 interferogram also contains contributions related to the deformation between the two dates and the atmospheric propagation delays and is also impacted by phase noise and wrapping.

For the synthetic data generation, we are only interested in the topographic component ξ_{12} because of the simplicity of the model given in (4.36). Therefore, we use ten pairs of Sentinel-1 Synthetic Aperture Radar (SAR) images covering different locations in the world, chosen to exhibit variations in topography. To maximize the effect of topography ξ_{12}, we choose image pairs with large $B_{12}^{\perp}$, and we also minimize the potential deformation contributions by taking dates that are temporally close, as shown in Table 4.1. We then adopt the standard simulation procedure of a differential InSAR processing chain to compute o_{12} and ξ_{12}. In particular, we use the metadata of the two acquisitions in the image pair, along with their precise orbit auxiliary metadata, to obtain their respective camera models. The camera model allows us to compute the 2D position in the image of a 3D point. We use the camera model and the SRTM30 [FRC+07] Digital Elevation Model (DEM) to estimate $B_{12}^{\parallel}$, $B_{12}^{\perp}$, h, and θ for all pixels. Values of λ and R_{st} can be obtained by simple computations from the metadata, and (4.35) and (4.36) are used to get o_{12} and ξ_{12}. Kayrros' toolbox for SAR data ("EOS-SAR") was used to perform the different computations. Note that when transforming the DEM into radar coordinates for estimating h, we adopt a mesh interpolation backgeocoding approach [LCRCPIP21]. This induces some small interpolation artifacts in the form of a triangular mesh. Nevertheless, it is interesting to use ξ_{12} for the synthetic data because it contains discontinuities related to terrain distortions due to the SAR acquisition geometry. In fact, besides phase noise, distortions such as layover and foreshortening are often cited as challenges for phase unwrapping [Han02].

On the other hand, after image alignment and burst stitching with the method described in [AAD+22], we compute the real interferograms compensated from the orbital component o_{12} with:

$$z_{12} = z_1 \cdot z_2^* \cdot e^{-jo_{12}}, \tag{4.37}$$

where z_1 and z_2 correspond to each complex SAR image. The interferometric phase is simply obtained by

$$\phi_{12} = \arg(z_{12}), \tag{4.38}$$

where arg(.) denotes the argument of the complex number. As previously explained, our choice of images is such that ϕ_{12} is dominated by ξ_{12}. Therefore, ϕ_{12} is a real wrapped noisy phase containing fringes mainly related to topography. Since it is standard practice to denoise the interferometric phase prior to unwrapping [Han02, Bra21], we apply a Goldstein phase filter [GW98] on the complex interferogram z_{12}, and the real denoised phase is given by

$$\widehat{\phi_{12}} = \arg(\widehat{z_{12}}), \tag{4.39}$$

Location	lon	lat	date 1	date 2	relorb	$B^{\perp}$(m)
Arz	36.5085	34.1601	20220615	20220627	21	100
Etna	14.6594	37.7404	20220411	20220505	124	-138
El Capitan	-120.0154	37.7357	20210710	20210728	144	-99
Kilimanjaro	37.1166	-2.9983	20180809	20180821	79	129
Mount Sinai	33.8923	28.5510	20230303	20230315	160	256
Korab	20.7207	41.8401	20220824	20220905	175	-271
Nevada	-119.2729	41.2154	20220518	20220530	144	156
Zeil	132.1604	-23.3499	20211209	20211221	2	-144
Wulonggou	96.0567	36.1646	20220824	20220905	172	-270
Warjan	65.0716	32.4197	20230118	20230130	42	-326

Table 4.1: Dataset definition: for each location given by its central coordinates, we process the images at the two dates for a given relative orbit. Contains modified Copernicus Sentinel data.

where $\widehat{z_{12}}$ is the Goldstein filtered intereferogram. The Goldstein filter is a patch-based method, where each patch is denoised in the Fourier domain via:

$$Z'(u,v) = Z(u,v)\left\{\left(\mathbf{1}_{N_f} * |Z|\right)(u,v)\right\}^{\alpha}, \tag{4.40}$$

where $Z(u,v)$ and $Z'(u,v)$ correspond to the input and output patch Fourier transforms respectively, $\mathbf{1}_{N_f}$ is a uniform filter of size N_f ($*$ is the convolution operation), and α is a factor between 0 and 1 that controls the amount of filtering. The patches are taken at a fixed step s, with a patch size equal to $4 \times s$, yielding an overlap of 75%. After the inverse Fourier transform of $Z'(u,v)$ is computed, the patches are recombined using a linear taper in the x and y dimensions. For our dataset, we set $\alpha = 1$, a step $s = 16$ pixels, and a uniform filter size $N_f = 5$.

The previously described process was repeated twice to obtain two datasets with different image sizes at full Sentinel-1 resolution. In both cases, we use the camera model to convert the centroid coordinates in Table 4.1 to the SAR image coordinates. For the first dataset, we compute crops of size 2048×2048 pixels centered on this location. For the second dataset, we compute the crops by stitching the full burst of the centroid with its previous and next burst in the same swath. This yields images of size about 4000×20000 pixels.

4.5.2 Weight computation

The weight matrices $\boldsymbol{C^v}$ and $\boldsymbol{C^h}$ are critical for the quality of the unwrapping. In classical SAR interferometry, the phase interferogram is usually provided along with the amplitudes A_1 and A_2 of the two acquired images, and a coherence map C. In our experiments, we use the statistical weights generated by SNAPHU, as they show very good practical performance and their computation is cheap compared to the running time of the algorithm. We therefore include the SNAPHU software in our distribution, and we slightly modify the original code to add the option of only computing the weights. For a detailed explanation of how those weights are generated, see [CZ01, Che01]

For the experiments with real images, the weights are calculated using the interferogram, amplitudes A_1 and A_2, and the coherence map C. For the experiments with simulated images, only the interferogram is provided.

4.5.3 Number of CG iterations and stopping criterion

The conjugate gradient method presented in Section 4.4.2 to solve the linear system (4.12) requires an explicit maximum number of iterations. We seek to balance running time of the

IRLS algorithm 15 with the quality of the output image. In particular, in early iterations of IRLS, it is often unnecessary to solve the linear system to a high precision to make significant progress on the value of the objective function, while higher precision is helpful in later iterations to refine image quality. This impacts the number of CG iterations at each IRLS step. One also needs to decide when to stop the IRLS algorithm as a whole. We propose a heuristic that aims at striking a good balance.

Let $m_{CG}(k)$ be the maximum number of iterations of CG allowed at iteration k of the IRLS algorithm. We start by setting $m_{CG}(0)$ to some predefined value. For $k \geq 1$, after each updates of the weight matrices $\boldsymbol{W}^{\boldsymbol{v},k}, \boldsymbol{W}^{\boldsymbol{h},k}$, we compute the relative improvement

$$\Delta_{k-1,k} = \frac{H_\delta(\boldsymbol{U}^k, \boldsymbol{V}^{\boldsymbol{v},k}, \boldsymbol{V}^{\boldsymbol{h},k}, \boldsymbol{W}^{\boldsymbol{v},k-1}, \boldsymbol{W}^{\boldsymbol{h},k-1}) - H_\delta(\boldsymbol{U}^k, \boldsymbol{V}^{\boldsymbol{v},k}, \boldsymbol{V}^{\boldsymbol{h},k}, \boldsymbol{W}^{\boldsymbol{v},k}, \boldsymbol{W}^{\boldsymbol{h},k})}{H_\delta(\boldsymbol{U}^k, \boldsymbol{V}^{\boldsymbol{v},k}, \boldsymbol{V}^{\boldsymbol{h},k}, \boldsymbol{W}^{\boldsymbol{v},k-1}, \boldsymbol{W}^{\boldsymbol{h},k-1})},$$

where H_δ is as defined in Theorem 4.4.3. We then increase the maximum number of CG iterations based on whether or not we deem the relative improvement $\Delta_{k-1,k}$ significant enough, comparing it to a user-defined tolerance ϵ_{tol}. If it is, then enough improvement was made by CG with the current number of maximum iterations, so we do not update $m_{CG}(k)$. If it is not, two cases arise. If the maximum number of CG iterations was already increased at the previous iteration, we consider that significant progress can no longer be made in the IRLS algorithm, so we stop and return $\boldsymbol{U}$. If the maximum number of CG iterations was not increased in the previous iteration, then it is possible that more CG iteration would lead to more progress, so we scale up m_{CG} by a constant $\alpha > 1$. We summarize this as follows:

1. If $\Delta_{k-1,k} > \epsilon_{tol}$, set $m_{CG}(k) = m_{CG}(k-1)$.
2. If $\Delta_{k-1,k} \leq \epsilon_{tol}$ and $m_{CG}(k-1) \neq m_{CG}(k-2)$, stop and return $\boldsymbol{U}^k$.
3. If $\Delta_{k-1,k} \leq \epsilon_{tol}$ and $m_{CG}(k-1) = m_{CG}(k-2)$, set $m_{CG}(k) = \alpha m_{CG}(k-1)$.

In the following experiments we have set $m_{CG}(0) = 5,\ \epsilon_{tol} = 10^{-3},\ \alpha = 1.7$.

4.5.4 Code details

The full code can be downloaded from https://github.com/bpauld/PhaseUnwrapping/tree/main. In this section we give a quick overview of the main parts of our code. The file **final_script.py** contains essentially the same steps. First we need a few imports.

```
import numpy as np
from python_code.parameters import ModelParameters
from python_code.parameters import IrlsParameters
from python_code.unwrap import unwrap
```

4.5.4.1 Loading data

Assuming the image to be unwrapped is contained in a .npy file called 'X.npy', simply load it

```
path_X = "X.npy" # replace with appropriate path
X = np.load(path_X)
```

4.5.4.2 Defining weighting strategy

The next step consists in defining $\boldsymbol{C}^{\boldsymbol{v}}$ and $\boldsymbol{C}^{\boldsymbol{h}}$. Several options are available. The first one is to provide user-defined weights.

```
path_Cv = "Cv.npy" # replace with appropriate path
path_Ch = "Ch.npy" # replace with appropriate path
Cv = np.load(path_Cv)
Ch = np.load(path_Ch)
```

Alternatively, if the user does not wish to supply precomputed weights, the values of the weights should be set to 'None', and the weighting strategy should be specified, as we will explain next.

```
Cv, Ch = None, None
```

One can then decide to use constant unity weights via two equivalent ways.

```
weighting_strategy = "uniform"
weighting_strategy = None # last two lines are equivalent
```

Alternatively, one can use weights generated from SNAPHU, as mentioned in Section 4.5.2. The quality of the weights can be further improved if two amplitude files and a correlation file are provided, although this is not mandatory.

```
weighting_strategy = "snaphu_weights"
path_amp1 = "amp1.npy" # replace with appropriate path
path_amp2 = "amp2.npy" # replace with appropriate path
path_corr = "corrfile.npy" # replace with appropriate path
amp1 = np.load(path_amp1)
amp2 = np.load(path_amp2)
corr = np.load(path_corrfile)
```

Note that if $\boldsymbol{C^v}$ and $\boldsymbol{C^h}$ are not set to **None**, their value will prevail over any weighting strategy specified and the algorithm will run with the supplied $\boldsymbol{C^v}$ and $\boldsymbol{C^h}$.

4.5.4.3 Defining model parameters

The parameters of the model, which correspond to the value of τ as introduced in (4.5) and the value of δ as introduced in (4.9), then need to be defined.

```
model_params = ModelParameters()
model_params.tau = 1e-2
model_params.delta = 1e-6
```

4.5.4.4 Defining IRLS parameters

Several parameters are necessary for the IRLS algorithm. Those are contained in the **IrlsParameters** class. The most important are related to the heuristics for updating the number of CG iterations described in section 4.5.3.

```
irls_params = IrlsParameters()
irls_params.max_iter_CG_strategy = "heuristics"
irls_params.max_iter_CG_start = 5
irls_params.rel_improvement_tol = 1e-3
irls_params.increase_CG_iteration_factor = 1.7
```

Other parameters can optionally be set, such as maximum number of IRLS iterations, maximum number of inner CG iterations, tolerance on the residue for stopping CG.

4.5.4.5 Unwrapping the image

Finally, the unwrapping is done by calling the corresponding function.

```
U, Vv, Vh = unwrap(X,
                   model_params=model_params,
                   irls_params=irls_params,
                   amp1=amp1,
                   amp2=amp2,
                   corr=corr,
                   weighting_strategy=weighting_strategy,
                   Cv=Cv, Ch=Ch)
```

4.5.5 Results

Recall that our model is invariant by a constant shift. To compute the error of our method, we therefore look for the shift that minimizes the norm between the original image and the shifted output. Namely, if the objective is an unwrapped image $\boldsymbol{X}_u \in \mathbb{R}^{N\times M}$, then for an output $\boldsymbol{U} \in \mathbb{R}^{N\times M}$ produced by our algorithm, we compute the error $E(\boldsymbol{U}, \boldsymbol{X}_u)$ as:

$$E(\boldsymbol{U}, \boldsymbol{X}_u) = \boldsymbol{X}_u - (\boldsymbol{U} + \alpha \mathbf{1}_{N\times M})$$

where

$$\alpha = \text{argmin}_{\alpha\in\mathbb{R}} \left\{ \|\boldsymbol{X}_u - (\boldsymbol{U} + \alpha \mathbf{1}_{N\times M})\|^2 \right\} = \frac{\sum_{\substack{1\leq i\leq N \\ 1\leq j\leq M}} (\boldsymbol{X}_u - \boldsymbol{U})_{i,j}}{NM}.$$

We start by exploring the performance of our proposed algorithm on simulated images such as the ones described in Section 4.5.1. This allows us to ensure that our method outputs an image which is consistent with the original simulated image, used as the ground truth to compute the error.

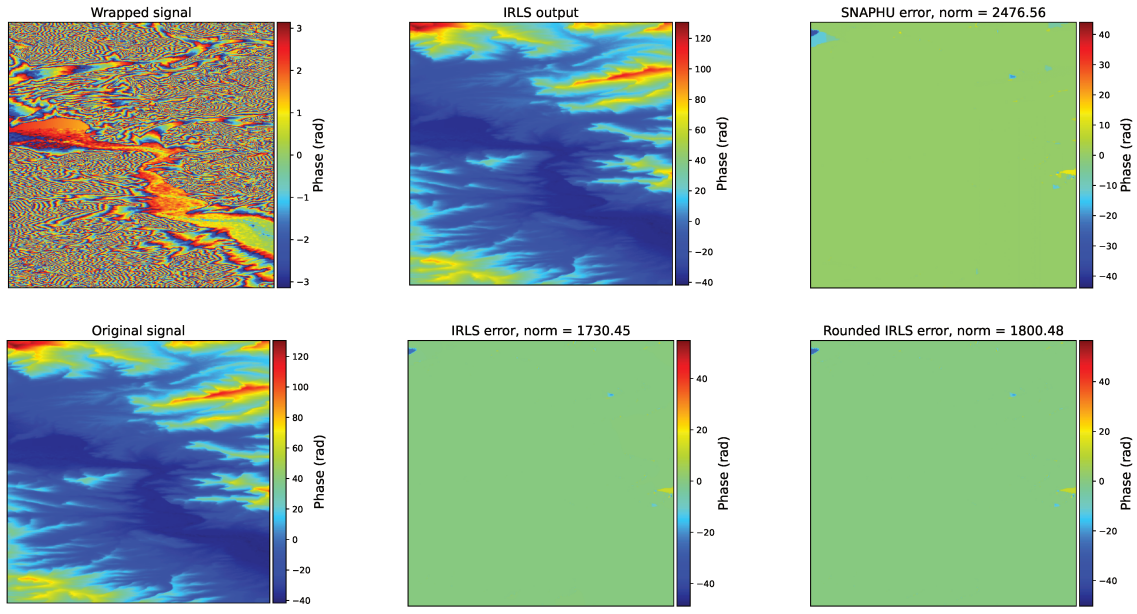

Figure 4.3: Comparison of the output of SNAPHU and IRLS on Warjan (Afghanistan) simulated image. Image size is 2048×2048.

Figure 4.3 shows the output of our method for a specific location in Afghanistan. We observe that our method is able to reconstruct the original image almost correctly, and that the error

made is comparable to the error made by the default SNAPHU method (MST + SC). We also explore the effect of rounding the IRLS output to enforce the output to be 2π-congruent to the input image. We call this output *rounded IRLS* in the plots. We observe that rounding does not seem to have significant effect on the performance of our method. Similar observations for different locations can be found in Appendix 4.C.1.

On real images, Figure 4.4 shows that our algorithm is similar to the SNAPHU outputs. In many cases, our solution is closest to the L^1-SNAPHU solution. Moreover, we observe that the MST solution yields many artefacts. Similar observations for different locations can be found in Appendix 4.C.2.

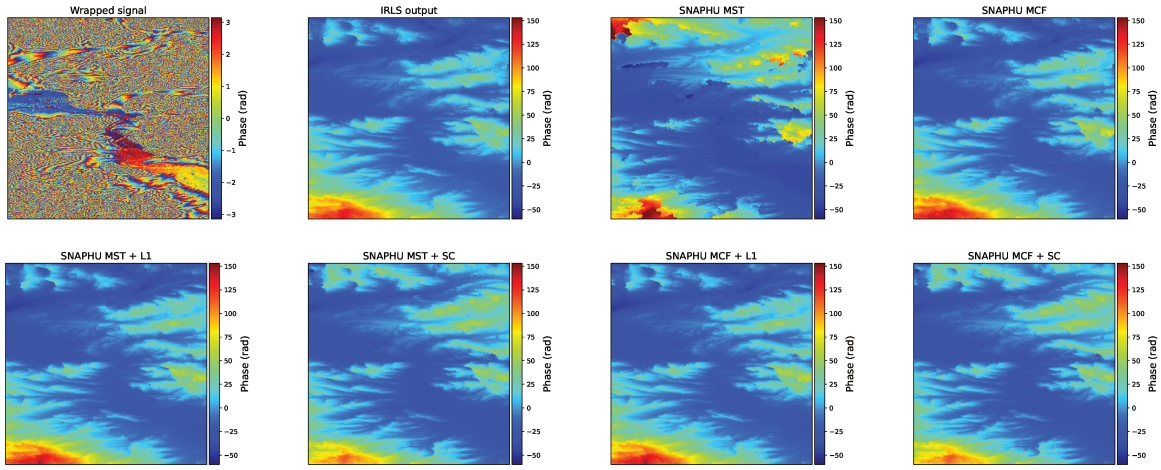

Figure 4.4: Comparison of the output of IRLS and of the different SNAPHU methods on Warjan (Afghanistan) image after Goldstein filtering. Image size is 2048×2048.

We now turn our attention to the running time of our method in comparison with the different methods implemented in SNAPHU in Table 4.2 for simulated images and in Table 4.3 for real images. We see that our algorithm significantly outperforms all other methods, except for the MST algorithm. However, MST yields poor-quality results on real images, and thus cannot be used as a full phase unwrapping method in practice, but only as an initialization procedure.

Table 4.2: Comparison of the running time (in seconds) of different phase unwrapping algorithms on **simulated** 2048×2048 *images. Our proposed method (IRLS) and the standard SNAPHU method (MST + SC) are in bold.*

Location	**IRLS**	MST	MCF	MST + L1	**MST + SC**	MCF + L1	MCF + SC
Arz	**13.36**	3.96	20.62	43.81	**44.33**	60.18	61.1
Etna	**24.12**	3.83	96.42	41.46	**42.93**	133.92	134.88
El Capitan	**24.2**	3.83	88.12	51.86	**67.94**	123.25	129.6
Korab	**31.81**	3.87	117.69	47.11	**275.84**	145.38	238.41
Kilimanjaro	**22.68**	3.94	95.86	44.22	**44.34**	134.04	139.04
Mount Sinai	**30.67**	3.96	92.87	26.02	**32.05**	112.53	120.94
Nevada	**7.13**	3.23	16.23	42.95	**43.76**	56.69	58.6
Zeil	**11.12**	3.81	20.68	39.03	**39.65**	55.48	58.22
Wulonggou	**32.0**	4.32	116.85	31.22	**194.32**	131.86	187.13
Warjan	**29.33**	4.22	112.37	22.0	**64.49**	127.0	172.47
Mean	**22.64**	3.9	77.77	38.97	**84.96**	108.03	130.04
Std	**8.63**	0.27	39.55	9.06	**77.92**	34.07	56.57

To end this section, we study how the running time scales with the size of the image, for our algorithm and SNAPHU (MST + SC). We unwrap images of size $4000 \times M$ for different values

Table 4.3: Comparison of the running time (in seconds) of different phase unwrapping algorithms on **real** 2048×2048 *images after Goldstein filtering. Our proposed method (IRLS) and the standard SNAPHU method (MST + SC) are in bold.*

Location	**IRLS**	MST	MCF	MST + L1	**MST + SC**	MCF + L1	MCF + SC
Arz	**22.98**	5.15	102.04	18.59	**18.82**	113.36	121.71
Etna	**24.15**	5.36	186.4	208.74	**513.65**	224.15	264.87
El Capitan	**23.26**	5.6	195.62	117.56	**146.91**	217.23	250.83
Korab	**22.78**	5.61	212.11	771.01	**1147.65**	325.68	384.24
Kilimanjaro	**25.37**	5.39	183.11	156.91	**231.65**	207.91	241.84
Mount Sinai	**17.75**	5.33	146.41	445.75	**844.97**	266.16	342.23
Nevada	**16.38**	5.22	113.12	14.75	**17.88**	121.66	131.02
Zeil	**22.82**	5.26	140.7	22.74	**36.67**	145.42	158.33
Wulonggou	**21.3**	5.4	226.34	1130.57	**2833.8**	289.99	796.35
Warjan	**21.21**	5.18	183.67	826.21	**1190.88**	295.21	540.06
Mean	**21.8**	5.35	168.95	371.28	**698.29**	220.68	323.15
Std	**2.65**	0.15	39.36	381.91	**833.35**	71.15	198.74

of M ranging from 2000 to 16000. We do so for both simulated images and Goldstein filtered real images. We use 10 simulated images and 4 real images. We only use 4 locations for real images because for the remaining ones, it was not possible to find large areas of high coherence, making the unwrapping impossible. Results are displayed in Figure 4.5. We not only observe a significant speedup in computing time, but also that our method scales better with the size of problem than SNAPHU.

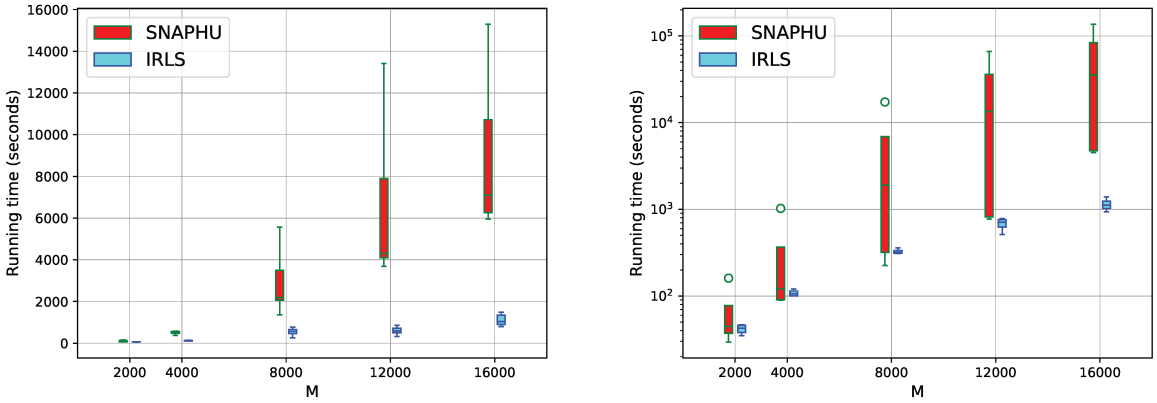

Figure 4.5: Box plot of the running times of IRLS and SNAPHU (MST + SC) on images of size $4000 \times M$ *for different values of* M*. Left: simulated images. Right: Goldstein filtered real images (log scale).*

4.A Matrix visualization

$$\mathbf{I}_M \otimes \boldsymbol{S} = \begin{pmatrix} -1 & 1 & & & & \\ 0 & -1 & 1 & & & \\ & 0 & -1 & \ddots & & \\ & & \ddots & \ddots & 1 & \\ & & & 0 & -1 & 1 \end{pmatrix} \in \mathbb{R}^{M(N-1)\times MN},$$

$$\mathbf{I}_M \otimes \boldsymbol{S}^\top \boldsymbol{S} = \begin{pmatrix} \boldsymbol{S}^\top \boldsymbol{S} & & \\ & \ddots & \\ & & \boldsymbol{S}^\top \boldsymbol{S} \end{pmatrix} \in \mathbb{R}^{MN\times MN},$$

with

$$\boldsymbol{S}^\top \boldsymbol{S} = \begin{pmatrix} 1 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & \ddots & & \\ & & \ddots & \ddots & -1 & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{pmatrix} \in \mathbb{R}^{N\times N},$$

and

$$\boldsymbol{T}\boldsymbol{T}^\top \otimes \mathbf{I}_N = \begin{pmatrix} \mathbf{I}_N & -\mathbf{I}_N & & & & \\ -\mathbf{I}_N & 2\mathbf{I}_N & -\mathbf{I}_N & & & \\ & -\mathbf{I}_N & 2\mathbf{I}_N & \ddots & & \\ & & \ddots & \ddots & -\mathbf{I}_N & \\ & & & -\mathbf{I}_N & 2\mathbf{I}_N & -\mathbf{I}_N \\ & & & & -\mathbf{I}_N & \mathbf{I}_N \end{pmatrix} \in \mathbb{R}^{MN\times MN},$$

$$\boldsymbol{T} \otimes \mathbf{I}_N = \begin{pmatrix} -\mathbf{I}_N & & & & \\ \mathbf{I}_N & -\mathbf{I}_N & & & \\ & \mathbf{I}_N & -\mathbf{I}_N & & \\ & & \ddots & \ddots & \\ & & & \mathbf{I}_N & -\mathbf{I}_N \\ & & & & \mathbf{I}_N \end{pmatrix} \in \mathbb{R}^{MN\times(M-1)N}$$

4.B Proof of Proposition 4.4.3

Proposition 4.4.3. *Solving*

$$\boldsymbol{A}x = b \tag{4.32}$$

with conjugate gradient with preconditioner $\boldsymbol{D}$ starting from x_0 is equivalent to solving the linear system

$$\boldsymbol{C}^*\boldsymbol{A}\boldsymbol{C}^*\tilde{x} = \boldsymbol{C}^* b \tag{4.33}$$

with conjugate gradient without a preconditioner starting from $\tilde{x}_0 = \boldsymbol{C}x_0$

Proof. We write plain letters for the iterates of CG to solve (4.32) with preconditioner $\boldsymbol{D}$, and use tilded letters for the iterates of CG without a preoconditioner to solve (4.33) starting from $\tilde{x}_0 = \boldsymbol{C}x_0$. We claim that $\tilde{x}_l = \boldsymbol{C}^* x_l$ for all $l \in \mathbb{N}$.

We have

$$\begin{aligned} r_0 &= b - \boldsymbol{A}x_0, \\ \tilde{r}_0 &= \boldsymbol{C}^* b - \boldsymbol{C}^* A \boldsymbol{C}^* \tilde{x}_0 = \boldsymbol{C}^*(b - \boldsymbol{A}\boldsymbol{C}^*\boldsymbol{C}x_0) = \boldsymbol{C}^*(b - \boldsymbol{A}x_0) = \boldsymbol{C}^* r_0 \end{aligned}$$

where the second to last inequality comes from the fact that $\boldsymbol{A}\boldsymbol{C}^*\boldsymbol{C} = \boldsymbol{A}$, which holds because $\boldsymbol{C}^*\boldsymbol{C} = \sum_{i=2}^n d_i d_i^\top = \mathbf{I}_n - d_1 d_1^\top$ and the fact that $\boldsymbol{A}d_1 = 0$.

Since by construction b is in the range of $\boldsymbol{A}$, we get that r_0 is in the range of $\boldsymbol{A}$ (this is because in our case, $\boldsymbol{A}$ and b come from the minimization of a least-squares problem). Since q_1 is the nullspace of $\boldsymbol{A}$, we get that $r_0^\top q_1 = r_0^\top d_1 = 0$ and the system $\boldsymbol{D}z_0 = r_0$ is well-defined. One solution, which is the one we choose in practice, is to set

$$z_0 = \left(\sum_{i=2}^n \frac{1}{\gamma_i} d_i d_i^\top \right) r_0 = \boldsymbol{C}^*\boldsymbol{C}^* r_0.$$

Moreover,

$$\tilde{z}_0 = \tilde{r}_0 = \boldsymbol{C}^* r_0 = \boldsymbol{C} z_0$$

since $\boldsymbol{C}z_0 = \boldsymbol{C}\boldsymbol{C}^*\boldsymbol{C}^* r_0$ and, as above, $\boldsymbol{C}\boldsymbol{C}^*\boldsymbol{C}^* = \boldsymbol{C}^*$, which holds because $\boldsymbol{C}\boldsymbol{C}^*\boldsymbol{C}^* = (\mathbf{I}_n - d_1 d_1^\top)\boldsymbol{C}^*$ and the fact that $d_1^\top \boldsymbol{C}^* = 0$. Then,

$$\begin{aligned} p_0 &= z_0 \\ \tilde{p}_0 &= \tilde{z}_0 = \boldsymbol{C}^* r_0 = \boldsymbol{C} z_0 = \boldsymbol{C} p_0 \\ \rho_0 &= z_0^\top r_0 \\ \tilde{\rho}_0 &= \tilde{z}_0^\top \tilde{r}_0 = z_0 \boldsymbol{C}\boldsymbol{C}^* r_0 = z_0^\top r_0 = \rho_0, \end{aligned}$$

since r_0 is in the range of $\boldsymbol{A}$, and thus of $\boldsymbol{C}\boldsymbol{C}^*$, and thus that $\boldsymbol{C}\boldsymbol{C}^* r_0 = r_0$. Then

$$\begin{aligned} \alpha_0 &= \frac{\rho_0}{p_0^\top \boldsymbol{A}^\top p_0} \\ \tilde{\alpha}_0 &= \frac{\tilde{\rho}_0}{\tilde{p}_0^\top \boldsymbol{C}^* \boldsymbol{A} \boldsymbol{C}^* \tilde{p}_0} = \frac{\rho_0}{p_0^\top \boldsymbol{C}\boldsymbol{C}^* \boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^* p_0}. \end{aligned}$$

As above we have that $p_0^\top \boldsymbol{C}\boldsymbol{C}^* \boldsymbol{A}\boldsymbol{C}\boldsymbol{C}^* p_0 = p_0^\top \boldsymbol{A} p_0$ so that $\tilde{\alpha}_0 = \alpha_0$. Then,

$$\begin{aligned} x_1 &= x_0 + \alpha_0 p_0, \\ \tilde{x}_1 &= \tilde{x}_0 + \alpha_0 \tilde{p}_0 = \boldsymbol{C}x_0 + \alpha_0 \boldsymbol{C} p_0 = \boldsymbol{C} x_1. \end{aligned}$$

Then

$$\begin{aligned} r_1 &= r_0 - \alpha_0 \boldsymbol{A} p_0 \\ \tilde{r}_1 &= \tilde{r}_0 - \alpha_0 \boldsymbol{C}^* \boldsymbol{A} \boldsymbol{C}^* \tilde{p}_0 = \boldsymbol{C}^*(r_0 - \alpha_0 \boldsymbol{A}\boldsymbol{C}^*\boldsymbol{C} p_0) = \boldsymbol{C}^* r_1. \end{aligned}$$

In particular, we get that r_1 is in the range of $\boldsymbol{A}$, and thus of $\boldsymbol{D}$. Therefore the system $\boldsymbol{D}z_1 = r_1$ is well-defined and we choose in practice to set

$$z_1 = \left(\sum_{i=2}^n \frac{1}{\gamma_i} d_i d_i^\top \right) r_1 = \boldsymbol{C}^*\boldsymbol{C}^* r_1.$$

Moreover, $\tilde{z}_1 = \tilde{r}_1 = \boldsymbol{C}^* r_1 = \boldsymbol{C} z_1$ by the same argument as above. Then

$$\begin{aligned}\rho_1 &= r_1^\top z_1 \\ \tilde{\rho}_1 &= \tilde{r}_1^\top \tilde{z}_1 = r_1^\top \boldsymbol{C}^* \boldsymbol{C} z_1 = r_1^\top z_1 = \rho_1,\end{aligned}$$

since again $\boldsymbol{C}^*\boldsymbol{C} = \mathbf{I}_n - d_1 d_1^\top$ ad r_1 is in the range of $\boldsymbol{A}$ and thus $r_1^\top d_1 = 0$. Thus $\beta = \tilde{\beta}$ and

$$\begin{aligned}p_1 &= z_1 + \beta p_0 \\ \tilde{p}_1 &= \tilde{z}_1 + \beta \tilde{p}_0 = \boldsymbol{C} p_1\end{aligned}$$

One can then apply the same reasoning at the next loop $l = 1$ and this shows that the iterates always satisfy $\tilde{x}_l = \boldsymbol{C} x_l$. □

4.C Additional experiments

4.C.1 Simulated images

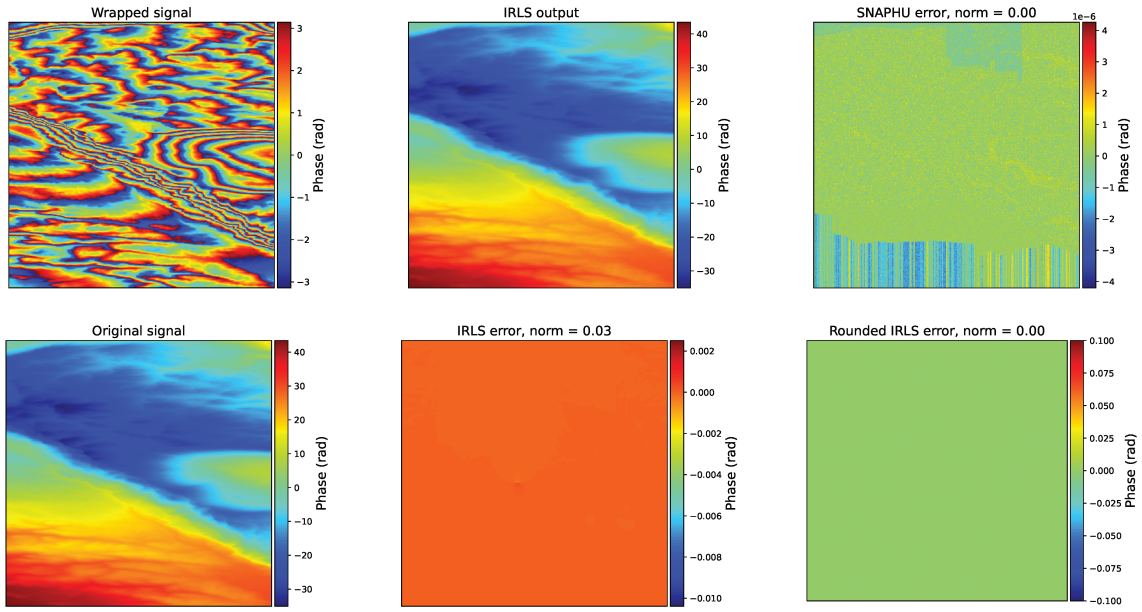

Figure 4.6: Comparison of the output of SNAPHU (MST + SC) and IRLS on Arz Lebanon simulated image. Image size is 2048×2048.

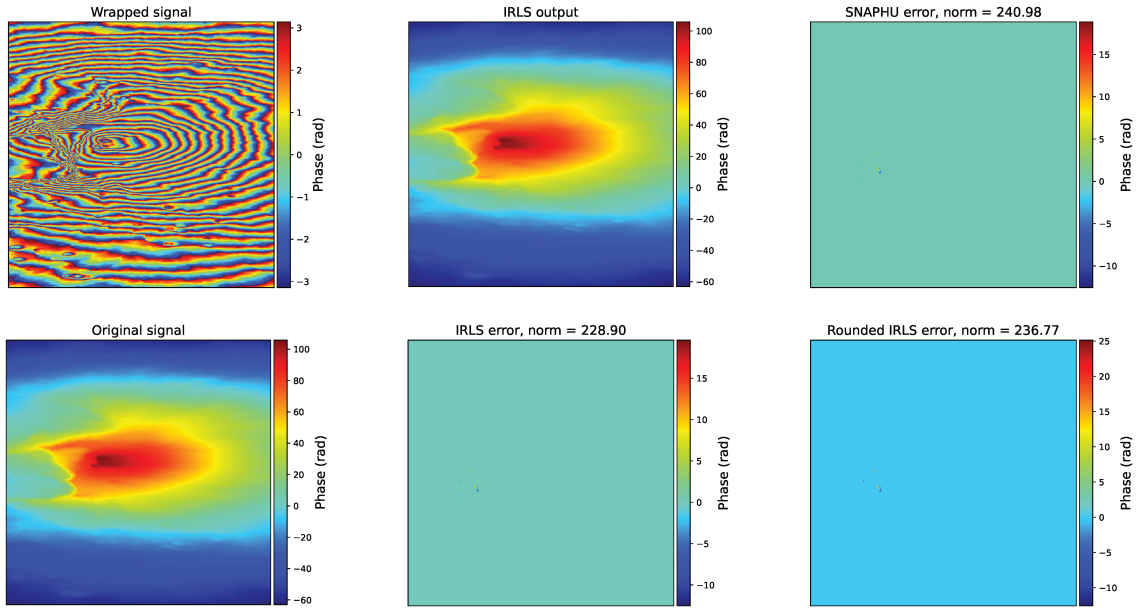

Figure 4.7: Comparison of the output of SNAPHU (MST + SC) and IRLS on Etna simulated image. Image size is 2048×2048.

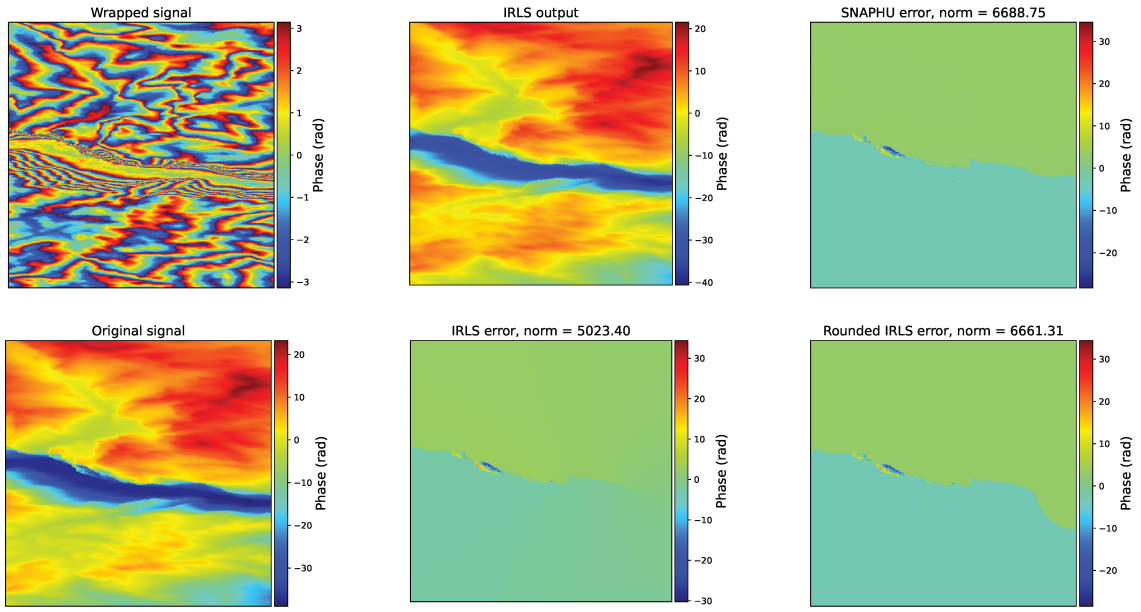

Figure 4.8: Comparison of the output of SNAPHU (MST + SC) and IRLS on El Capitan simulated image. Image size is 2048×2048.

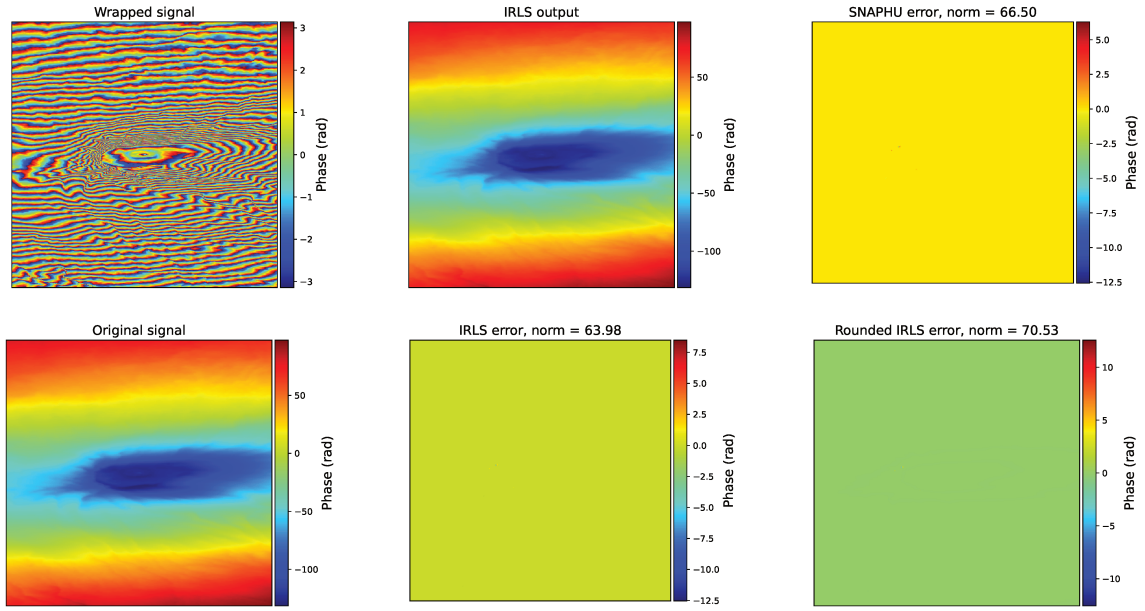

Figure 4.9: Comparison of the output of SNAPHU (MST + SC) and IRLS on Kilimanjaro simulated image. Image size is 2048×2048.

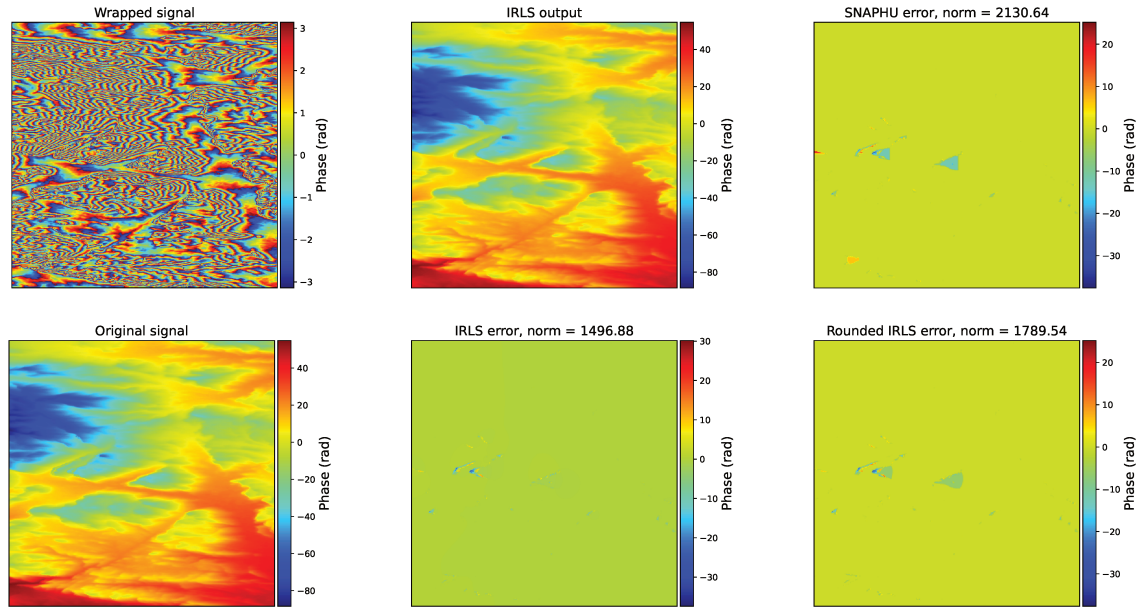

Figure 4.10: Comparison of the output of SNAPHU (MST + SC) and IRLS on Mount Sinai simulated image. Image size is 2048×2048.

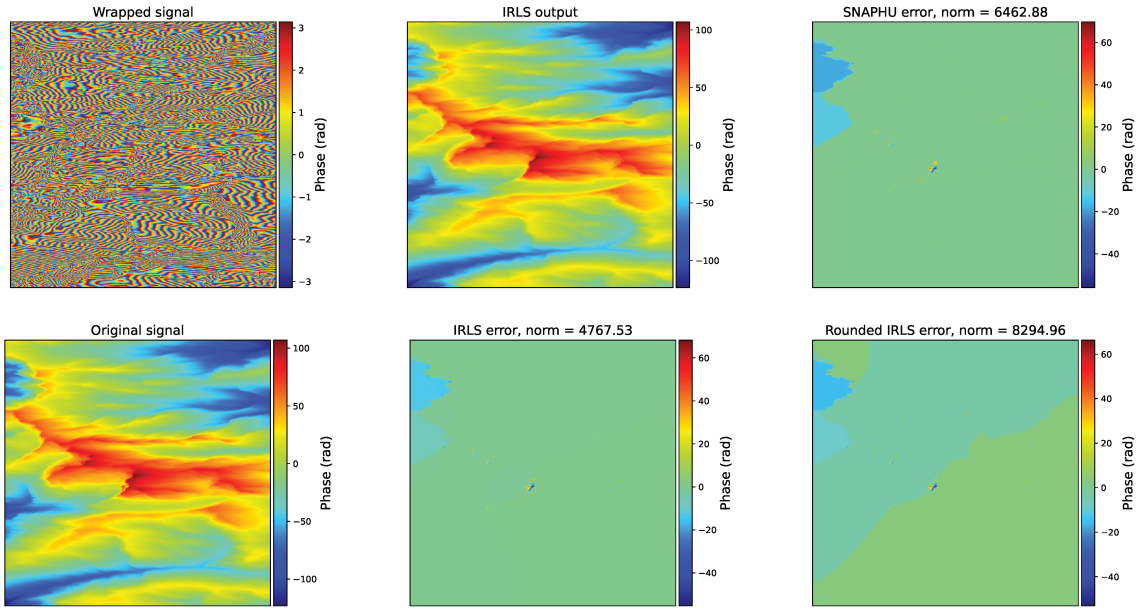

Figure 4.11: Comparison of the output of SNAPHU (MST + SC) and IRLS on Korab simulated image. Image size is 2048×2048.

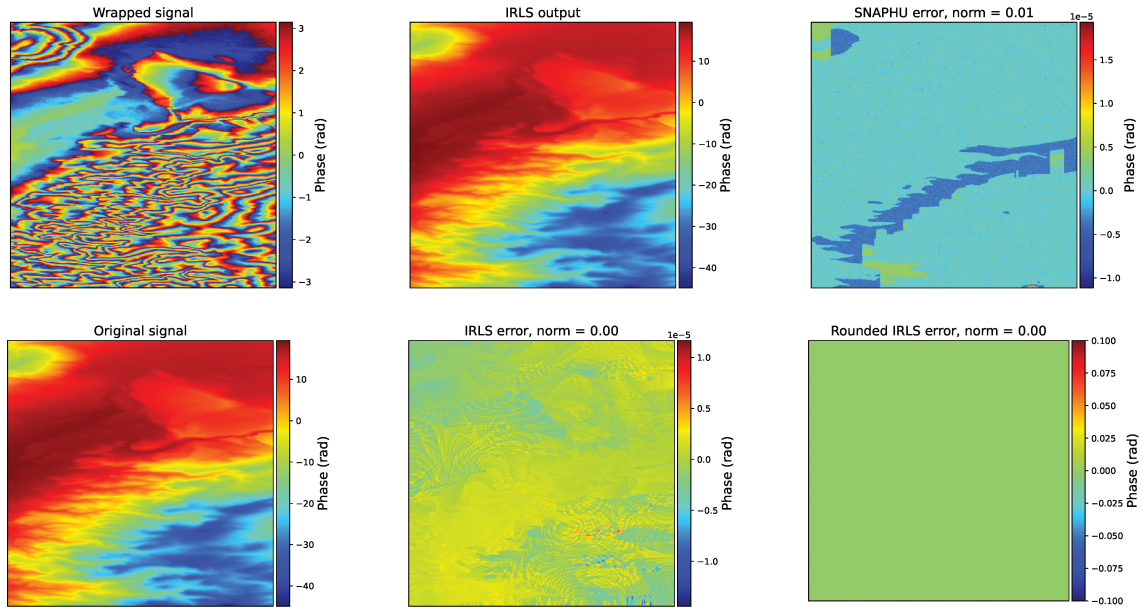

Figure 4.12: Comparison of the output of SNAPHU (MST + SC) and IRLS on Nevada simulated image. Image size is 2048×2048.

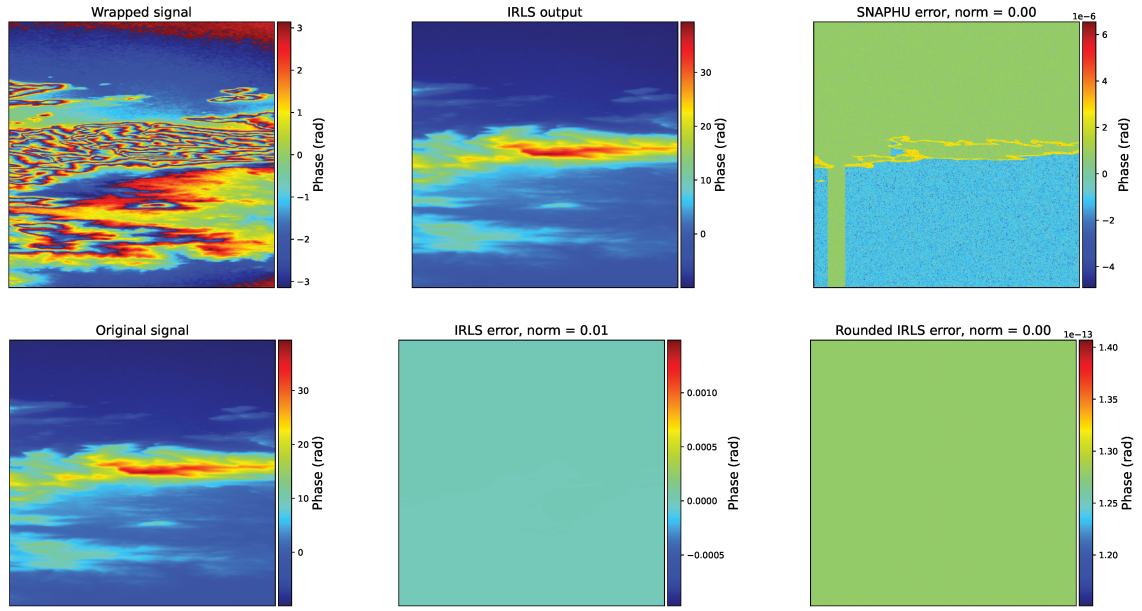

Figure 4.13: Comparison of the output of SNAPHU (MST + SC) and IRLS on Zeil (Australia) simulated image. Image size is 2048×2048.

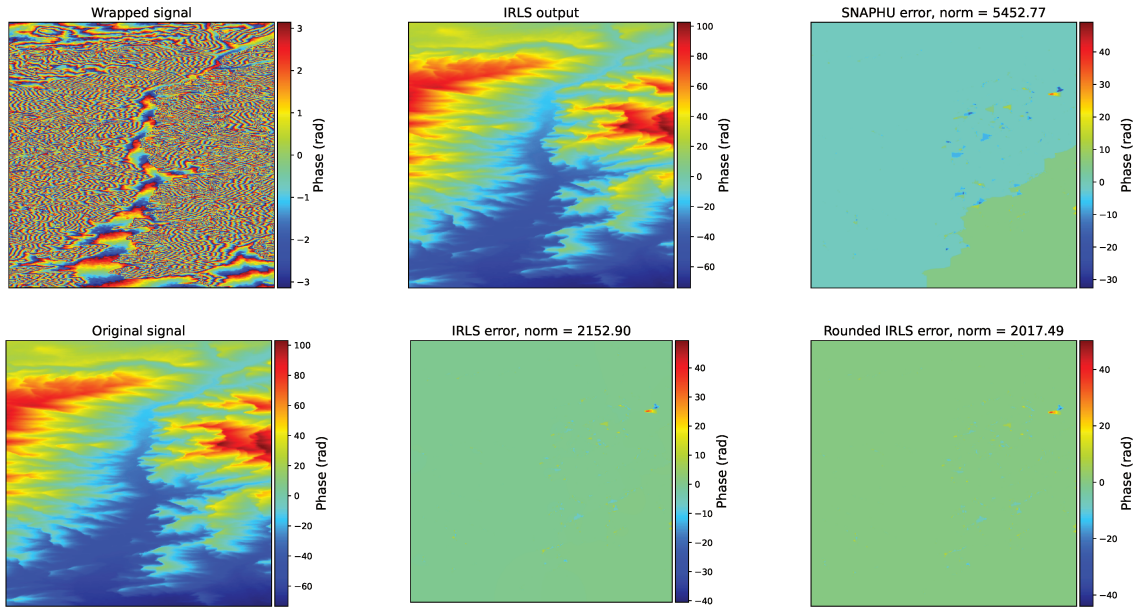

Figure 4.14: Comparison of the output of SNAPHU (MST + SC) and IRLS on Wulonggou (China) simulated image. Image size is 2048×2048.

4.C.2 Real images with Goldstein filtering

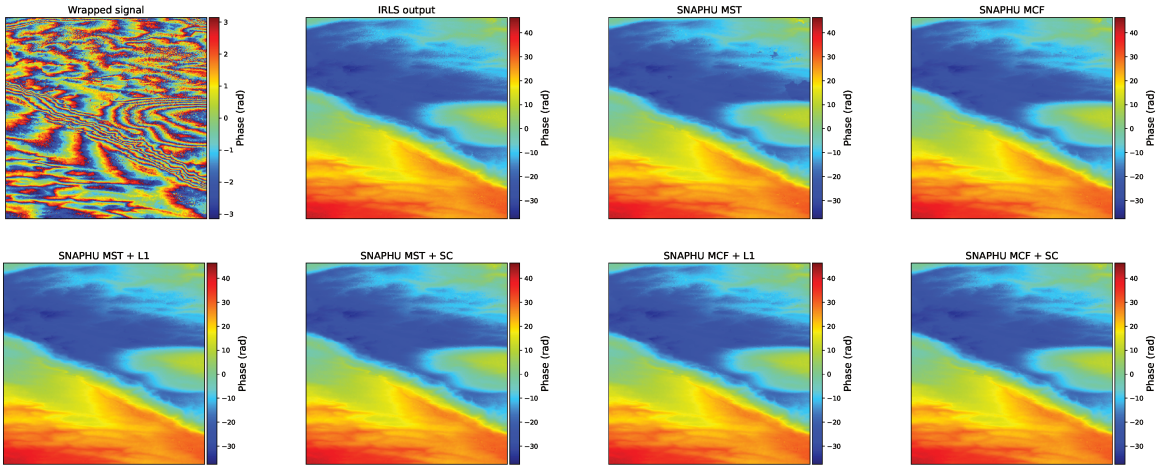

Figure 4.15: Comparison of the output of IRLS and of the different SNAPHU methods on Arz Lebanon real image after Goldstein filtering. Image size is 2048×2048.

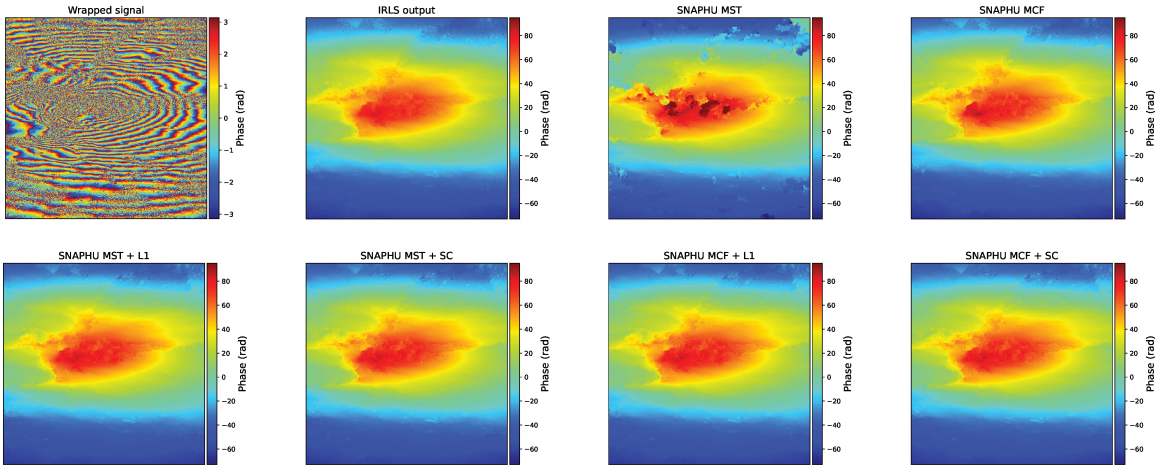

Figure 4.16: Comparison of the output of IRLS and of the different SNAPHU methods on Etna real image after Goldstein filtering. Image size is 2048×2048.

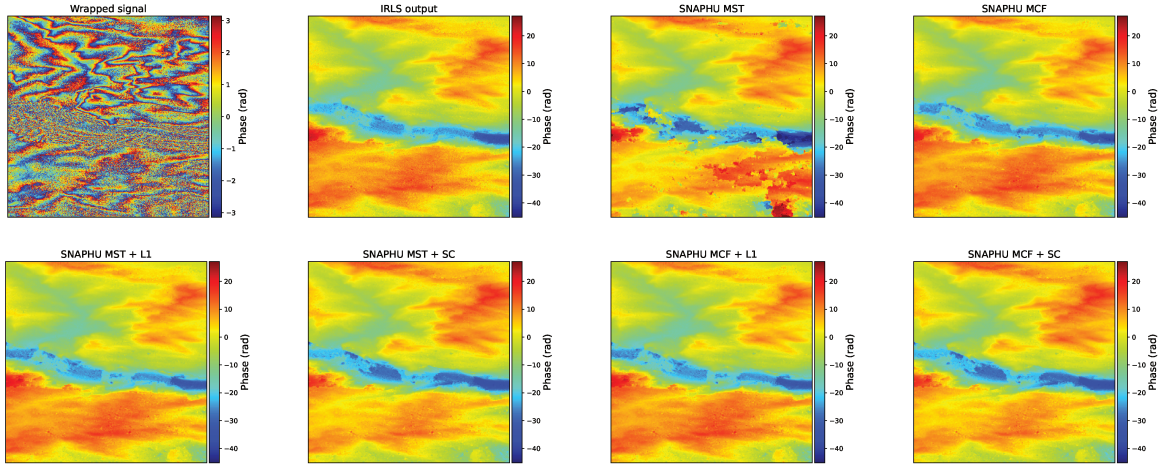

Figure 4.17: Comparison of the output of IRLS and of the different SNAPHU methods on El Capitan real image after Goldstein filtering. Image size is 2048×2048.

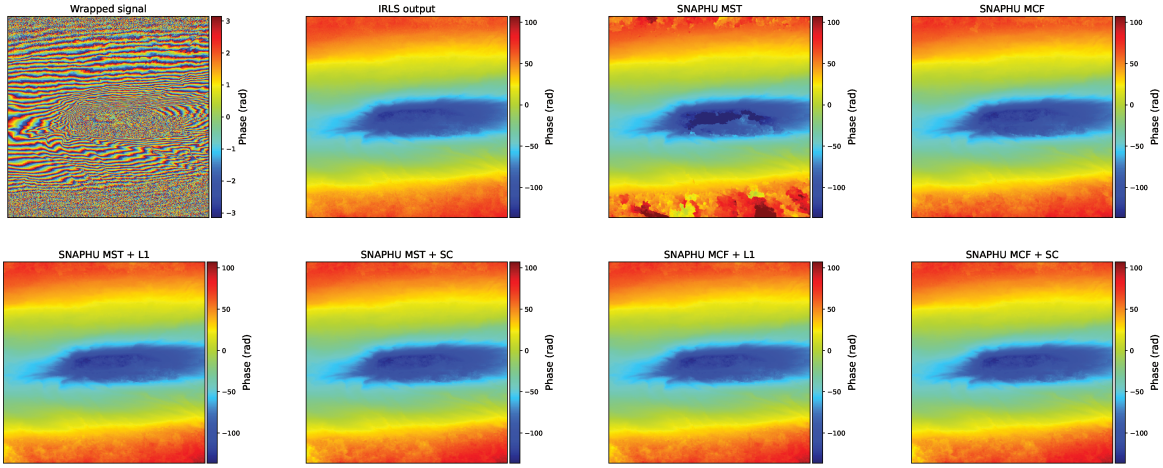

Figure 4.18: Comparison of the output of IRLS and of the different SNAPHU methods on Kilimanjaro real image after Goldstein filtering. Image size is 2048×2048.

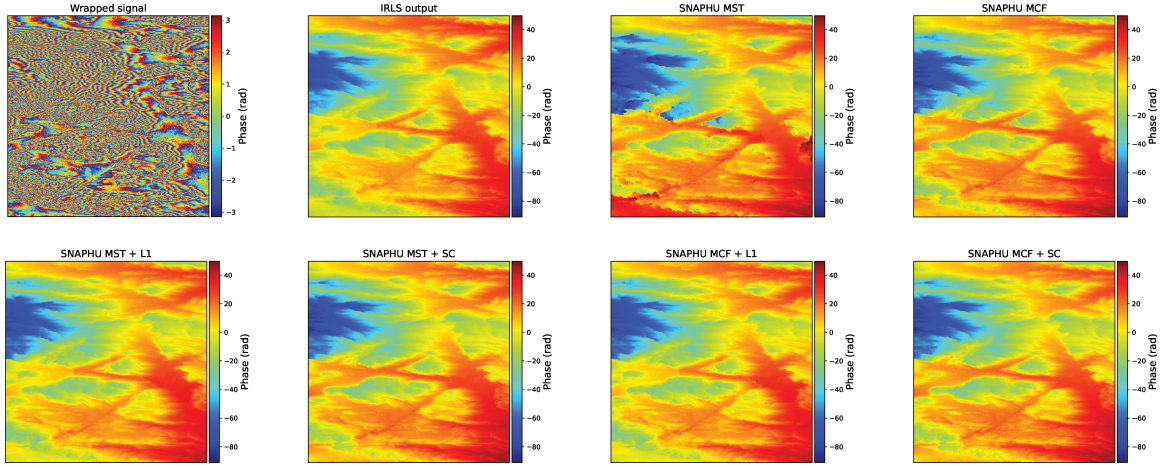

Figure 4.19: Comparison of the output of IRLS and of the different SNAPHU methods on Mount Sinai real image after Goldstein filtering. Image size is 2048×2048.

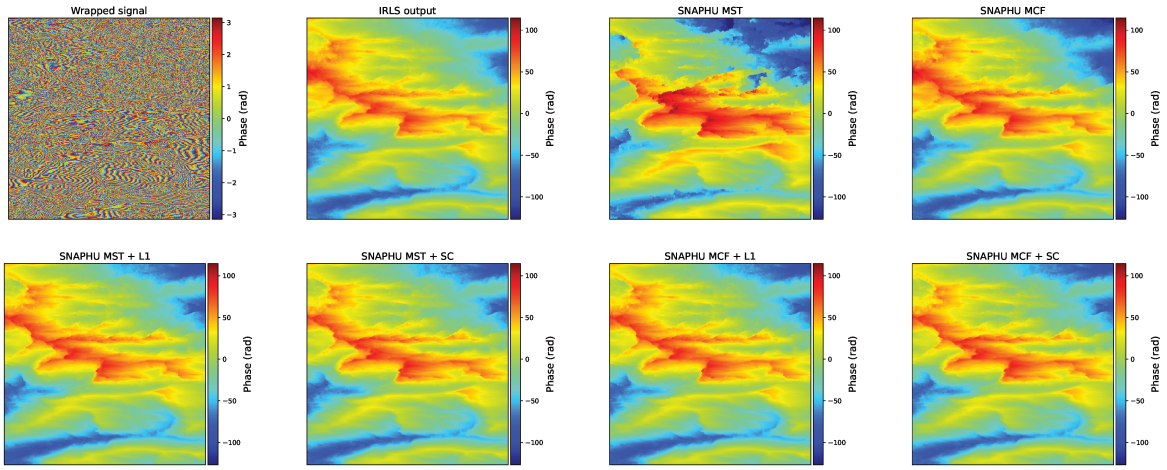

Figure 4.20: Comparison of the output of IRLS and of the different SNAPHU methods on Korab real image after Goldstein filtering. Image size is 2048×2048.

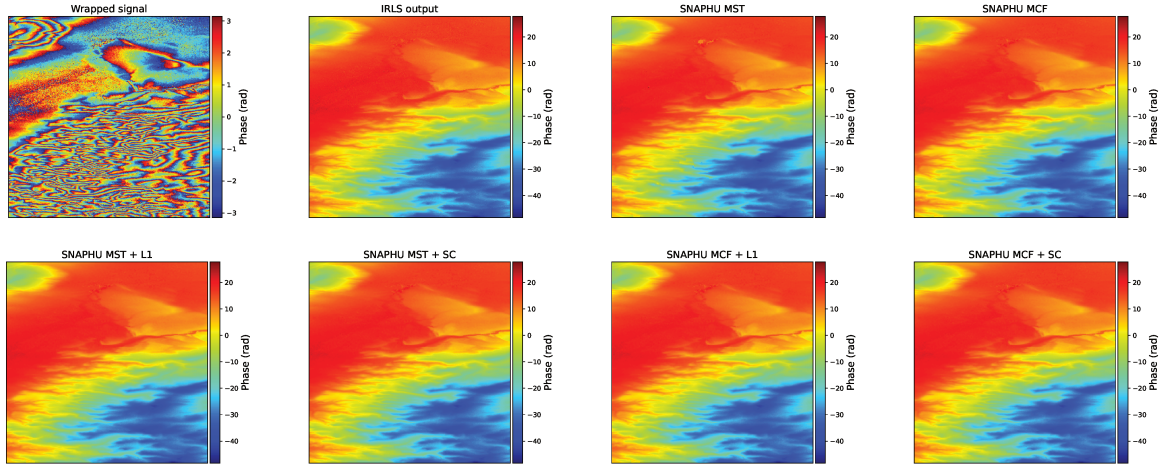

Figure 4.21: Comparison of the output of IRLS and of the different SNAPHU methods on Nevada real image after Goldstein filtering. Image size is 2048×2048.

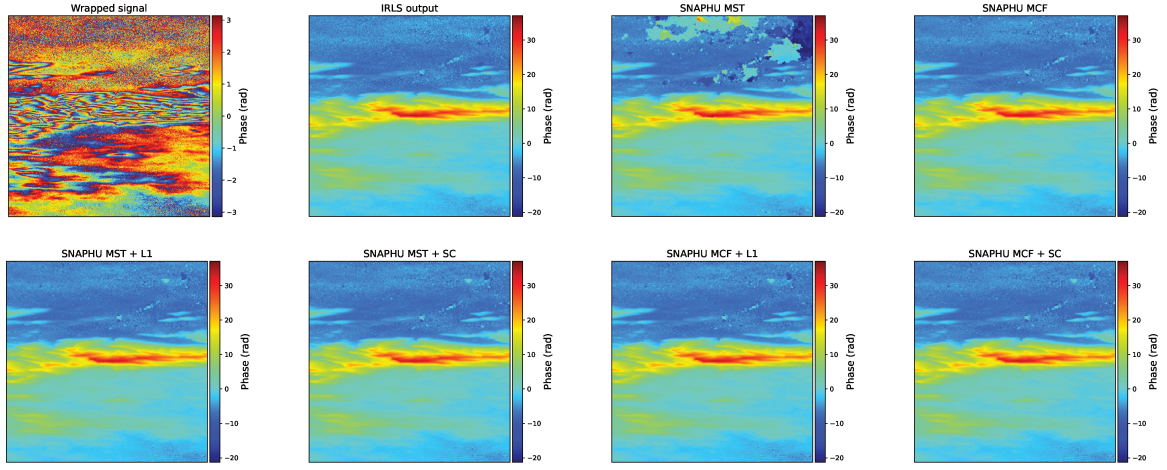

Figure 4.22: Comparison of the output of IRLS and of the different SNAPHU methods on Zeil (Australia) real image after Goldstein filtering. Image size is 2048×2048.

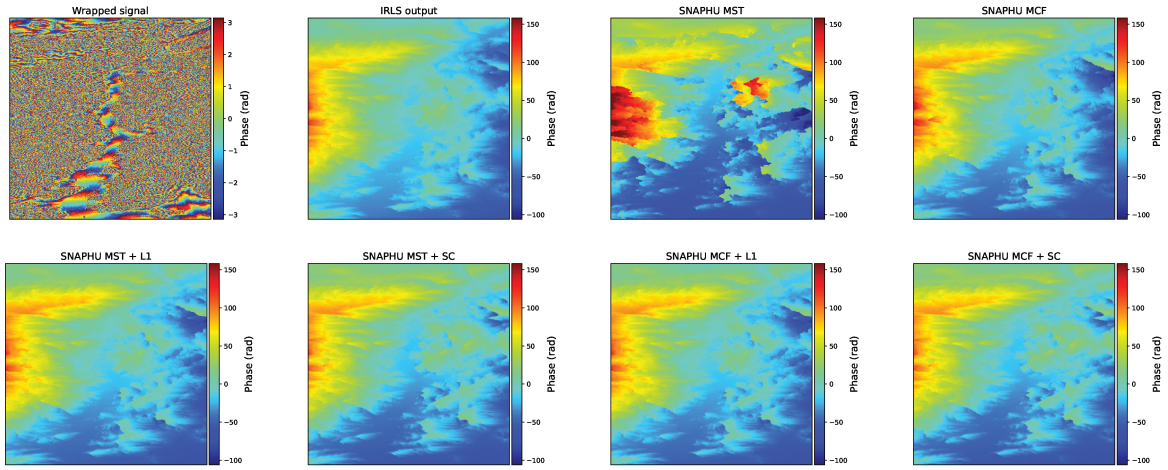

Figure 4.23: Comparison of the output of IRLS and of the different SNAPHU methods on Wulonggou (China) real image after Goldstein filtering. Image size is 2048×2048.

Conclusion and perspectives

In Chapter 2 we introduced a stochastic proximal Bregman algorithm for convex composite optimization problems. This algorithm converges to a neighborhood of the solution at the same accelerated rate as its deterministic counterpart. We used this result to design a variant of the Frank-Wolfe algorithm able to achieve accelerated rates under parallelization. One drawback is that the resulting algorithm is not "any time", in the sense that the desired precision ϵ must be given as an input to the algorithm. In the presented experiments we circumvented this by designing a simple heuristic which slowly decreases the value of ϵ. Yet a theoretical analysis for this restarted algorithm is lacking. Moreover, it would be worth exploring how the randomized smoothing techniques we used to develop our accelerated Frank-Wolfe method can be exploited to deal with stochasticity in the objective function.

Chapter 3 dealt with nonconvex separable optimization problems under affine constraints. We provided a way to construct a solution satisfying the classical duality gap bound obtained by the Shapley-Folkman theorem, through the use of conditional gradient methods and (approximate) Carathéodory algorithms. A promising area for future research could involve bounding the approximation error in the Shapley-Folkman theorem. A bound on this error was derived for a simpler version of the Shapley-Folkman result in [EG69], when the convex hull is taken on the sum of identical sets. This bound is based on the diameter of the underlying set and can therefore be quite loose. We believe that it could be improved and generalized to the sum of any sets, thereby obtaining a refined Shapley-Folkman theorem. This could also provide refined duality gap bounds for nonconvex optimization problems.

In Chapter 4 we studied the 2D phase unwrapping ℓ_1-norm minimization problem. We proposed an iteratively reweighted least squares algorithm which, combined with an efficient conjugate gradient method, allows to solve the problem using only simple linear algebra operations. This led to a GPU-compatible algorithm, for which we showed competitive performance compared to commonly-used phase unwrapping methods. A common way to enhance phase unwrapping algorithms for large images is through *tiling strategies*. The idea is to split the image into smaller tiles, to apply a classical unwrapping step on each of the tiles, and to assemble the unwrapped tiles so as to produce a full unwrapped solution [CZ02]. It remains to be explored how the iteratively reweighted least squares approach can be efficiently combined with tiling strategies.

Bibliography

[AAD+22] Roland Akiki, Jeremy Anger, Carlo De Franchis, Gabriele Facciolo, Jean Michel Morel, and Raphael Grandin. Improved Sentinel-1 IW Burst Stitching Through Geolocation Error Correction Considerations. In *International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 3404–3407, New York, NY, 2022. IEEE. 100

[AE76] Jean-Pierre Aubin and Ivar Ekeland. Estimates of the duality gap in nonconvex optimization. *Mathematics of Operations Research*, 1(3):225–245, 1976. 45, 46, 52, 53

[AFGO19] Necdet S. Aybat, Alireza Fallah, Mert Gurbuzbalaban, and Asuman Ozdaglar. A universally optimal multistage accelerated stochastic gradient method. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 21, 25

[AG18] GAMMA Remote Sensing AG. Interferometric sar processing: Gamma remote sensing, 2018. 84

[ALST14] Jacob Abernethy, Chansoo Lee, Abhinav Sinha, and Ambuj Tewari. Online linear optimization via smoothing. In *Conference on Learning Theory (COLT)*, 2014. 21, 23

[ALT16] Jacob Abernethy, Chansoo Lee, and Ambuj Tewari. Perturbation techniques in online learning and optimization. *Perturbations, Optimization, and Statistics*, page 223, 2016. 21, 23

[AMO88] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. *Network flows*. Cambridge, Mass.: Alfred P. Sloan School of Management, Massachusetts ..., 1988. 87

[ApS19] Mosek ApS. Mosek optimization toolbox for matlab. *User's Guide and Reference Manual, Version*, 4(1), 2019. 50

[AR21] Zeeshan Akhtar and Ketan Rajawat. Momentum based projection free stochastic optimization under affine constraints. In *2021 American Control Conference (ACC)*, pages 2619–2624. IEEE, 2021. 15

[AZHHL17] Zeyuan Allen-Zhu, Elad Hazan, Wei Hu, and Yuanzhi Li. Linear convergence of a frank-wolfe type algorithm over trace-norm balls. *Advances in neural information processing systems*, 30, 2017. 18

[B+13] Francis Bach et al. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends® in machine learning*, 6(2-3):145–373, 2013. 17

[Bac15] Francis Bach. Duality between subgradient and conditional gradient methods. *SIAM Journal on Optimization*, 25(1):115–129, 2015. 17, 21

[Bac21] Francis Bach. On the effectiveness of Richardson extrapolation in data science. *SIAM Journal on Mathematics of Data Science*, 3(4):1251–1277, 2021. 12, 22

[BBC+21] Mathieu Blondel, Quentin Berthet, Marco Cuturi, Roy Frostig, Stephan Hoyer, Felipe Llinares-López, Fabian Pedregosa, and Jean-Philippe Vert. Efficient and modular implicit differentiation. *arXiv preprint arXiv:2105.15183*, 2021. 21

[BBT+20] Quentin Berthet, Mathieu Blondel, Olivier Teboul, Marco Cuturi, Jean-Philippe Vert, and Francis Bach. Learning with differentiable pertubed optimizers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 21, 23

[BCC+22] Gábor Braun, Alejandro Carderera, Cyrille W Combettes, Hamed Hassani, Amin Karbasi, Aryan Mokhtari, and Sebastian Pokutta. Conditional gradient methods. *arXiv preprint arXiv:2211.14103*, 2022. 15

[BCL94] Keith Ball, Eric A Carlen, and Elliott H Lieb. Sharp uniform convexity and smoothness inequalities for trace norms. *Inventiones mathematicae*, 115(1):463–482, 1994. 40

[BCN18] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018. 28

[BD09] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and computational harmonic analysis*, 27(3):265–274, 2009. 89

[BDG23] Aleksandr Beznosikov, David Dobre, and Gauthier Gidel. Sarah frank-wolfe: Methods for constrained optimization with best rates and practical features. *arXiv preprint arXiv:2304.11737*, 2023. 16

[BDV07] Jos M Bioucas-Dias and Gonalo Valadao. Phase unwrapping via graph cuts. *IEEE Transactions on Image processing*, 16(3):698–709, 2007. 85

[Bec15] Amir Beck. On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes. *SIAM Journal on Optimization*, 25(1):185–209, 2015. 92, 93

[Bec17] Amir Beck. *First-order methods in optimization*. SIAM, 2017. 4, 6, 17, 136

[Ber14] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, Cambridge, MA, 2014. 45, 46, 53, 88

[BJM+12] Francis Bach, Rodolphe Jenatton, Julien Mairal, Guillaume Obozinski, et al. Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106, 2012. 85, 89

[BL08] Kristian Bredies and Dirk A Lorenz. Iterated hard shrinkage for minimization problems with sparsity constraints. *SIAM Journal on Scientific Computing*, 30(2):657–683, 2008. 17

[BLB+14] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 628–643. Springer, 2014. 18

[BLG+15] Piotr Bojanowski, Rémi Lajugie, Edouard Grave, Francis Bach, Ivan Laptev, Jean Ponce, and Cordelia Schmid. Weakly-supervised alignment of video with text. In *Proceedings of the IEEE international conference on computer vision*, pages 4462–4470, 2015. 18

[BLM09] Kristian Bredies, Dirk A Lorenz, and Peter Maass. A generalized conditional gradient method and its connection to an iterative shrinkage method. *Computational Optimization and applications*, 42:173–193, 2009. 17

[BPZ19] Gábor Braun, Sebastian Pokutta, and Daniel Zink. Lazifying conditional gradient algorithms. *Journal of Machine Learning Research*, 20(71):1–42, 2019. 17

[BR96] Michael J Black and Anand Rangarajan. On the unification of line processes, outlier rejection, and robust statistics with applications in early vision. *International journal of computer vision*, 19(1):57–91, 1996. 85, 89

[Bra21] Andreas Braun. Retrieval of digital elevation models from Sentinel-1 radar data – open applications, techniques, and limitations. *Open Geosciences*, 13(1):532–569, may 2021. 100

[BRZ21] Immanuel M Bomze, Francesco Rinaldi, and Damiano Zeffiro. Frank–wolfe and friends: a journey into projection-free first-order optimization methods. *4OR*, 19:313–345, 2021. 14

[BS72] Richard H. Bartels and George W Stewart. Solution of the matrix equation ax+ xb= c [f4]. *Communications of the ACM*, 15(9):820–826, 1972. 96

[BS17] Amir Beck and Shimrit Shtern. Linearly convergent away-step conditional gradient for non-strongly convex functions. *Mathematical Programming*, 164:1–27, 2017. 14, 15, 49, 51

[BSEK+23] Antoine Bambade, Fabian Schramm, Sarah El Kazdadi, Stéphane Caron, Adrien Taylor, and Justin Carpentier. Proxqp: an efficient and versatile quadratic programming solver for real-time robotics applications and beyond. In *RSS 2022-Robotics: Science and Systems*, 2023. 50, 68

[BT09] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009. 4, 6, 89

[BT16] Yingjie Bi and Ao Tang. Refined shapley-folkman lemma and its application in duality gap estimation. *arXiv preprint arXiv:1610.05416*, 2016. 45, 46

[BV04] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 2, 3, 72, 135, 136

[C+47] Augustin Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847. 3, 136

[CC68] Michael D Canon and Clifton D Cullum. A tight upper bound on the rate of convergence of frank-wolfe algorithm. *SIAM Journal on Control*, 6(4):509–516, 1968. 10

[CGT12] Coralia Cartis, Nicholas IM Gould, and Ph L Toint. An adaptive cubic regularization algorithm for nonconvex optimization with convex constraints and its function-evaluation complexity. *IMA Journal of Numerical Analysis*, 32(4):1662–1695, 2012. 16

[Che01] Curtis William Chen. *Statistical-cost network-flow approaches to two-dimensional phase unwrapping for radar interferometry*. stanford university, Stanford, CA, 2001. 101

[CJK14] Deeparnab Chakrabarty, Prateek Jain, and Pravesh Kothari. Provable submodular minimization using wolfe's algorithm. *Advances in Neural Information Processing Systems*, 27, 2014. 17

[CL15] Yiu-ming Cheung and Jian Lou. Efficient generalized conditional gradient with gradient sliding for composite optimization. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015. 17

[Cla10] Kenneth L Clarkson. Coresets, sparse greedy approximation, and the frank-wolfe algorithm. *ACM Transactions on Algorithms (TALG)*, 6(4):1–30, 2010. 7, 14, 18, 138

[Cos98] Mario Costantini. A novel phase unwrapping method based on network programming. *IEEE Transactions on geoscience and remote sensing*, 36(3):813–821, 1998. 85, 87, 99

[CP21] Cyrille W Combettes and Sebastian Pokutta. Complexity of linear minimization and projection on some sets. *Operations Research Letters*, 49(4):565–571, 2021. 7, 18, 137

[CP23] Cyrille W Combettes and Sebastian Pokutta. Revisiting the approximate carathéodory problem via the frank-wolfe algorithm. *Mathematical Programming*, 197(1):191–214, 2023. 10, 14, 18, 46, 50, 52, 138

[CSP20] Cyrille W Combettes, Christoph Spiegel, and Sebastian Pokutta. Projection-free adaptive gradients for large-scale optimization. *arXiv preprint arXiv:2009.14114*, 2020. 15

[CST21] Michael B Cohen, Aaron Sidford, and Kevin Tian. Relative lipschitzness in extragradient methods and a direct recipe for acceleration. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185, 2021. 26

[CZ00] Curtis W Chen and Howard A Zebker. Network approaches to two-dimensional phase unwrapping: intractability and two new algorithms. *JOSA A*, 17(3):401–414, 2000. 85, 99

[CZ01] Curtis W Chen and Howard A Zebker. Two-dimensional phase unwrapping with use of statistical models for cost functions in nonlinear optimization. *JOSA A*, 18(2):338–351, 2001. 85, 99, 101

[CZ02] Curtis W Chen and Howard A Zebker. Phase unwrapping for large sar interferograms: Statistical segmentation and generalized network models. *IEEE Transactions on Geoscience and Remote Sensing*, 40(8):1709–1719, 2002. 99, 118

[CZYG20] Jinghui Chen, Dongruo Zhou, Jinfeng Yi, and Quanquan Gu. A frank-wolfe framework for efficient and effective adversarial attacks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 3486–3494, 2020. 18

[DAST08] S Damla Ahipasaoglu, Peng Sun, and Michael J Todd. Linear convergence of a modified frank–wolfe algorithm for computing minimum-volume enclosing ellipsoids. *Optimisation Methods and Software*, 23(1):5–19, 2008. 15

[DBLJ14] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014. 15

[DBW12] John C. Duchi, Peter L. Bartlett, and Martin J. Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012. 21, 23

[DDDM04] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004. 85, 89

[DDFG10] Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C Sinan Güntürk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 63(1):1–38, 2010. 85, 89

[Dev13] Olivier Devolder. *Exactness, inexactness and stochasticity in first-order methods for large-scale convex optimization*. PhD thesis, PhD thesis, ICTEAM and CORE, Université Catholique de Louvain, 2013. 3, 6, 21, 136, 137

[DG21] Jelena Diakonikolas and Cristóbal Guzmán. Complementary composite minimization, small gradients in general norms, and applications to regression problems. *arXiv preprint 2101.11041*, 2021. 20, 21, 24

[DH78] Joseph C Dunn and S Harshbarger. Conditional gradient algorithms with open loop step size rules. *Journal of Mathematical Analysis and Applications*, 62(2):432–444, 1978. 15

[DL02] José MB Dias and José MN Leitão. The z/spl pi/m algorithm: a method for interferometric image reconstruction in sar/sas. *IEEE Transactions on Image processing*, 11(4):408–422, 2002. 85

[DR70] Vladimir Fedorovich Demyanov and Aleksandr Moiseevich Rubinov. *Approximate methods in optimization problems*. Elsevier Publishing Company, 1970. 11, 22

[dST21] Alexandre d'Aspremont, Damien Scieur, and Adrien Taylor. Acceleration methods. *Foundations and Trends® in Optimization*, 5(1-2):1–245, 2021. 21

[DTAd24] Benjamin Dubois-Taine, Roland Akiki, and Alexandre d'Aspremont. Iteratively reweighted least squares for phase unwrapping. *arXiv preprint arXiv:2401.09961*, 2024. 19, 139

[DTBBT22] Benjamin Dubois-Taine, Francis Bach, Quentin Berthet, and Adrien Taylor. Fast stochastic composite minimization and an accelerated frank-wolfe algorithm under parallelization. *Advances in Neural Information Processing Systems*, 35:33523–33535, 2022. 19, 139

[DTd24] Benjamin Dubois-Taine and Alexandre d'Aspremont. Frank-wolfe meets shapley-folkman: a systematic approach for solving nonconvex separable problems with linear constraints. *arXiv preprint arXiv:2406.18282*, 2024. 19, 139

[Dun79] Joseph C Dunn. Rates of convergence for conditional gradient algorithms near singular and nonsingular extremals. *SIAM Journal on Control and Optimization*, 17(2):187–211, 1979. 11, 15

[EG69] William R Emerson and Frederick P Greenleaf. Asymptotic behavior of products $C^p = C + \cdots + C$ in locally compact abelian groups. *Transactions of the American Mathematical Society*, 145:171–204, 1969. 118

[ET99] Ivar Ekeland and Roger Temam. *Convex analysis and variational problems*. SIAM, 1999. 52

[FALA17] Salar Fattahi, Morteza Ashraphijuo, Javad Lavaei, and Alper Atamtürk. Conic relaxations of the unit commitment problem. *Energy*, 134:1079–1095, 2017. 67

[Faz02] Maryam Fazel. *Matrix rank minimization with applications*. PhD thesis, PhD thesis, Stanford University, 2002. 18

[FCTR16] Rémi Flamary, Nicholas Courty, Davis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE Trans. Pattern Anal. Mach. Intell*, 1(1-40):2, 2016. 18

[FGM17] Robert M. Freund, Paul Grigas, and Rahul Mazumder. An extended Frank–Wolfe method with "in-face" directions, and its application to low-rank matrix completion. *SIAM Journal on optimization*, 27(1):319–346, 2017. 10, 18, 22, 138

[FKP+14] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6:327–363, 2014. 50

[Fly96] Thomas J Flynn. Consistent 2-d phase unwrapping guided by a quality map. In *IGARSS'96. 1996 International Geoscience and Remote Sensing Symposium*, volume 4, pages 2057–2059. IEEE, 1996. 84

[Fly97] Thomas J Flynn. Two-dimensional phase unwrapping with minimum weighted discontinuity. *JOSA A*, 14(10):2692–2701, 1997. 85

[FMGP+07] Alessandro Ferretti, Andrea Monti-Guarnieri, Claudio Prati, Fabio Rocca, and Didier Massonet. *InSAR principles-guidelines for SAR interferometry processing and interpretation*, volume 19. European Space Agency, 2007. 85

[FPRW16] Massimo Fornasier, Steffen Peter, Holger Rauhut, and Stephan Worm. Conjugate gradient acceleration of iteratively re-weighted least squares methods. *Computational optimization and applications*, 65:205–259, 2016. 85, 88, 89, 99

[FRC+07] Tom G Farr, Paul A Rosen, Edward Caro, Robert Crippen, Riley Duren, Scott Hensley, Michael Kobrick, Mimi Paller, Ernesto Rodriguez, Ladislav Roth, et al. The shuttle radar topography mission. *Reviews of geophysics*, 45(2), 2007. 100

[Fuj80] Satoru Fujishige. Lexicographically optimal base of a polymatroid with respect to a weight vector. *Mathematics of Operations Research*, 5(2):186–196, 1980. 17

[FW56] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956. 6, 21, 136

[GB14] Pontus Giselsson and Stephen Boyd. Monotonicity and restart in fast gradient methods. In *53rd IEEE Conference on Decision and Control*, pages 5058–5063. IEEE, 2014. 16

[GER31] S GERSCHGORIN. Uber die abgrenzung der eigenwerte einer matrix. *lzv. Akad. Nauk. USSR. Otd. Fiz-Mat. Nauk*, 7:749–754, 1931. 92

[GH15] Dan Garber and Elad Hazan. Faster rates for the frank-wolfe method over strongly-convex sets. In *International Conference on Machine Learning*, pages 541–549. PMLR, 2015. 11, 22, 28, 51, 52

[GH16] Dan Garber and Elad Hazan. A linearly convergent variant of the conditional gradient algorithm under strong convexity, with applications to online and stochastic optimization. *SIAM Journal on Optimization*, 26(3):1493–1528, 2016. 16

[Gha19] Saeed Ghadimi. Conditional gradient type methods for composite nonlinear and stochastic optimization. *Mathematical Programming*, 173(1):431–464, 2019. 28

[GIZ17] Donald Goldfarb, Garud Iyengar, and Chaoxu Zhou. Linear convergence of stochastic frank wolfe variants. In *Artificial Intelligence and Statistics*, pages 1066–1074. PMLR, 2017. 15

[GL16] Saeed Ghadimi and Guanghui Lan. Accelerated gradient methods for nonconvex nonlinear and stochastic programming. *Mathematical Programming*, 156(1):59–99, 2016. 16

[GM86] Jacques Guélat and Patrice Marcotte. Some comments on Wolfe's 'away step'. *Mathematical Programming*, 35(1):110–119, 1986. 11, 12, 22, 50, 51

[GN18] Alexander V. Gasnikov and Yurii Nesterov. Universal method for stochastic composite optimization problems. *Computational Mathematics and Mathematical Physics*, 58(1):48–64, 2018. 21

[GP98] Dennis C Ghiglia and Mark D Pritt. Two-dimensional phase unwrapping: theory, algorithms, and software. *Wiely-Interscience, first ed.(April 1998)*, 1998. 83, 84, 85

[GR94] Dennis C Ghiglia and Louis A Romero. Robust two-dimensional weighted and unweighted phase unwrapping that uses fast transforms and iterative methods. *JOSA A*, 11(1):107–117, 1994. 84, 96

[GR96] Dennis C Ghiglia and Louis A Romero. Minimum lp-norm two-dimensional phase unwrapping. *JOSA A*, 13(10):1999–2013, 1996. 85, 88

[Gra74] Leroy C Graham. Synthetic interferometer radar for topographic mapping. *Proceedings of the IEEE*, 62(6):763–768, 1974. 83

[GSK18] Dan Garber, Shoham Sabach, and Atara Kaplan. Fast generalized conditional gradient method with applications to matrix recovery problems. *arXiv preprint arXiv:1802.05581*, 3, 2018. 18

[Gum54] Emil J. Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1954. 28

[GVL13] Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013. 18, 50

[GW98] Richard M Goldstein and Charles L Werner. Radar interferogram filtering for geophysical applications. *Geophysical Research Letters*, 25(21):4035–4038, 1998. 100

[GZW88] Richard M Goldstein, Howard A Zebker, and Charles L Werner. Satellite radar interferometry: Two-dimensional phase unwrapping. *Radio science*, 23(4):713–720, 1988. 84

[Han02] Ramon Hanssen. *Radar Interferometry, Data Interpretation and Error Analysis*. Kluwer Academic Publishers, Dordrecht, Netherlands, 2002. 100

[Hay18] Ken Hayami. Convergence of the conjugate gradient method on singular systems. *arXiv preprint arXiv:1809.00793*, 2018. 94, 97, 98

[HBM20] Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. Dual-free stochastic decentralized optimization with variance reduction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 26

[HBSA12] Andrew Hooper, David Bekaert, Karsten Spaans, and Mahmut Arıkan. Recent advances in sar interferometry time series analysis for measuring crustal deformation. *Tectonophysics*, 514:1–13, 2012. 85

[HDP+12] Zaid Harchaoui, Matthijs Douze, Mattis Paulin, Miroslav Dudik, and Jérôme Malick. Large-scale image classification with trace-norm regularization. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3386–3393. IEEE, 2012. 18

[HJN15] Zaid Harchaoui, Anatoli Juditsky, and Arkadi Nemirovski. Conditional gradient algorithms for norm-regularized smooth convex optimization. *Mathematical Programming*, 152(1):75–112, 2015. 7, 22, 137

[HK12] Elad Hazan and Satyen Kale. Projection-free online learning. *arXiv preprint arXiv:1206.4657*, 2012. 7, 137

[HL16] Elad Hazan and Haipeng Luo. Variance-reduced and projection-free stochastic optimization. In *International Conference on Machine Learning*, pages 1263–1271. PMLR, 2016. 16, 28

[Hol74] Charles A Holloway. An extension of the frank and wolfe method of feasible directions. *Mathematical Programming*, 6:14–27, 1974. 14

[HR92] Mark Hedley and Dov Rosenfeld. A new two-dimensional phase unwrapping algorithm for mri images. *Magnetic Resonance in Medicine*, 24(1):177–181, 1992. 83

[HUL96] Jean-Baptiste Hiriart-Urruty and Claude Lemaréchal. *Convex analysis and minimization algorithms*, volume 305. Springer science & business media, 1996. 46, 50, 79, 81

[HW77] Paul W Holland and Roy E Welsch. Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827, 1977. 89

[HZW+22] Jie Hou, Xianlin Zeng, Gang Wang, Jian Sun, and Jie Chen. Distributed momentum-based frank-wolfe algorithm for stochastic optimization. *IEEE/CAA Journal of Automatica Sinica*, 10(3):685–699, 2022. 15

[Ito82] Kazuyoshi Itoh. Analysis of the phase unwrapping algorithm. *Applied optics*, 21(14):2470–2470, 1982. 84

[Jag11] Martin Jaggi. *Sparse convex optimization methods for machine learning*. PhD thesis, ETH Zürich, 2011. 21, 56

[Jag13] Martin Jaggi. Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning (ICML)*, 2013. 7, 10, 12, 15, 21, 138

[Jia16] Gao Jian. Reliability-map-guided phase unwrapping method. *IEEE Geoscience and Remote Sensing Letters*, 13(5):716–720, 2016. 84

[JST22] Yujia Jin, Aaron Sidford, and Kevin Tian. Sharper rates for separable minimax and finite sum optimization via primal-dual extragradient methods. *arXiv preprint arXiv:2202.04640*, 2022. 26

[JTFF14] Armand Joulin, Kevin Tang, and Li Fei-Fei. Efficient image and video co-localization with frank-wolfe algorithm. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pages 253–268. Springer, 2014. 18

[JZ13] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013. 15

[Kaa88] Erik F Kaasschieter. Preconditioned conjugate gradients for solving singular systems. *Journal of Computational and Applied mathematics*, 24(1-2):265–275, 1988. 94, 97

[KCd17] Thomas Kerdreux, Igor Colin, and Alexandre d'Aspremont. An approximate shapley-folkman theorem. *arXiv preprint arXiv:1712.08559*, 2017. 47, 53

[KdP21] Thomas Kerdreux, Alexandre d'Aspremont, and Sebastian Pokutta. Projection-free optimization on uniformly convex sets. In *International Conference on Artificial Intelligence and Statistics*, pages 19–27. PMLR, 2021. 11

[KdP22] Thomas Kerdreux, Alexandre d'Aspremont, and Sebastian Pokutta. Restarting frank–wolfe: faster rates under hölderian error bounds. *Journal of Optimization Theory and Applications*, 192(3):799–829, 2022. 16

[Kel95] Carl T Kelley. *Iterative methods for linear and nonlinear equations*. SIAM, Philadelphia, PA, 1995. 96, 98

[Ker20] Thomas Kerdreux. *Accelerating conditional gradient methods*. PhD thesis, Université Paris sciences et lettres, 2020. 15

[KSST+09] Sham Kakade, Shai Shalev-Shwartz, Ambuj Tewari, et al. On the duality of strong convexity and strong smoothness: Learning applications and matrix regularization. *Unpublished Manuscript, http://ttic. uchicago. edu/shai/papers/KakadeShalevTewari09. pdf*, 2(1), 2009. 40

[KST08] Sham M Kakade, Karthik Sridharan, and Ambuj Tewari. On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. *Advances in neural information processing systems*, 21, 2008. 40

[KW92] Jacek Kuczyński and Henryk Woźniakowski. Estimating the largest eigenvalue by the power and lanczos algorithms with a random start. *SIAM journal on matrix analysis and applications*, 13(4):1094–1122, 1992. 18

[KY05] Piyush Kumar and E Alper Yildirim. Minimum-volume enclosing ellipsoids and core sets. *Journal of Optimization Theory and applications*, 126(1):1–21, 2005. 18

[KY11] Piyush Kumar and E Alper Yıldırım. A linearly convergent linear-time first-order algorithm for support vector classification with a core set result. *INFORMS Journal on Computing*, 23(3):377–391, 2011. 15, 18

[Lan12] Guanghui Lan. An optimal method for stochastic composite optimization. *Mathematical Programming*, 133(1):365–397, 2012. 21, 26

[Lan13] Guanghui Lan. The complexity of large-scale convex programming under a linear optimization oracle. *arXiv preprint 1309.5550*, 2013. 12, 16, 22, 138

[Lau73] Paul C Lauterbur. Image formation by induced local interactions: examples employing nuclear magnetic resonance. *nature*, 242(5394):190–191, 1973. 83

[Law61] Charles Lawrence Lawson. Contribution to the theory of linear least maximum approximation. *Ph. D. dissertation. Univ. Calif.*, 1961. 89

[LCGL20] Bingcong Li, Mario Coutino, Georgios B. Giannakis, and Geert Leus. How does momentum help Frank Wolfe? *arXiv preprint 2006.11116*, 2020. 12, 22

[LCRCPIP21] Alejandro Linde-Cerezo, Marc Rodriguez-Cassola, Pau Prats-Iraola, and Muriel Pinheiro. Systematic Comparison of Backgeocoding Algorithms for SAR Processing and Simulation Environments. In *EUSAR 2021; 13th European Conference on Synthetic Aperture Radar*, pages 383–386, 2021. 100

[Lem12] Claude Lemaréchal. Cauchy and the gradient method. *Doc Math Extra*, 251(254):10, 2012. 3, 136

[LJ16] Simon Lacoste-Julien. Convergence rate of frank-wolfe for non-convex objectives. *arXiv preprint arXiv:1607.00345*, 2016. 16

[LJJ13] Simon Lacoste-Julien and Martin Jaggi. An affine invariant linear convergence analysis for frank-wolfe algorithms. *arXiv preprint arXiv:1312.7864*, 2013. 7, 11, 138

[LJJ15] Simon Lacoste-Julien and Martin Jaggi. On the global linear convergence of frank-wolfe optimization variants. *Advances in neural information processing systems*, 28, 2015. 12, 13, 14, 15, 22, 51, 52, 66

[LJJSP13] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. In *International Conference on Machine Learning*, pages 53–61. PMLR, 2013. 15, 17

[LP66] Evgeny S. Levitin and Boris T. Polyak. Constrained minimization methods. *USSR Computational mathematics and mathematical physics*, 6(5):1–50, 1966. 11, 22

[LPZZ17] Guanghui Lan, Sebastian Pokutta, Yi Zhou, and Daniel Zink. Conditional accelerated lazy stochastic gradient descent. In *International Conference on Machine Learning*, pages 1965–1974. PMLR, 2017. 17

[LR01] Claude Lemaréchal and Arnaud Renaud. A geometric study of duality gaps, with applications. *Mathematical Programming*, 90:399–427, 2001. 47, 80, 81

[LSPC19] Giulia Luise, Saverio Salzo, Massimiliano Pontil, and Carlo Ciliberto. Sinkhorn barycenters with free support via frank-wolfe algorithm. *Advances in neural information processing systems*, 32, 2019. 18

[LZ16] Guanghui Lan and Yi Zhou. Conditional gradient sliding for convex optimization. *SIAM Journal on Optimization*, 26(2):1379–1409, 2016. 15, 17, 22, 28

[MAB+17] Antoine Miech, Jean-Baptiste Alayrac, Piotr Bojanowski, Ivan Laptev, and Josef Sivic. Learning from video and text via large-scale discriminative clustering. In *Proceedings of the IEEE international conference on computer vision*, pages 5257–5266, 2017. 18

[MBP+14] Julien Mairal, Francis Bach, Jean Ponce, et al. Sparse modeling for image and vision processing. *Foundations and Trends® in Computer Graphics and Vision*, 8(2-3):85–283, 2014. 85, 89

[MDM74] BF Mitchell, Vladimir Fedorovich Dem'yanov, and VN Malozemov. Finding the point of a polyhedron closest to the origin. *SIAM Journal on Control*, 12(1):19–26, 1974. 13

[MHK20] Aryan Mokhtari, Hamed Hassani, and Amin Karbasi. Stochastic conditional gradient methods: From convex minimization to submodular maximization. *Journal of machine learning research*, 21(105):1–49, 2020. 17

[MLP+22] Louis Montaut, Quentin Le Lidec, Vladimir Petrik, Josef Sivic, and Justin Carpentier. Collision detection accelerated: An optimization perspective. *arXiv preprint arXiv:2205.09663*, 2022. 18

[MS14] Rammohan Mallipeddi and Ponnuthurai Nagaratnam Suganthan. Unit commitment–a survey and comparison of conventional and nature inspired algorithms. *International Journal of Bio-Inspired Computation*, 6(2):71–90, 2014. 67

[Naz23] Hamid Nazari. New variants of frank-wolfe algorithm for video co-localization problem. *arXiv preprint arXiv:2307.04319*, 2023. 18

[NDT+20] Geoffrey Négiar, Gideon Dresdner, Alicia Tsai, Laurent El Ghaoui, Francesco Locatello, Robert Freund, and Fabian Pedregosa. Stochastic frank-wolfe for constrained finite-sum minimization. In *International Conference on Machine Learning*, pages 7253–7262. PMLR, 2020. 16, 28

[Nes83] Yurii Nesterov. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. In *Doklady an USSR*, volume 269, pages 543–547, 1983. 5, 21, 22, 136

[Nes03] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Science & Business Media, 2003. 3, 4, 5, 16, 21, 22, 135, 136

[Nes13] Yurii Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013. 21

[ÑFSA14] Ricardo Ñanculef, Emanuele Frandi, Claudio Sartori, and Héctor Allende. A novel frank–wolfe algorithm. analysis and applications to large-scale svm training. *Information Sciences*, 285:66–99, 2014. 15, 17

[NLST17] Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International conference on machine learning*, pages 2613–2621. PMLR, 2017. 16

[NO09] Angelia Nedić and Asuman Ozdaglar. Approximate primal solutions and rate analysis for dual subgradient methods. *SIAM Journal on Optimization*, 19(4):1757–1780, 2009. 72

[NPD00] Giovanni Nico, Gintautas Palubinskas, and Mihai Datcu. Bayesian approaches to phase unwrapping: theoretical study. *IEEE Transactions on Signal processing*, 48(9):2545–2556, 2000. 85

[NS17] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17(2):527–566, 2017. 21, 23

[NW06] Jorge Nocedal and Stephen J Wright. Conjugate gradient methods. *Numerical optimization*, pages 101–134, 2006. 85, 93, 96

[OAL+16] Anton Osokin, Jean-Baptiste Alayrac, Isabella Lukasewitz, Puneet Dokania, and Simon Lacoste-Julien. Minding the gaps for block frank-wolfe optimization of structured svms. In *international conference on machine learning*, pages 593–602. PMLR, 2016. 17

[OC15] Brendan O'donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15:715–732, 2015. 16

[Osb85] Michael Robert Osborne. *Finite algorithms in optimization and data analysis*. John Wiley & Sons, Inc., Hoboken, NJ, 1985. 89

[Pad04] Narayana Prasad Padhy. Unit commitment-a bibliographical survey. *IEEE Transactions on power systems*, 19(2):1196–1205, 2004. 67

[PANJ18] Fabian Pedregosa, Armin Askari, Geoffrey Negiar, and Martin Jaggi. Step-size adaptivity in projection-free optimization. *arXiv preprint arXiv:1806.05123*, 2018. 9

[PC19] François-Pierre Paty and Marco Cuturi. Subspace robust wasserstein distances. In *International conference on machine learning*, pages 5072–5081. PMLR, 2019. 18

[PJJ94] SM Pandit, N Jordache, and GA Joshi. Data-dependent systems methodology for noise-insensitive phase unwrapping in laser interferometric surface characterization. *JOSA A*, 11(10):2584–2592, 1994. 83

[PNAJ20] Fabian Pedregosa, Geoffrey Negiar, Armin Askari, and Martin Jaggi. Linearly convergent frank-wolfe with backtracking line-search. In *International conference on artificial intelligence and statistics*, pages 1–10. PMLR, 2020. 10, 13, 15, 16, 138

[PR19] Javier Pena and Daniel Rodriguez. Polytope conditioning and linear convergence of the frank–wolfe algorithm. *Mathematics of Operations Research*, 44(1):1–18, 2019. 14, 51, 52, 66

[PRS16] Javier Pena, Daniel Rodríguez, and Negar Soheili. On the von neumann and frank–wolfe algorithms with away steps. *SIAM Journal on Optimization*, 26(1):499–512, 2016. 14, 51

[PSLS17] Julia Peyre, Josef Sivic, Ivan Laptev, and Cordelia Schmid. Weakly-supervised learning of visual relations. In *Proceedings of the ieee international conference on computer vision*, pages 5179–5188, 2017. 18

[PSZ20] Sebastian Pokutta, Christoph Spiegel, and Max Zimmer. Deep neural network training with frank-wolfe. *arXiv preprint arXiv:2010.07243*, 2020. 18

[QLX18] Chao Qu, Yan Li, and Huan Xu. Non-convex conditional gradient sliding. In *international conference on machine learning*, pages 4208–4217. PMLR, 2018. 17

[RCS19] Sathya N Ravi, Maxwell D Collins, and Vikas Singh. A deterministic nonsmooth frank wolfe algorithm with coreset guarantees. *Informs Journal on Optimization*, 1(2):120–142, 2019. 18

[Rd17] Vincent Roulet and Alexandre d'Aspremont. Sharpness, restart and acceleration. *Advances in Neural Information Processing Systems*, 30, 2017. 16

[RHJ+00] Paul A Rosen, Scott Hensley, Ian R Joughin, Fuk K Li, Soren N Madsen, Ernesto Rodriguez, and Richard M Goldstein. Synthetic aperture radar interferometry. *Proceedings of the IEEE*, 88(3):333–382, 2000. 83

[Rob82] Stephen M Robinson. *Generalized equations and their solutions, Part II: Applications to nonlinear programming*. Springer, 1982. 15

[Roc15] Ralph T. Rockafellar. *Convex Analysis*. Princeton University Press, 2015. 10, 23, 26, 47

[RSPS16] Sashank J Reddi, Suvrit Sra, Barnabás Póczos, and Alex Smola. Stochastic frank-wolfe methods for nonconvex optimization. In *2016 54th annual Allerton conference on communication, control, and computing (Allerton)*, pages 1244–1251. IEEE, 2016. 15

[RW09] Ralph T. Rockafellar and Roger J.-B. Wets. *Variational Analysis*. Springer Science & Business Media, 2009. 26

[SBG+20] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020. 50

[SBLL16] Guillaume Seguin, Piotr Bojanowski, Rémi Lajugie, and Ivan Laptev. Instance-level video segmentation from object tracks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3678–3687, 2016. 18

[SCHU17] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017. 18

[SLRB17] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162:83–112, 2017. 15

[SNA] SNAP - ESA Sentinel Application Platform v2.0.2. `http://step.esa.int`. 85

[SRD18] Vishnu Sai Rao Lokhande Vikas Singh, Sathya N Ravi, and Tuan Dinh. Constrained deep learning using conditional gradient and applications in computer vision. *arXiv preprint arXiv:1803.0645*, 2018. 18

[SSGS11] Shai Shalev-Shwartz, Alon Gonen, and Ohad Shamir. Large-scale convex minimization with a low-rank constraint. *arXiv preprint arXiv:1106.1622*, 2011. 10, 18, 138

[Sta69] Ross M Starr. Quasi-equilibria in markets with non-convex preferences. *Econometrica: journal of the Econometric Society*, pages 25–38, 1969. 45, 52

[TR22] Theodoros Tsiligkaridis and Jay Roberts. Understanding and increasing efficiency of frank-wolfe adversarial training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 50–59, 2022. 18

[UB16] Madeleine Udell and Stephen Boyd. Bounding duality gap for separable problems with linear constraints. *Computational Optimization and Applications*, 64(2):355–378, 2016. 45, 53

[VCTF19] Titouan Vayer, Nicolas Courty, Romain Tavenard, and Rémi Flamary. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, pages 6275–6284. PMLR, 2019. 18

[VDTB22] Sharan Vaswani, Benjamin Dubois-Taine, and Reza Babanezhad. Towards noise-adaptive, problem-adaptive (accelerated) stochastic gradient descent. In *International Conference on Machine Learning*, pages 22015–22059. PMLR, 2022. 28

[VEG+16] Robin Vujanic, Peyman Mohajerin Esfahani, Paul J Goulart, Sébastien Mariéthoz, and Manfred Morari. A decomposition method for large scale milps, with performance guarantees and a power system application. *Automatica*, 67:144–156, 2016. 45, 46, 54, 63, 72

[VEGM14] Robin Vujanic, Peyman Mohajerin Esfahani, Paul Goulart, and Manfred Morari. Large scale mixed-integer optimization: A solution method with supply chain applications. In *22nd Mediterranean Conference on Control and Automation*, pages 804–809. IEEE, 2014. 45, 46, 53, 54, 63

[VML+19] Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. *Advances in neural information processing systems*, 32, 2019. 28

[Wir24] Elias Wirth. *Frank-Wolfe Algorithms in Polytope Settings*. PhD thesis, Technische Universität Berlin, 2024. 45, 49

[Wol70] Philip Wolfe. Convergence theory in nonlinear programming. *Integer and nonlinear programming*, pages 1–36, 1970. 12, 50

[WX17] Jialei Wang and Lin Xiao. Exploiting strong convexity from data with primal-dual first-order algorithms. In *International Conference on Machine Learning (ICML)*, 2017. 26

[XSZ+20] Jiahao Xie, Zebang Shen, Chao Zhang, Boyu Wang, and Hui Qian. Efficient projection-free online methods with stochastic recursive gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6446–6453, 2020. 18

[YLB11] Hanwen Yu, Zhenfang Li, and Zheng Bao. Residues cluster-based segmentation and outlier-detection method for large-scale phase unwrapping. *IEEE Transactions on Image Processing*, 20(10):2865–2875, 2011. 85

[YLY+19] Hanwen Yu, Yang Lan, Zhihui Yuan, Junyi Xu, and Hyongki Lee. Phase unwrapping in insar: A review. *IEEE Geoscience and Remote Sensing Magazine*, 7(1):40–58, 2019. 84

[ZG86] Howard A Zebker and Richard M Goldstein. Topographic mapping from interferometric synthetic aperture radar observations. *Journal of Geophysical Research: Solid Earth*, 91(B5):4993–4999, 1986. 83

[ZHZ+11] Ming Zhao, Lei Huang, Qican Zhang, Xianyu Su, Anand Asundi, and Qian Kemao. Quality-guided phase unwrapping technique: comparison of quality maps and guiding strategies. *Applied optics*, 50(33):6214–6224, 2011. 84

[ZTZC10] Heping Zhong, Jinsong Tang, Sen Zhang, and Ming Chen. An improved quality-guided phase-unwrapping algorithm based on priority queue. *IEEE Geoscience and Remote Sensing Letters*, 8(2):364–368, 2010. 84

Résumé long en français

1.1 Introduction à l'optimisation

Le besoin de solutions « optimales » est, d'une manière ou d'une autre, au cœur de la plupart des problèmes modernes en ingénierie, en économie et en sciences computationnelles. Mathématiquement, un problème d'optimisation peut s'écrire comme suit:

$$\text{minimiser} \quad f(x) \quad \text{tel que } x \in \mathcal{C}, \tag{1.41}$$

où f est communément appelée la *fonction objectif* et $\mathcal{C}$ l'*ensemble des solutions admissibles*, ou *ensemble des contraintes*.

La plupart des algorithmes visant à résoudre des problèmes de la forme (1.41) relève de la catégorie des méthodes *itératives*; ils commencent par une estimation initiale de la solution x_0 et affinent cette estimation de manière itérative à mesure qu'ils obtiennent plus d'informations sur le problème, générant ainsi une suite d'estimations $x_0, x_1, x_2, \ldots$.

Ce qui différencie les méthodes itératives réside précisément dans le type d'information auquel elles ont accès et dans la manière dont elles affinent leur estimation à partir de cette information. Dans cette thèse, nous nous intéressons principalement aux méthodes dites de *premier ordre*. Ces méthodes mettent à jour leur estimation en se basant sur le gradient ∇f de la fonction objectif, par opposition, par exemple, aux méthodes de *second ordre*, qui s'appuient sur des informations issues de la hessienne $\nabla^2 f$. À partir de maintenant, nous supposons que la fonction objectif f est différentiable. Nous supposons également que le problème (1.41) a une solution, c'est-à-dire qu'il existe un $x^* \in \mathcal{C}$ tel que $x^* \in \arg\min_{x \in \mathcal{C}} f(x)$.

Dans le reste de cette section, nous introduisons des notions classiques comme la convexité et la régularité de la fonction objectif, qui sont quelques-uns des principaux outils mathématiques utilisés pour étudier les méthodes de premier ordre.

1.1.1 Notations générales

Nous considérons un espace vectoriel de dimension finie $\mathbb{R}^d$ muni du produit scalaire euclidien standard $\langle \cdot, \cdot \rangle : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, et une norme $\|\cdot\| : \mathbb{R}^d \to \mathbb{R}$. Nous n'assumons pas que la norme est induite par le produit scalaire. Nous considérons également la norme duale correspondante $\|\cdot\|_* : \mathbb{R}^d \to \mathbb{R}$ induite par le choix de $\|\cdot\|$ à l'aide de $\|z\|_* = \sup_{\|x\| \leq 1} \langle z, x \rangle$. Nous désignons par $\|\cdot\|_2$ la norme euclidienne, définie par $\|x\|_2 = \sqrt{\langle x, x \rangle}$. Étant donné un ensemble $\mathcal{C} \subset \mathbb{R}^d$, nous notons son diamètre par rapport à la norme $\|\cdot\|$ comme $\operatorname{diam}(\mathcal{C}, \|\cdot\|) = \sup_{x,y \in \mathcal{C}} \|x - y\|$.

Notez que tous les résultats présentés dans cette section s'appliquent également à l'espace des matrices $\mathbb{R}^{p \times q}$ muni du produit scalaire euclidien sur les matrices.

1.1.2 Hypothèses de régularité

Nous introduisons d'abord la notion de convexité d'un ensemble.

Definition 1.1.1. *Un ensemble $\mathcal{C} \subset \mathbb{R}^d$ est convexe si*

$$\forall x, y \in \mathcal{C}, \forall \lambda \in [0, 1], \ \lambda x + (1 - \lambda) y \in \mathcal{C}.$$

Definition 1.1.2. *Une fonction $f : \mathbb{R}^d \to \mathbb{R}$ est convexe si*

$$\forall x, y \in \mathbb{R}^d, \forall \lambda \in [0,1],\ f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y).$$

On dit que f est strictement convexe si l'inégalité est stricte dès que $x \neq y$ et $\lambda \in (0,1)$.

La définition ci-dessus est une définition de la convexité d'ordre zéro. Si la fonction f est différentiable, on peut également obtenir une caractérisation de la convexité d'ordre un [BV04, Section 3.1.3].

Proposition 1.1.1. *Une fonction différentiable $f : \mathbb{R}^d \to \mathbb{R}$ est convexe si et seulement si*

$$\forall x, y \in \mathbb{R}^d,\ f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle.$$

D'après le résultat ci-dessus, une fonction est convexe si, en tout point, elle se situe au-dessus de son approximation du premier ordre. Une conséquence importante de la convexité est que tous les minima locaux de f sont des minima globaux. Cela constitue un avantage majeur pour les méthodes itératives du premier ordre, car cela garantit que ces méthodes ne peuvent pas se retrouver « bloquées » dans des minima locaux. La notion de convexité peut également être renforcée pour incorporer un terme quadratique dans la borne inférieure.

Definition 1.1.3. *Une fonction différentiable $f : \mathbb{R}^d \to \mathbb{R}$ est dite μ-fortement convexe par rapport à $\|\cdot\|$ si*

$$\forall x, y \in \mathbb{R}^d,\ f(x) \geq f(y) + \langle \nabla f(y), x - y \rangle + \frac{\mu}{2} \|x - y\|^2 .$$

Pour étudier les méthodes du premier ordre, une hypothèse courante est celle que la fonction objectif f est *lisse*. Mathématiquement, la fonction f est lisse si son gradient est lipschitzien (voir par exemple [Nes03]). De manière équivalente ici, nous définissons lisse comme l'existence d'une borne quadratique supérieure sur la fonction f. Cette version est plus utile lorsqu'il s'agit de prouver la convergence des méthodes du premier ordre.

Definition 1.1.4. *Une fonction différentiable $f : \mathbb{R}^d \to \mathbb{R}$ est L-lisse par rapport à $\|\cdot\|$ si*

$$\forall x, y \in \mathbb{R}^d,\ f(x) \leq f(y) + \langle \nabla f(y), x - y \rangle + \frac{L}{2} \|x - y\|^2 . \tag{1.42}$$

1.2 Descente de gradient projeté

Dans le cas non contraint, c'est-à-dire lorsque $\mathcal{C} = \mathbb{R}^d$, une manière naturelle d'affiner l'estimation actuelle x_k est de faire un pas dans la direction opposée au gradient:

$$x_{k+1} = x_k - \eta_k \nabla f(x_k), \quad k = 0, 1, \ldots, \tag{1.43}$$

où $\eta_k > 0$ contrôle la taille du pas dans la direction de descente. Lorsque l'ensemble admissible $\mathcal{C}$ n'est pas l'espace entier, la mise à jour (1.43) peut produire un point en dehors de l'ensemble admissible. Une manière de gérer cela est de *projeter* le point sur $\mathcal{C}$.

Definition 1.2.1. *Étant donné $\tilde{x} \in \mathbb{R}^d$ et un ensemble fermé $\mathcal{C} \subset \mathbb{R}^d$, la projection de $\tilde{x}$ sur $\mathcal{C}$ par rapport à la norme $\|\cdot\|$, notée $\Pi_{\mathcal{C}}(\tilde{x})$, est définie comme*

$$\Pi_{\mathcal{C}}(\tilde{x}) = \operatorname*{arg\,min}_{x \in \mathcal{C}} \ \frac{1}{2} \|x - \tilde{x}\|^2 . \tag{1.44}$$

Pour simplifier l'exposition, nous supposons tout au long de cette section que nous travaillons avec la norme euclidienne $\|\cdot\|_2$. La projection est toujours bien définie, ce qui signifie que le minimum est toujours atteint dans $\mathcal{C}$. La projection est également unique lorsque $\mathcal{C}$ est convexe [BV04]. La règle de mise à jour (1.43) peut alors être rendue admissible en ajoutant une étape de projection. Cela donne l'algorithme de descente de gradient projeté (PGD), présenté dans l'algorithme 17. L'influence de cet algorithme en mathématiques numériques est immense, et il reste à ce jour l'un des principaux ingrédients pour de nombreuses méthodes d'optimisation. L'algorithme a été introduit pour la première fois par Louis-Augustin Cauchy, qui l'a publié dans son *Compte Rendu à l'Académie des Sciences* en 1847 (voir [C$^+$47] pour l'article original, et aussi [Lem12] pour une discussion moderne du travail de Cauchy).

Algorithm 17 Descente de Gradient Projetée (PGD)

Entrée : $x_0 \in \mathcal{C}$.
for $k = 0, 1, \ldots$ **do**

$$x_{k+1} = \Pi_{\mathcal{C}} \left(x_k - \eta_k \nabla f(x_k) \right) \tag{1.45}$$

end for

1.2.1 Taux de convergence

Nous donnons ici le taux de convergence de l'algorithme 17. Notez que les méthodes de gradient projeté pour des normes générales peuvent également être étudiées, et la praticabilité des algorithmes ainsi que les taux de convergence dépendent fortement du choix de la norme (voir par exemple [Dev13]).

Theorem 1.2.1. *Soit $f : \mathbb{R}^d \to \mathbb{R}$ convexe et L-lisse par rapport à $\|\cdot\|_2$, et $\mathcal{C} \subset \mathbb{R}^d$ fermé et convexe. Les itérés de l'algorithme 17 avec une entrée $x_0 \in \mathcal{C}$ et un pas constant $\eta_k = \frac{1}{L}$ satisfont*

$$f(x_k) - f(x^*) \leq \frac{L \|x_0 - x^*\|_2^2}{2k}.$$

Preuve. Voir par exemple [Bec17, Chapitre 10].

Il ressort du théorème ci-dessus que la convergence de l'algorithme de gradient projeté est de l'ordre $O(\frac{1}{k})$, si on omet les constantes. Il se trouve qu'une légère modification de l'algorithme, souvent appelée *accélération de Nesterov* [Nes83], permet d'obtenir une convergence $O(\frac{1}{k^2})$, et que ce taux est optimal pour les méthodes du premier ordre se basant sur des projections [Nes03].

1.3 L'algorithme de Frank-Wolfe

Dans certains cas pratiques, l'opérateur de projection (1.44) peut être difficile à traiter ou trop coûteux à calculer. L'algorithme de Frank-Wolfe est une manière d'éviter les projections en remplaçant l'oracle quadratique (1.44) par un oracle linéaire. Il a été introduit pour la première fois par Marguerite Frank et Philip Wolfe [FW56]. La méthode de Frank-Wolfe est également appelée méthode de gradient conditionnel. Cependant, pour simplifier l'exposition, nous utilisons le terme Frank-Wolfe pour désigner l'algorithme original de [FW56], et le terme méthodes de gradient conditionnel (ou variantes de Frank-Wolfe) pour toutes les méthodes de premier ordre s'appuyant sur des oracles de minimisation linéaire.

Commençons par motiver l'algorithme de Frank-Wolfe en réécrivant la mise à jour de la descente de gradient projetée dans le cadre euclidien. Comme vu dans la section 1.2, étant

donné $x_k \in \mathcal{C}$, la descente de gradient met à jour $x_{k+1} = x^+$, où x^+ est donné par :

$$\begin{aligned}
& x^+ = \Pi_{\mathcal{C}}\left(x_k - \eta_k \nabla f(x_k)\right) \\
\iff & x^+ \in \arg\min_{x \in \mathcal{C}} \ \frac{1}{2}\left\|x - (x_k - \eta_k \nabla f(x_k))\right\|_2^2 \\
\iff & x^+ \in \arg\min_{x \in \mathcal{C}} \ \frac{\eta_k^2}{2}\left\|\nabla f(x_k)\right\|_2^2 + \eta_k \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2}\left\|x - x_k\right\|_2^2 \\
\iff & x^+ \in \arg\min_{x \in \mathcal{C}} \ f(x_k) + \langle \nabla f(x_k), x - x_k\rangle + \frac{1}{2\eta_k}\left\|x - x_k\right\|_2^2,
\end{aligned}$$

où, dans la dernière équivalence, nous avons ajouté et retiré des termes qui ne dépendent pas de x. Cette équivalence montre qu'à chaque itération k, la descente de gradient projeté minimise une approximation quadratique de f en x_k. Cette formulation est courante et permet des généralisations intuitives à d'autres normes [Dev13]. La motivation derrière les méthodes de gradient conditionnel est de supprimer le terme quadratique et de simplement minimiser l'approximation linéaire de f en x_k :

$$x^+ \in \arg\min_{x \in \mathcal{C}} \ f(x_k) + \langle \nabla f(x_k), x - x_k\rangle, \tag{1.46}$$

ou, en supprimant les termes inutiles,

$$x^+ \in \arg\min_{x \in \mathcal{C}} \ \langle \nabla f(x_k), x\rangle. \tag{1.47}$$

Ce calcul est communément appelé *oracle de minimisation linéaire.*

Definition 1.3.1. *Étant donné un ensemble convexe compact $\mathcal{C} \subset \mathbb{R}^d$, l'oracle de minimisation linéaire (ou LMO) pour $y \in \mathbb{R}^d$ est défini comme suit :*

$$LMO(y) \in \arg\min_{x \in \mathcal{C}} \ \langle y, x\rangle. \tag{1.48}$$

Étant donné x_k et le résultat du LMO, noté x^+, la méthode de Frank-Wolfe met à jour l'itéré suivant comme une combinaison convexe de x^+ et x_k. Nous résumons la méthode dans l'algorithme 18.

Algorithm 18 Frank-Wolfe

Entrée : $x_0 \in \mathcal{C}$
for $k = 0, 1, \ldots$ **do**
  $s_k = \mathrm{LMO}(\nabla f(x_k))$
  (Variante 1) : $\eta_k = \frac{2}{k+2}$
  (Variante 2) : $\eta_k \in \arg\min_{\eta \in [0,1]} f(x_k + \eta(s_k - x_k))$
  (Variante 3) : $\eta_k = \min\left\{1, \frac{\langle \nabla f(x_k), x_k - s_k\rangle}{C_f}\right\}$
  $x_{k+1} = (1 - \eta_k)x_k + \eta_k s_k$
end for

L'opérateur de projection (1.44) est une minimisation quadratique sur l'ensemble admissible, tandis que LMO (1.48) est une minimisation linéaire sur cet ensemble. Dans de nombreuses applications pratiques, la minimisation linéaire est moins coûteuse que la projection; cela inclut, par exemple, les boules de norme ℓ_p pour $p \in]1, 2[\cup]2, +\infty[$, la boule de norme nucléaire, de nombreux polytopes d'intérêt et la boule de variation totale. Pour en savoir plus sur les avantages computationnels des minimisations linéaires par rapport aux projections, nous renvoyons le lecteur à [HK12, HJN15, CP21].

1.3.1 Taux de convergence

Contrairement aux méthodes de gradient projeté, dont l'analyse repose souvent sur l'hypothèse d'une fonction objectif lisse, les méthodes de gradient conditionnel sont généralement étudiées à l'aide de la notion de *courbure* de la fonction objectif [Cla10, Jag13, LJJ13]. Intuitivement, la courbure borne l'écart de la fonction f par rapport à son approximation au premier ordre sur un segment de droite donné.

Definition 1.3.2. *La courbure C_f de la fonction f sur l'ensemble $\mathcal{C}$ est définie comme :*

$$C_f = \sup_{x,v\in\mathcal{C},\ \eta\in[0,1]\ y=x+\eta(v-x)} \left\{ \frac{2}{\eta^2} \left(f(y) - f(x) - \langle \nabla f(x), y-x \rangle \right) \right\}.$$

Si f est L-lisse par rapport à $\|\cdot\|$, on a toujours (voir [Jag13] pour une preuve) :

$$C_f \leq L\text{diam}(\mathcal{C}, \|\cdot\|)^2. \tag{1.49}$$

Nous présentons maintenant le principal résultat de convergence pour l'algorithme 18. La preuve est un résultat classique en optimisation convexe (voir par exemple [Cla10, Jag13]).

Theorem 1.3.1. *Soit $\mathcal{C}$ compact et convexe, et $f : \mathbb{R}^d \to \mathbb{R}$ une fonction convexe avec une courbure C_f. Les itérés de l'algorithme 18 satisfont:*

$$f(x_k) - f(x^*) \leq \frac{2C_f}{k+1}.$$

La convergence de l'algorithme de Frank-Wolfe est donc de l'ordre $O(\frac{1}{k})$, comme celle de l'agorithme de gradient projeté. En revanche, au contraire des méthodes de projection, il n'est pas possible d'accélerer la convergence de Frank-Wolfe pour obtenir un taux $O(\frac{1}{k^2})$ [Lan13]. L'algorithme de Frank-Wolfe et ses variantes possèdent cependant différentes propriétés qui les rendent très intéressants dans de nombreuses applications, en plus du bas coût de calcul des minimisations linéaires par rapport au coût des projections. Nous listons certaines de ces propriétés maintenant.

- Les algorithmes de Frank-Wolfe ne sont pas impactés par le choix de la norme, que ce soit dans leur implémentation et (le plus souvent) dans les taux de convergence théoriques.
- L'algorithme de Frank-Wolfe est invariant après une transformation affine du problème originel.
- Chaque minimisation linéaire sur l'ensemble des contraintes a pour solution un point extrême de l'ensemble. Cela implique que les premiers itérés des algorithmes de Frank-Wolfe sont des combinaisons convexes parsimonieuses de points extrêmes de l'ensemble des contraintes, ce qui est très utile dans plusieurs cas pratiques [SSGS11, FGM17, CP23].
- A chaque itération k de Frank-Wolfe, il est possible de calculer une borne supérieure sur le saut de dualité à l'itéré x_k. Cela permet d'obtenir un critère d'arrêt pour l'algorithme, sans coût de calcul supplémentaire. De plus, ce saut de dualité est assuré de converger vers zero à un taux de $O(\frac{1}{k})$ [Jag13, PNAJ20].

1.4 Contributions de la thèse

Nous présentons ici les différentes contributions de cette thèse.

Chapter 2 : Afin d'atteindre une précision ϵ pour des problèmes d'optimisation convexes et lisses, nous avons vu que la méthode de Frank-Wolfe nécessite $O(1/\epsilon)$ itérations, tandis que

les méthodes de gradient projeté accélérées nécessitent seulement $O(1/\sqrt{\epsilon})$ itérations. Dans ce chapitre, nous prouvons qu'en utilisant la parallélisation, il est possible d'atteindre des taux accélérés avec une variante de l'algorithme de Frank-Wolfe. Plus précisément, nous montrons que cette variante nécessite $\tilde{O}(1/\sqrt{\epsilon})$ itérations pour atteindre un saut de dualité de ϵ, lorsqu'il est possible d'utiliser $O(1/\sqrt{\epsilon})$ unités de calcul en parallèle pour effectuer les minimisations linéaires sur l'ensemble admissible. Cette variante est basée sur un algorithme de type Bregman avec convergence accélérée pour des problèmes de minimisation stochastique composite. Cet algorithme de Bregman présente un intérêt théorique et pratique en tant que tel, indépendamment de son application dans le cadre de Frank-Wolfe. Ce chapitre est basé sur l'article [DTBBT22] :

- Benjamin Dubois-Taine, Francis Bach, Quentin Berthet et Adrien Taylor. 'Fast Stochastic Composite Minimization and an Accelerated Frank-Wolfe Algorithm under Parallelization'. *Neural Information Processing Systems (NeurIPS)*, 2022.

Chapter 3 : Comme mentionné précédemment, une caractéristique importante des méthodes de gradient conditionnel est que chaque itéré est une combinaison convexe de points extrêmes de l'ensemble admissible. En effet, on peut supposer que chaque étape de minimisation linéaire retourne un point extrême. Dans ce chapitre, nous exploitons cette propriété pour résoudre des problèmes d'optimisation non convexes séparables avec des contraintes affines. Pour ces problèmes, le théorème de Shapley-Folkman fournit une borne sur le saut de dualité qui dépend de la non-convexité des fonctions composantes et de la dimension sous-jacente des contraintes affines, mais il ne fournit pas de méthode pour obtenir une solution atteignant cette borne. Nous proposons un algorithme en deux étapes pour construire une telle solution. La première étape approxime la valeur optimale duale en appliquant un algorithme de Frank-Wolfe sur l'épigraphe de la relaxation convexe du problème initial. Ici, l'oracle de minimisation linéaire est équivalent au sous-gradient de la conjugué de Fenchel des fonctions composantes. La seconde étape consiste à ajuster la combinaison convexe obtenue via des algorithmes de représentation de Carathéodory. Par exemple, des représentations approchées de Carathéodory peuvent être obtenues en utilisant à nouveau des méthodes de gradient conditionnel. Alternativement, nous proposons un algorithme qui calcule des représentations coniques exactes de Carathéodory avec une probabilité de 1. Globalement, la solution reconstruite atteint le saut de dualité prédit par Shapley-Folkman avec une erreur additive arbitrairement petite. Ce chapitre est basé sur l'article [DTd24] :

- Benjamin Dubois-Taine et Alexandre d'Aspremont. 'Frank-Wolfe meets Shapley-Folkman: a systematic approach for solving nonconvex separable problems with linear constraints'. *arXiv preprint arXiv:2406.18282*, 2024 (Soumis).

Chapter 4 : Enfin, le dernier chapitre de cette thèse n'est pas directement lié aux méthodes de gradient conditionnel. Nous nous concentrons ici sur le problème de déroulement de phase en norme ℓ_1, qui vise à reconstruire une image à partir de son observation modulo 2π. En termes d'optimisation, il s'agit d'un problème simple de régularisation parcimonieuse, mais en très grande dimension. Nous suggérons d'utiliser une approche par moindres carrés repondérés itérativement, qui remplace l'objectif non différentiable en norme ℓ_1 par une séquence de sous-problèmes de moindres carrés pondérés. Nous résolvons ces sous-problèmes à l'aide d'une méthode de gradient conjugué préconditionné soigneusement conçue. Nous montrons que la méthode qui en résulte bénéficie d'un taux de convergence sous-linéaire en termes de valeurs de fonction. De plus, la méthode est simple à implémenter et facilement portable sur GPU, et nous montrons à travers des expériences approfondies que l'implémentation obtenue surpasse les méthodes de déroulement de phase actuelles. Ce chapitre est basé sur l'article [DTAd24] :

- Benjamin Dubois-Taine, Roland Akiki et Alexandre d'Aspremont. 'Iteratively Reweighted Least Squares for Phase Unwrapping'. *arXiv preprint arXiv:2401.09961*, 2024 (Soumis).

RÉSUMÉ

Les méthodes de gradient conditionnel, également connues sous le nom de méthodes de Frank-Wolfe, sont des algorithmes itératifs de premier ordre pour résoudre des problèmes d'optimisation convexe. Contrairement aux méthodes de descente de gradient, qui reposent sur des projections, les méthodes de gradient conditionnel s'appuient sur une minimisation linéaire sur l'ensemble des contraintes. Cette propriété est particulièrement avantageuse dans de nombreuses applications d'apprentissage automatique et de traitement du signal, où la projection sur l'ensemble des contraintes est souvent soit infaisable, soit trop coûteuse.
Dans la première partie de cette thèse, nous développons un algorithme stochastique proximal accéléré et exploitons des résultats récents sur le lissage de fonctions de support pour concevoir une variante de la méthode de Frank-Wolfe atteignant un taux de convergence accéléré similaire à celui de la descente de gradient accélérée, lorsque des unités de calcul sont disponibles en parallèle pour calculer l'oracle de minimisation linéaire.
Dans la deuxième partie de cette thèse, nous étudions des problèmes d'optimisation non convexes séparables avec des contraintes affines. Nous montrons qu'il est possible d'approcher la valeur duale optimale en exécutant Frank-Wolfe sur l'épigraphe de la relaxation convexe du problème, et que la solution obtenue peut être réduite en utilisant des algorithmes calculant des représentations (approximées) de Carathéodory. Nous montrons que la solution du problème primal ainsi construite atteint la borne classique sur le saut de dualité obtenue par le théorème de Shapley-Folkman, à un terme additif arbitrairement petit près.
Enfin, la dernière partie de cette thèse se concentre sur le problème de déroulement de phase en norme ℓ_1, qui vise à reconstruire une image à partir de son observation modulo 2π. Nous proposons d'utiliser une approche de moindres carrés itérativement repondérés, couplée à un algorithme de gradient conjugué préconditionné. Nous montrons que la méthode obtenue bénéficie d'un taux de convergence sous-linéaire. Nous démontrons à travers de nombreuses expériences que l'implémentation sur processeur graphique (GPU) de cette méthode surpasse les méthodes de déroulement de phase de l'état de l'art.

MOTS CLÉS

Optimisation du premier ordre ⋆ Frank-Wolfe ⋆ Méthodes de gradient conditionnel ⋆ Applications

ABSTRACT

Conditional gradient methods, also known as Frank-Wolfe methods, are first-order iterative algorithms to solve convex optimization problems. As opposed to gradient descent methods, which rely on projections, conditional gradient methods rely on linear minimizations on the feasible set. This property is particularly advantageous in numerous machine learning and signal processing applications, where projecting onto the feasible set is often either intractable or prohibitively expensive.
In the first part of this thesis, we develop a stochastic accelerated proximal algorithm and leverage recent results on smoothed support functions to design a variant of the Frank-Wolfe method achieving accelerated rates similar to those of accelerated gradient descent, when computing units are available in parallel to compute the linear minimization oracle.
In the second part of this thesis, we study nonconvex separable optimization problems with affine constraints. We show that one can approximate the optimal dual value by running Frank-Wolfe on the epigraph of the convex relaxation of the problem, and that the obtained solution can be trimmed down using algorithms computing (approximate) Carathédory representations. We show that this yields a solution to the primal problem achieving the classical duality gap bound obtained by the Shapley-Folkman theorem up to an arbitrarily small additive term.
Finally, the last part of this thesis focuses on the ℓ_1-norm phase unwrapping problem, which aims at recovering an image from its observation modulo 2π. We use the iteratively reweighted least squares approach coupled with a preconditioned conjugate gradient algorithm. We show that the resulting method enjoys a sublinear convergence rate. We demonstrate through extensive experiments that the resulting GPU implementation outperforms state of the art phase unwrapping methods.

KEYWORDS

First-order optimization ⋆ Frank-Wolfe ⋆ Conditional gradient methods ⋆ Applications