Constructive approaches to worst-case complexity analyses of gradient methods for convex optimization : contributions, new insights, and novel results

Baptiste Goujaud

▶ **To cite this version:**

Baptiste Goujaud. Constructive approaches to worst-case complexity analyses of gradient methods for convex optimization : contributions, new insights, and novel results. Optimization and Control [math.OC]. Institut Polytechnique de Paris, 2024. English. NNT : 2024IPPAX039 . tel-04746799

HAL Id: tel-04746799

https://theses.hal.science/tel-04746799v1

Submitted on 21 Oct 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2024IPPAX039

Thèse de doctorat

ÉCOLE POLYTECHNIQUE

IP PARIS

Constructive approaches to worst-case complexity analyses of gradient methods for convex optimization: contributions, new insights, and novel results

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à l'École polytechnique

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

Thèse présentée et soutenue à Palaiseau, le 5 avril 2024, par

BAPTISTE GOUJAUD

Composition du Jury :

Antonin Chambolle Professeur, Université Paris Dauphine-PSL (CEREMADE)	Rapporteur
Laurent Lessard Professor, Northeastern University (Department of Mechanical & Industrial Engineering (MIE))	Rapporteur
Andrea Simonetto Professor, ENSTA (Unité de mathématiques appliquées (UMA))	Président / Examinateur
Hamza Fawzi Professor, University of Cambridge (Department of Applied Mathematics and Theoretical Physics (DAMTP))	Examinateur
Jelena Diakonikolas Assistant Professor, University of Wisconsin-Madison (Department of Statistics)	Examinateur
Aymeric Dieuleveut Professeur, Ecole Polytechnique (CMAP)	Directeur de thèse
Adrien Taylor Chargé de recherche, INRIA (SIERRA)	Co-directeur de thèse

Abstract

In the current era marked by an unprecedented surge in available data and computational prowess, the field of machine learning, and more specifically deep learning, has witnessed an extraordinary evolution. Machine learning algorithms heavily rely on optimization techniques to tune their parameters and enhance predictive accuracy. Among the myriad of optimization approaches, the first-order optimization methods have emerged as cornerstones, demonstrating a remarkable balance between efficacy and computational efficiency. Crucially, the development of strong optimization theory is pivotal in unraveling the full potential of first-order optimization. Theoretical underpinnings not only deepen our understanding of optimization landscapes but also pave the way for the design of novel algorithms. The momentum-based algorithms have proven their effectiveness by significantly accelerating training procedures. The conceptual foundation provided by optimization theory has enabled the formulation of momentum, turning theoretical insights into a powerful and widely adopted practical optimization tool.

The role of this thesis is to pursue and accelerate the effort to develop a strong theoretical foundation of first-order optimization. We proved various results, exploiting the general structures of the certificate proofs. (i) We used the link between quadratic optimization and polynomial theory to explain empirically observed phenomena. (ii) We implemented a Python package to support the *Performance estimation* framework. (iii) We wrote a tutorial to explain how to derive natural proofs in optimization based on this framework. (iv) We applied this methodology, with the help of our Python package, to derive a complete first-order optimization theory on a very large class of functions. (v) We complemented the theoretical *Performance estimation* framework to disprove the convergence of a specific family of methods and applied it to the famous Heavy-ball method to provably disprove an acceleration over the class of smooth and strongly convex functions.

Résumé

À l'heure actuelle, caractérisée par une croissance sans précédent des données disponibles et des capacités computationnelles, le domaine de l'apprentissage automatique, et plus particulièrement de l'apprentissage profond, a connu une évolution exceptionnelle. Les algorithmes d'apprentissage automatique reposent largement sur des techniques d'optimisation pour ajuster leurs paramètres et améliorer leurs prédictions. Parmi les différentes approches d'optimisation, les méthodes du premier ordre ont émergé comme des fondements incontournables, démontrant un équilibre notable entre rapidité et précision. Il est aujourd'hui crucial de développer une théorie solide de l'optimisation du premier ordre pour en exploiter pleinement le potentiel. Ces fondements théoriques approfondissent notre compréhension des algorithmes d'optimisation actuels et ouvrent la voie à la création d'algorithmes innovants. L'efficacité démontrée du concept de momentum dans l'accélération significative de la convergence de problèmes réels témoigne de l'importance de la théorie de l'optimisation. Cette théorie a permis la formulation du momentum, transformant des intuitions théoriques en un outil d'optimisation pratique, largement adopté.

Cette thèse vise à poursuivre et accélérer les efforts visant à développer une base théorique solide de l'optimisation du premier ordre. Nous avons présenté plusieurs résultats en exploitant les structures générales des certificats de preuves. (i) Le lien entre l'optimisation quadratique et la théorie des polynômes a été utilisé pour expliquer des phénomènes observés empiriquement. (ii) Un package Python a été mis en place pour faciliter l'utilisation du framework d'*estimation de performance*. (iii) Un tutoriel détaillé expliquant la dérivation de preuves naturelles en optimisation basée sur ce cadre a été rédigé. (iv) En utilisant notre package Python, nous avons appliqué cette méthodologie pour dériver une théorie complète de l'optimisation du premier ordre sur une vaste classe de fonctions. (v) Le framework théorique d'*estimation de performance* a été étendu pour réfuter la convergence d'une famille spécifique de méthodes, démontrant finalement la non-accélération de la célèbre méthode "Heavy-ball" sur la classe des fonctions lisses et fortement convexes.

Remerciements

Chaque rencontre, fortuite ou non, peut grandement impacter notre vie. Mon parcours et le présent travail de recherche en sont une bonne illustration et auraient sans doute été bien différents si je n'avais pas croisé la route de chacune des personnes citées ci-après.

Tout d'abord, j'ai une pensée pour le corps enseignant dans son ensemble qui m'a, petit à petit, aiguillé jusqu'ici. Je remercie en particulier Madame De Peretti et Monsieur Imperor qui ont parié sur moi et ont permis mon admission en classe préparatoire bien que mon dossier de lycée ne fasse pas partie des meilleurs et n'ait retenu l'attention d'aucune école parisienne. Au lycée Blaise Pascal d'Orsay, pourtant réputé pour son excellent niveau et ses très bons résultats aux concours, je n'ai pas ressenti l'ambiance pesante et stressante qui accompagne souvent les récits d'étudiants. Être admis dans cette classe aura sans doute été l'une des meilleures choses qui pouvaient m'arriver.

Mon parcours m'a ensuite mené à l'Ecole Normale Supérieure de Cachan où Monsieur Pascal, professeur du département de Mathématiques, a su me motiver et m'aider à définir mon orientation, d'abord en me poussant vers l'agrégation, puis vers le MVA en dernière année. Durant ces 4 années, j'ai également fait la connaissance de Ritavan, devenu l'un des plus proches et avec lequel j'ai partagé un stage à Grenoble et beaucoup de bons moments. J'honorerai bientôt ma promesse de venir le voir à Munich.

Me voici désormais pour une année au sein des équipes d'Apple, à Portland, où Bruno et Louise m'ont accueilli à bras ouverts. À mes débuts là-bas, la programmation n'était pas mon point fort mais Bruno, armé de sa bienveillance, a pris le temps de me former et de m'accompagner tout au long de mon séjour. Bruno et sa femme sont des personnes formidables et je garde les meilleurs souvenirs de chaque soirée passée en leur compagnie.

Mes pas me mènent ensuite à Montréal où Ioannis Mitliagkas me convertit à l'optimisation, qui deviendra plus tard mon sujet de thèse, et où je retrouve Charles, un ancien ami de l'ENS. Notre expatriation nous rapproche et me permet de rencontrer, par son biais, Léonard. Je n'oublierai jamais les soirées passées ensemble à l'escalade ou au MILA, à jouer au pingpong entre deux calculs sur tableau blanc en plein milieu de la nuit.

Tout ce périple m'amène finalement à la réalisation de la thèse vers laquelle on m'a poussé toute ma scolarité, à croire que j'étais le seul à ne pas l'avoir envisagé comme une étape incontournable de mon parcours.

Comme la classe préparatoire, vivre une thèse peut être une formidable aventure ou une période difficile. J'ai là aussi eu la chance d'être parfaitement bien entouré et conseillerais à chaque futur doctorant de bien choisir ses superviseurs : leur impact n'est pas à négliger. Grâce à eux, j'ai pu travailler dans d'excellentes conditions, prélude essentiel à la *compréhension des fonctions quadratiques*, et produire un travail, je l'espère, de qualité. J'ai eu la chance d'être entouré des meilleurs superviseurs qu'un doctorant puisse rêver d'avoir. J'adresse donc mes remerciements les plus sincères à Aymeric et Adrien pour leur encadrement mais, surtout, pour le soutien constant qu'ils m'ont manifesté pendant toute

la durée de cette thèse.

J'ai également une pensée pour toutes les personnes du CMAP et plus particulièrement pour l'équipe d'Aymeric, à commencer par ceux qui sont là depuis mes débuts : Margaux et Constantin, mais aussi pour Rémi, Jean-Baptiste, Renaud, Mahmoud et Damien.

L'aventure de la thèse ne s'écrit pas seul. Je profite donc de ces quelques lignes pour remercier également mes coauteurs Damien, Fabian, Céline, François et Julien sans lesquels ce manuscrit ne serait pas ce qu'il est. Je terminerai mes remerciements professionnels par les membres de mon jury et plus particulièrement par mes deux rapporteurs qui ont accepté de me consacrer du temps et de me fournir de précieux retours.

Cette page de ma carrière se conclut avec une pensée chaleureuse pour mes amis de longue date et ma famille. Il est difficile de résumer leur impact sur ma vie et sur mes choix tant leur présence à mes côtés aura rythmé les trois dernières décennies (ou presque). Je me contenterai donc d'un grand "merci pour tout".

Enfin, merci à Caroline d'être à mes côtés au jour le jour, même lorsqu'elle m'entend dériver sur la clôture algébrique de $\mathbb{C}$ avec Aymeric en visio à 4 heures du matin (bien que je la soupçonne parfois d'envoyer Madison et Pumpkin s'interposer lorsque ce genre de discussion dérive trop longtemps ...).

Contents

0

Introduction

Contents

Dans cette thèse, nous nous intéressons aux problèmes d'optimisation de la forme

$$f_\star \triangleq \min_{x \in \mathbb{R}^d} f(x), \tag{OPT}$$

où l'on suppose que la fonction à valeur réelle $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ (propre, i.e., non identiquement égale à $+\infty$) possède un ensemble de minimiseurs globaux, noté $\mathcal{X}_\star$, compacte non vide.

Bien que l'optimisation soit un vieux problème (Cauchy, 1847), elle a trouvé de nombreuses applications au fil du temps. En effet, de nombreux problèmes pratiques peuvent être réduits sous la forme (OPT). *L'apprentissage automatique* et la *vision par ordinateur* sont des exemples de domaines ayant récemment suscité beaucoup d'intérêt (voir Krenn et al., 2022, Figure 1) et reposant fortement sur l'optimisation.

Par conséquent, comprendre les algorithmes d'optimisation existants et en développer de nouveaux est devenu un défi majeur. La théorie de l'optimisation a déjà connu des succès dans le passé, notamment en introduisant des méthodes telles que la *descente de gradient* (GD) (Cauchy, 1847) ou des méthodes basées sur le momentum telles que les méthodes *Heavy-ball* (HB) (Polyak, 1963) et *Fast Gradient* (FGM) (Nesterov, 1983). Fait intéressant, étant si efficaces, ces méthodes sont même utilisées bien au-delà de leur domaine de convergence théorique, par exemple dans l'apprentissage profond.

Cette introduction décrit les notions clés utilisées dans les chapitres suivants ainsi que l'organisation des différentes parties. La partie I traite de l'optimisation quadratique tandis que nous nous concentrons sur l'optimisation non quadratique dans la partie II. Ces deux parties utilisent des outils différents. Nous décrivons brièvement les deux contextes dans la section 0.2 et la section 0.3. Enfin, nous résumons nos contributions dans la section 0.4.

0.1 Oracles et performance des algorithmes

Les propriétés de convergence de l'optimisation de premier ordre sont généralement étudiées à travers des analyses de pire cas sous le modèle de la *boîte noire* (Nemirovskii and Yudin, 1983a).

Optimisation en boîte noire et oracles. Le modèle en boîte noire suppose que la procédure n'a pas accès à une description analytique complète de la fonction objectif, mais seulement à des informations discrètes partielles fournies par ce qu'on appelle des oracles. En d'autres termes, un algorithme propose un premier point (ou itéré) $x_0 \in \mathbb{R}^d$ et interroge un oracle $\mathcal{O}$ qui renvoie $\mathcal{O}(x_0)$. À partir de ces informations, l'algorithme choisit l'itéré suivant, et ainsi de suite. Les exemples les plus classiques d'oracles sont les oracles d'ordre n définis comme $\mathcal{O}(x) \triangleq (f(x), \nabla f(x), \cdots, \nabla^n f(x))$, en particulier les oracles d'ordre zéro, un ou deux.

Comment choisir un oracle? Plus l'ordre de l'oracle est élevé, plus nous obtenons d'informations. Il semble donc naturel de considérer un oracle d'ordre élevé lorsque c'est possible. Les méthodes du second ordre sont connues pour converger rapidement, du moins localement. Cependant, les hessiennes peuvent être difficiles ou coûteuses à calculer, voire simplement inexistantes. Par exemple, en l'optimisation à grande échelle, les hessiennes sont trop volumineuses pour être correctement manipulées (par exemple, inversées) et parfois même pour être stockées en mémoire.

D'autre part, un oracle d'ordre zéro est généralement bon marché et donc pratique pour les problèmes à grande échelle. Ils sont également utilisés pour optimiser des fonctions non différentiables et des fonctions définies sur un ensemble discret, rencontrés par exemple lors de l'optimisation des hyperparamètres des modèles d'apprentissage automatique (Bergstra et al., 2011).

Par conséquent, il est nécessaire de trouver un compromis entre l'efficacité d'un oracle et son coût. Les modèles différentiables de grande dimension sont souvent bien adaptés aux oracles de premier ordre, surtout lorsque le modèle bénéficie d'une structure de graphe, adaptée pour appliquer une règle de dérivation des composées implémentée dynamiquement (Rumelhart et al., 1986).

En raison du succès pratique des méthodes de premier ordre dans des applications à grande échelle telles que l'apprentissage profond, leurs analyses pour minimiser de telles fonctions dans le modèle en boîte noire (voir par exemple (Nemirovskii and Yudin, 1983a; Polyak, 1987; Nesterov, 2003)) attirent beaucoup d'attention et sont au centre de cette thèse. Notez cependant que, dans certaines situations, d'autres oracles peuvent être considérés, tels que les oracles de gradient inexact (voir De Klerk et al. (2020)), les oracles de gradient stochastique (voir (Robbins and Monro, 1951)), ou les oracles proximaux (voir Rockafellar (1976); Combettes and Pesquet (2011); Chambolle and Pock (2011)).

Méthodes classiques du premier ordre. L'un des premiers algorithmes introduits est la méthode de *descente de gradient* (GD), définie comme

$$x_{t+1} = x_t - \gamma \nabla f(x_t). \tag{GD}$$

GD consiste à se déplacer dans la direction la plus raide localement. Il s'agit donc d'un algorithme glouton, se déplaçant toujours orthogonalement aux ensembles de niveaux visités. La figure 0.1 montre que cette procédure peut être très lente dans certains cas. Cela est dû aux oscillations des itérations. Des mouvements plus uniformes accélèrent cette procédure. C'est l'idée principale derrière les méthodes basées sur le momentum telles que les méthodes *Heavy-ball* (HB) et *Fast Gradient* (FGM), définies comme suit:

$$x_{t+1} = x_t - \gamma \nabla f(x_t) + \beta(x_t - x_{t-1}) \tag{HB}$$

$$x_{t+1} = x_t - \gamma \nabla f\left(x_t + \beta(x_t - x_{t-1})\right) + \beta(x_t - x_{t-1}) \tag{FGM}$$

Ces trois algorithmes sont les plus mentionnés dans cette thèse.

Performance des algorithmes et régularité. Un algorithme d'optimisation peut être vu comme un décideur, devant produire l'itéré suivant en fonction de toutes les informations obtenues jusqu'à présent de l'oracle. En d'autres termes, après chaque appel à l'oracle, nous obtenons une connaissance *locale* de la fonction objectif. En conséquence, nous réduisons l'ensemble des fonctions possibles compatibles avec les informations obtenues à chaque étape. Notre objectif ultime est de trouver approximativement le minimiseur de la fonction objectif aussi rapidement que possible. Cependant, sans contrainte sur la fonction, la connaissance locale fournie par l'oracle n'est pas suffisante pour effectuer cette tâche avec précision. En effet, on peut facilement se convaincre que si aucune hypothèse n'est faite sur f, quelles que soient les valeurs et les gradients de f sur un ensemble fini de points, ses minimiseurs peuvent toujours être ailleurs et la valeur minimale de f peut toujours être inférieure à la plus petite observée. Par conséquent, si l'on attend un résultat d'un

algorithme, nous avons besoin d'une *connaissance de régularité* sur la fonction objectif, qui étend les connaissances locales fournies par l'oracle à tout l'espace.

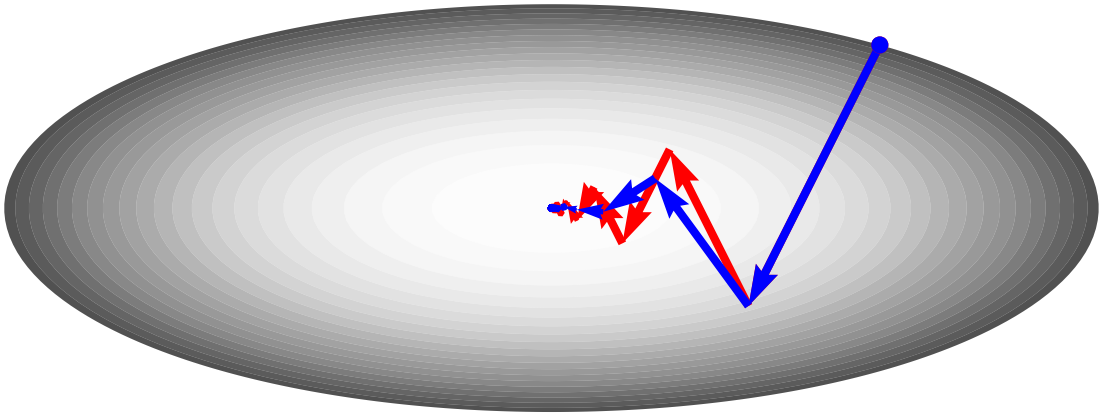


Figure 0.1: Comparaison de la vitesse de convergence de la descente de gradient et de Heavy-ball sur une fonction objectif quadratique. La méthode de descente de gradient (courbe rouge) oscille tandis que la méthode Heavy-ball (courbe bleue) effectue des mouvements plus réguliers.

Classes de fonctions. Pour rendre l'optimisation possible, nous supposons que $f \in \mathcal{F}$ pour certaines classes de fonctions $\mathcal{F}$. Cette hypothèse de régularité doit apporter des informations globales qui étendent la connaissance locale fournie par l'oracle. À titre d'exemple, une classe classique de fonctions est la classe des fonctions de gradients L-Lipschitz (ou L-lisses), définie comme

$$\{f \mid \forall x, y, \|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|\}\,.$$

Si nous supposons que cette relation est vérifiée, alors tout gradient observé localement contribue à contrôler tous les autres non observés. Remarquez que des valeurs de L plus petites sont plus restrictives, c'est-à-dire qu'elles correspondent à une hypothèse plus forte, apportant plus de connaissance, dans le sens où les bornes obtenues sur les gradients sont plus restrictives. Les principales classes que nous étudions dans cette thèse sont

$$\mathcal{Q}_\Lambda \triangleq \{x \to \tfrac{1}{2}(x - x_\star)^T H(x - x_\star) \mid \mathrm{Sp}(H) \in \Lambda\}, \tag{1}$$

$$\mathcal{F}_{\mu,L} \triangleq \{f \mid \forall x, y,\ f(x) \geqslant f(y) + \langle \nabla f(y), x - y\rangle + \tfrac{\mu}{2}\|x - y\|^2 \tag{2}$$

$$\text{et}\ \ f(x) \leqslant f(y) + \langle \nabla f(y), x - y\rangle + \tfrac{L}{2}\|x - y\|^2\},$$

$$\mathrm{QG}^+(L) \triangleq \{f \mid \forall x, y,\ f(x) \leqslant f_\star + \tfrac{L}{2}\|x - x_\star\|^2\}, \tag{3}$$

où Λ est un sous-ensemble de $\mathbb{R}_{\geqslant 0}$ et $-\infty \leqslant \mu \leqslant \max(\mu, 0) \leqslant L \leqslant +\infty$. En particulier

- lorsque $\mu > 0$, $\mathcal{F}_{\mu,L}$ est l'ensemble des fonctions μ-fortement convexes L-lisses,
- lorsque $\mu = 0$, $\mathcal{F}_{\mu,L} = \mathcal{F}_{0,L}$ est l'ensemble des fonctions convexes L-lisses,
- lorsque $\mu = -L$, $\mathcal{F}_{\mu,L} = \mathcal{F}_{-L,L}$ est l'ensemble des fonctions L-lisses,
- lorsque $\mu > 0$ et $L = +\infty$, $\mathcal{F}_{\mu,L} = \mathcal{F}_{\mu,+\infty}$ est l'ensemble des fonctions μ-fortement convexes,
- lorsque $\mu = 0$ et $L = +\infty$, $\mathcal{F}_{\mu,L} = \mathcal{F}_{0,+\infty}$ est l'ensemble des fonctions convexes propres et fermées.

Types d'analyse de performance. Soit f une fonction. Étant donné un budget de calcul, nous pouvons comparer la performance de 2 algorithmes d'optimisation sur cette fonction objective f. Maintenant, étant donné une classe de fonctions $\mathcal{F}$, *comment pouvons-nous comparer deux algorithmes?* Dans de rares cas, nous trouvons un algorithme qui est meilleur que tous les autres sur *chaque fonction de la classe* $\mathcal{F}$. C'est le cas de certains algorithmes tels que celui que nous proposons dans le chapitre 2. Cependant, la plupart du temps, un algorithme donné est meilleur qu'un autre sur une fonction spécifique, et pire sur une autre fonction. Nous devons donc définir une métrique. De nombreux types d'analyse sont utilisés en théorie de l'optimisation, parmi lesquels les 2 suivants:

1. L'analyse de pire cas (voir Nemirovskii (1992, 1994)) consiste à définir la performance d'un algorithme sur une classe comme étant sa pire performance sur les fonctions de cette classe. Bien que l'analyse du pire cas soit largement utilisée pour expliquer la performance de nombreuses méthodes d'optimisation (voir par exemple, Nesterov, 1983), elle peut parfois être non représentative de la complexité observée lorsque le problème du pire cas n'est jamais rencontré en pratique.

2. L'analyse de cas moyen (voir Borgwardt (1977, 1980)) surmonte ce problème en moyennant la performance d'un algorithme sur toutes les fonctions de la classe d'intérêt. Bien que moins étudiée, l'analyse du cas moyen a récemment été employée dans l'optimisation quadratique (voir par exemple Pedregosa and Scieur (2020); Scieur and Pedregosa (2020); Cunha et al. (2022)). Cependant, ce type d'analyse dépend de la distribution de probabilité sur les problèmes, qui est choisie arbitrairement.

L'analyse de pire cas est la plus courante, la première analyse généralement effectuée pour tenter d'expliquer la complexité observée, et celle sur laquelle nous nous concentrons dans cette thèse.

Bornes supérieures en pire cas. En l'analyse de pire cas, une borne supérieure est donc une garantie qui s'applique à chaque fonction de la classe étudiée. Par exemple, dans cette thèse, nous prouvons les résultats suivants:

1. Soit $\mu_1 < L_1 < \mu_2 < L_2$ des nombres réels positifs avec $L_1 - \mu_1 = L_2 - \mu_2$. Nous prouvons dans le corrolaire 3.3.2 qu'une version de (HB) vérifie

$$\forall f \in \mathcal{Q}_{[\mu_1,L_1]\cup[\mu_2,L_2]}, \quad d(x_t, \mathcal{X}_\star) \leqslant \Big(\frac{\sqrt{\rho^2 - R^2} - \sqrt{\rho^2 - 1}}{\sqrt{1 - R^2}}\Big)^t \Big(1 + t\sqrt{\frac{\rho^2 - 1}{\rho^2 - R^2}}\Big) d(x_0, \mathcal{X}_\star)$$

   pour t pair, où $\rho = \frac{L_2+\mu_1}{L_2-\mu_1}$ et $R = \frac{\mu_2-L_1}{L_2-\mu_1}$.

2. Nous prouvons dans le théorème 6.2.4 qu'une version de (HB) vérifie

$$\forall f \in \mathrm{QG}^+(L) \cap \mathcal{F}_{0,+\infty}, \quad f(x_t) - f_\star \leqslant \frac{L}{2}\frac{1}{t+1} d(x_0, \mathcal{X}_\star)^2$$

   pour tout t.

Bornes inférieures en pire cas. D'autre part, une borne inférieure est valide dès qu'une fonction de la classe empêche l'algorithme de performer plus rapidement que cette borne. Par exemple, dans cette thèse, nous prouvons les résultats suivants:

1. Soit $\mu_1 < L_1 < \mu_2 < L_2$ des nombres réels positifs avec $L_1 - \mu_1 = L_2 - \mu_2$. Nous prouvons dans le corrolaire 3.3.2 que pour tout algorithme de premier ordre

$$\exists f \in \mathcal{Q}_{[\mu_1,L_1]\cup[\mu_2,L_2]}, \quad d(x_t, \mathcal{X}_\star) \geqslant \Big(\frac{\sqrt{\rho^2 - R^2} - \sqrt{\rho^2 - 1}}{\sqrt{1 - R^2}}\Big)^t \Big(1 + t\sqrt{\frac{\rho^2 - 1}{\rho^2 - R^2}}\Big) d(x_0, \mathcal{X}_\star)$$

pour t pair, où $\rho = \frac{L_2+\mu_1}{L_2-\mu_1}$ et $R = \frac{\mu_2-L_1}{L_2-\mu_1}$.

2. Nous prouvons dans le théorème 6.2.4 que pour tout algorithme de premier ordre

$$\exists f \in \mathrm{QG}^+(L) \cap \mathcal{F}_{0,+\infty}, \quad f(x_t) - f_\star \geqslant \frac{L}{2}\frac{1}{t+1} d(x_0, \mathcal{X}_\star)^2$$

pour tout t.

3. Soit $0 < \mu < L < +\infty$. Nous prouvons dans le théorème 8.3.6 que pour toute version stationnaire de (HB),

$$\exists f \in \mathcal{F}_{\mu,L}, \quad \|x_t - x_\star\| \geqslant \left(\frac{1 - C\kappa}{1 + C\kappa}\right)^t \|x_0 - x_\star\|$$

pour tout t.

Garanties optimales en pire cas. Une garantie optimale est à la fois une borne supérieure et une borne inférieure. Les résultats du corrolaire 3.3.2 et du théorème 6.2.4 fournissent donc des garanties optimales. Le chapitre 5 aborde le problème mathématique de trouver la garantie optimale vérifiée par un algorithme sur une classe donnée. À titre d'exemple, ce problème sur (GD) peut être formulé comme suit

$$\left|\begin{array}{ll} \underset{\substack{f\in\mathcal{F}, d\geqslant 1 \\ (x_t)_{t\leqslant T}\in(\mathbb{R}^d)^{T+1}}}{\text{maximiser}} & f(x_T) - f_\star \\ \text{sous contrainte} & \left\{\begin{array}{c} d(x_0, \mathcal{X}_\star) \leqslant 1, \\ x_{t+1} = x_t - \gamma\nabla f(x_t). \end{array}\right. \end{array}\right. \qquad (\mathcal{P})$$

Approches constructives. Ces dernières années, deux principales approches ont été proposées pour automatiser la recherche de bornes supérieures, inférieures ou optimales en analyse de pire cas. Le formalisme des Contraintes Quadratiques Intégrales (IQC) (voir Lessard et al. (2016); Lessard (2022)) se concentre principalement sur la recherche de bornes supérieures, tandis que le formalisme des Problèmes d'Estimation de Performance (PEP) (voir Drori and Teboulle (2014); Taylor et al. (2017c); Drori and Taylor (2020)) considère le problème ($\mathcal{P}$) et cherche donc des bornes optimales (en résolvant exactement ($\mathcal{P}$)) ou des bornes inférieures (en trouvant un point acceptable pour ($\mathcal{P}$)). Les approches IQC et PEP ont rencontré plusieurs succès dans la conception de certaines méthodes (voir par exemple Van Scoy et al. (2017); Cyrus et al. (2018); Taylor and Drori (2022)), et ont été combinées dans Taylor et al. (2018a). Le chapitre 5 décrit l'approche PEP pour aborder ce problème et offre une liste non exhaustive des contributions de la littérature PEP sur la structure de preuve des analyses en pire cas en optimisation.

0.2 Optimisation quadratique de premier ordre

Une classe d'intérêt est celle des fonctions convexes quadratiques $\mathcal{Q}$. Résoudre (OPT) sur $\mathcal{Q}$ permet de résoudre des régressions linéaires. De plus, toute fonction deux fois différentiable peut être approximée comme une fonction quadratique autour de ses points optimaux. Ainsi, étudier le comportement d'une méthode sur les fonctions quadratiques fournit de bonnes indications sur le comportement asymptotique de cette méthode sur toute fonction deux fois différentiable. Un bon exemple en est la méthode Heavy-ball, historiquement dérivée sur $\mathcal{Q}$ (Polyak, 1963), elle est maintenant utilisée sur des classes beaucoup plus larges, achevant de bons résultats pratiques.

Formellement, nous pouvons définir F comme $F(x) \triangleq \frac{1}{2}(x-x_\star)^T H(x-x_\star) + F_\star$ où H est la matrice Hessienne constante de F. Son gradient en x est donc $\nabla F(x) = H(x-x_\star)$. Cette paramétrisation de $\mathcal{Q}$ par la matrice Hessienne H aide à exprimer les algorithmes classiques de manière simple. Par exemple:

1. Une itération de (GD) s'écrit $x_{t+1} - x_\star = (I - \gamma H)(x_t - x_\star)$,
2. Une itération de (HB) s'écrit $x_{t+1} - x_\star = (I - \gamma H)(x_t - x_\star) + \beta(x_t - x_{t-1})$,
3. Une itération de (FGM) s'écrit $x_{t+1} - x_\star = (I - \gamma H)(x_t - x_\star + \beta(x_t - x_{t-1}))$.

De plus, $(\mathcal{P})$ peut être écrit sous la forme

$$\left| \begin{array}{ll} \underset{\substack{d \geqslant 1, H \in \mathcal{S}_d^+ \\ (x_t)_{t \leqslant T} \in \left(\mathbb{R}^d\right)^{T+1}}}{\text{maximiser}} & \frac{1}{2}(x_T - x_\star)^T H(x_T - x_\star) \\ \text{sous contrainte} & \left\{ \begin{array}{c} d(x_0, \mathcal{X}_\star) \leqslant 1, \\ x_{t+1} - x_\star = (I - \gamma H)(x_t - x_\star). \end{array} \right. \end{array} \right.$$

Pour étudier les propriétés de convergence de (GD), nous composons les itérations pour obtenir $x_t - x_\star = (I - \gamma H)^t (x_0 - x_\star)$ et réduisons l'étude des propriétés de convergence de (GD) à l'étude de la matrice $(I - \gamma H)^t$. Polyak (1964) procède de manière similaire pour étudier les méthodes de momentum. Par exemple, en définissant $X_t \triangleq \begin{pmatrix} x_t \\ x_{t-1} \end{pmatrix}$, (HB) vérifie $X_{t+1} = M X_t$ avec $M \triangleq \begin{pmatrix} (1+\beta)I - \gamma H & -\beta I \\ I & 0 \end{pmatrix}$. L'étude de (HB) se réduit à l'étude de la matrice M, permettant d'établir la convergence (voir la figure 8.1 dans le chapitre 8).

Plus généralement, toute méthode dont l'itération est de forme très générique $x_t = x_0 - \sum_{i=0}^{t-1} \gamma_i^{(t)} \nabla f(x_i)$, comme suggéré dans (Nemirovskii, 1992, 1994), peut également être explicitement écrite sous la forme $\forall t \geqslant 0, x_t - x_\star = P_t(H)(x_0 - x_\star)$ pour un polynôme P_t. Cette connexion forte entre l'optimisation quadratique et la théorie des polynômes, d'abord décrite dans Fischer (2011), est détaillée et utilisée dans les chapitres 2 et 3.

Organisation de la partie I sur l'optimisation quadratique. Dans le chapitre 2, nous utilisons la théorie des polynômes pour dériver une alternative à la méthode du gradient conjugué sous la forme d'une itération de type Heavy-ball en utilisant le pas de Polyak. Puis, dans le chapitre 3, toujours en utilisant la théorie des polynômes, nous montrons que

la méthode Heavy-ball avec un pas cyclique est optimale dans le pire des cas sur la classe des fonctions quadratiques dont les valeurs propres sont dans l'union de 2 intervalles. Cette méthode est activement utilisée en pratique et s'est avérée efficace dans des problèmes complexes. le chapitre 3 apporte la première analyse de cette méthode. Enfin, tandis que le chapitre 8 se concentre sur le comportement de (HB) sur une classe plus large de fonctions, les résultats sur les quadratiques sont rappelés et utilisés pour prouver que le comportement de convergence connu de (HB) est spécifique à la classe des fonctions quadratiques.

0.3 Au-delà de l'optimisation quadratique

Contrairement au cas spécifique de $\mathcal{Q}$, les classes de fonctions ne possèdent généralement pas de représentation paramétrique intrinsèque. Au lieu de cela, elles sont définies comme des ensembles de fonctions vérifiant certaines inégalités. Des exemples typiques de telles restrictions, mentionnées précédemment, sont:

- fonctions convexes: (voir par exemple Rockafellar (1997, §4), Nesterov (2003, Definition 2.1.1)) $\forall x, y \in \mathbb{R}^d, \quad f(y) \geqslant f(x) + \langle \nabla f(x), y - x \rangle$,
- fonctions fortement convexes: (voir par exemple Nesterov (2003, Definition 2.1.2)) $\forall x, y \in \mathbb{R}^d, \quad f(y) \geqslant f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2$,
- fonctions lisses: (voir par exemple Nesterov (2003, eq. 1.2.3)) $\forall x, y \in \mathbb{R}^d, \quad \|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|$,
- fonctions Lipschitziennes: (voir par exemple Rockafellar (1997, §10)) $\forall x, y \in \mathbb{R}^d, \quad |f(x) - f(y)| \leqslant L\|x - y\|$.

Combiner certaines de ces inégalités peut conduire à des inégalités plus fortes. Par exemple, une fonction à la fois convexe et L-lisse vérifie (voir Nesterov, 2003, Theorem 2.1.5)

$$\forall x, y \in \mathbb{R}^d, \quad f(y) \geqslant f(x) + \langle \nabla f(x), y - x \rangle + \tfrac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2. \tag{4}$$

Étant donné une classe de fonctions et un algorithme, nous pouvons dériver une garantie dans le pire des cas de l'algorithme sur la classe de fonctions. Par exemple, (GD) sur des fonctions convexes L-lisses vérifie

$$\begin{aligned}\|x_{t+1} - x_\star\|^2 =& \|x_t - x_\star\|^2 - \tfrac{2}{L} \underbrace{\langle \nabla f(x_t), x_t - x_\star \rangle}_{\geqslant f(x_t) - f_\star + \frac{1}{2L}\|\nabla f(x_t)\|^2} + \tfrac{1}{L^2}\|\nabla f(x_t)\|^2 \\ \leqslant& \|x_t - x_\star\|^2 - \tfrac{2}{L}(f(x_t) - f_\star).\end{aligned}$$

En sommant pour t de 0 à un certain T, la sommation télescopique donne

$$\|x_{T+1} - x_\star\|^2 \leqslant \|x_0 - x_\star\|^2 - \tfrac{2}{L}\sum_{t=0}^{T}(f(x_t) - f_\star),$$

prouvant en particulier que $\min_{t \in [\![0,T]\!]} f(x_t) - f_\star \leqslant \frac{L}{2}\frac{\|x_0 - x_\star\|^2}{T+1}$.

Cette courte démonstration soulève plusieurs questions: Est-ce que la garantie obtenue est optimale? Ou pouvons-nous trouver une garantie plus forte pour le même algorithme sur la même classe? En particulier, nous avons utilisé (4) au lieu des inégalités classiques

des fonctions lisses et fortement convexes. Un autre argument fonctionnerait-il sans elle? Quelles inégalités devrions-nous considérer dans des cas plus généraux? Comment devons-nous les organiser dans une preuve complexe?

Les réponses à ces questions résident dans la reformulation de ($\mathcal{P}$) sous une forme traitable. Ce domaine de travail, communément appelé *estimation de performance* et initié par Drori and Teboulle (2014); Taylor et al. (2017c), a été utilisé au cours des dernières années pour dériver des certificats d'optimisation et a été développé pour d'autres applications telles que la recherche de fonctions de Lyapunov (Taylor et al., 2018a) ou la conception de nouveaux algorithmes (Drori and Taylor, 2020). Une introduction détaillée à ce cadre peut être trouvée dans (Taylor, 2020).

Organisation de la partie II au-delà de l'optimisation quadratique. Dans le chapitre 4, nous décrivons un package Python que nous avons développé pour faciliter largement l'utilisation de ce cadre et automatiser la recherche de certificats numériques et de leurs preuves. Dans le chapitre 5, nous expliquons la structure générale de ces preuves. Dans le chapitre 6, nous utilisons à la fois le package Python et les connaissances théoriques décrits dans le chapitre 5 pour dériver une théorie complète de l'optimisation du premier ordre sur une large classe de fonctions. Enfin, dans les chapitres 7 et 8, nous complétons l'utilisation habituelle de l'estimation de performance en l'étendant à la recherche de cycles dans la séquence des itérations produites par un algorithme, réfutant sa convergence. De plus, nous l'appliquons à certaines instances de (HB), prouvant sa non-accélération sur $\mathcal{F}_{\mu,L}$.

0.4 Contributions

Dans la partie I, nous nous concentrons sur l'optimisation quadratique. Plus précisément, nous utilisons l'équivalence mentionnée ci-dessus entre l'optimisation quadratique et la théorie des polynômes telle que proposée dans Fischer (2011) pour aborder deux problèmes:

Dans le chapitre 2, nous concevons un algorithme de type Heavy-ball, ajusté avec des pas de Polyak, qui est optimal instance par instance sur $\mathcal{Q}_{0,+\infty}$, c'est-à-dire qu'il minimise $\|x_t - x_\star\|$ pour tout t et toute fonction quadratique convexe. Il est intéressant de noter que le pas de Polyak est connu comme étant optimal en une étape pour (GD) sur les fonctions convexes quadratiques, et (HB) est connu pour accélérer par rapport à (GD) sur les fonctions convexes quadratiques. Nous montrons que la stratégie optimale à adopter sur $\mathcal{Q}_{0,+\infty}$ consiste à combiner les deux approches avec un paramètre de momentum bien choisi.

Dans le chapitre 3, nous établissons un lien théorique entre deux observations empiriques. D'une part, (Loshchilov and Hutter, 2017; Smith, 2017) ont montré des résultats de pointe d'une stratégie de pas cyclique de (HB) sur différents benchmarks d'apprentissage profond. D'autre part, (Sagun et al., 2017; Papyan, 2018; Ghorbani et al., 2019; Papyan, 2019) ont étudié empiriquement les propriétés des hessiennes des réseaux neuronaux profonds montrant que leur spectre se situe le plus souvent dans l'union de deux intervalles. Sur la base de cette dernière observation, nous fournissons une condition nécessaire et suffisante que doit vérifier un algorithme pour être optimal dans le pire des cas sur $\mathcal{Q}_\Lambda$, et en particulier, nous montrons que l'algorithme optimal dans le pire des cas lorsque Λ est une union de 2 intervalles de même longueur est (HB) avec une stratégie de pas cyclique.

Dans la partie II, nous nous concentrons sur l'optimisation sur de grandes classes de fonctions, au-delà du cas quadratique. Notre travail est principalement basé sur un cadre théorique proposé dans Drori and Teboulle (2014); Drori (2014); Taylor et al. (2017c,a) pour étudier l'optimisation sur différentes classes. Nous facilitons son utilisation en fournissant un package Python et un tutoriel résumant quelques-uns des points forts essentiels et des connaissances théoriques fournies par ces outils. Nous développons également des garanties de non-convergence basées sur ce cadre.

Dans le chapitre 4, nous décrivons le package Python PEPIT que nous avons implémenté. Ce package rend l'utilisation des PEPs très facile et naturelle. Il peut être vu comme une version open source étendue en PYTHON du package MATLAB PESTO (Taylor et al., 2017b).

Plus qu'une simple automatisation de la recherche de preuves numériques de garanties en optimisation, le cadre offert par les PEPs apporte beaucoup de connaissances sur la structure de ces preuves. Dans le chapitre 5, nous passons en revue la dérivation classique de cet outil (Drori and Teboulle, 2014; Drori, 2014; Taylor et al., 2017c,a) et détaillons les conséquences que nous apprenons sur la structure de preuve à partir de l'approche PEP, comme par exemple l'ensemble minimal d'inégalités à utiliser et la bonne manière de les combiner. Nous passons également en revue les techniques pour trouver des fonctions de Lyapunov (Lessard et al., 2016; Taylor et al., 2018a). Enfin, ce cadre a été utilisé et étendu pour concevoir de nouveaux algorithmes optimaux dans le pire des cas (voir Drori and Teboulle, 2014; Kim and Fessler, 2016, 2021; Sundararajan et al., 2020; Park and Ryu, 2022; Das Gupta et al., 2023; Jang et al., 2023; Barré et al., 2023). Nous revoyons l'approche proposée dans (Drori and Taylor, 2020), inspirée des travaux de Nemirovskii (Nemirovskii, 1982; Nemirovskii and Yudin, 1983b) pour concevoir initialement des méthodes accélérées. Malheureusement, ces travaux sont difficiles à trouver, nous nous référons donc à la revue (Narkiss and Zibulevsky, 2005). Il est intéressant de noter que cette approche est compatible avec la recherche de fonctions de Lyapunov.

Dans le chapitre 6, nous étudions une classe particulière de fonctions non lisses: $\mathrm{QG}^+(L) \cap \mathcal{F}_{0,+\infty}$. Dans ce chapitre, nous dérivons des garanties optimales dans le pire des cas de variantes de (GD) et (HB) ainsi que la borne inférieure du premier ordre de cette classe, montrant que (HB) est optimal dans le pire des cas pour cette classe de fonctions. Nous décrivons également une version de cet algorithme basé sur le procédé de line-search qui atteint la même garantie sans connaissance des paramètres du problème. Nous étendons également ce résultat à d'autres classes de fonctions non lisses.

Dans le chapitre 7, nous décrivons une approche basée sur les PEPs pour automatiser la recherche de cycles des méthodes du premier ordre stationnaires. Cette approche permet de prouver la non-convergence d'un algorithme et est donc complémentaire à l'approche PEP classique.

Dans le chapitre 8, nous améliorons l'approche précédente pour la simplifier, et tirons des conclusions sur la forme générique des cycles. En particulier, nous prouvons la non-accélération de (HB) sur $\mathcal{F}_{\mu,L}$, un problème ouvert depuis très longtemps.

Principales apparitions des algorithmes.

- (GD) est étudié dans les chapitres 4, 6 et 7,
- (HB) est étudié dans les chapitres 2, 3, 6, 7 et 8,

- (FGM) et la méthode *Three operator splitting*(TOS) sont étudiés dans le chapitre 7.

Apparitions des classes de fonctions.

- $\mathcal{Q}_\Lambda$ est étudiée dans les chapitres 2, 3 et 8,
- $\mathcal{F}_{\mu,L}$ est étudiée dans les chapitres 4, 7, et 8,
- $\mathrm{QG}^+(L)$ est étudiée dans le chapitre 6.

Publications. Cette thèse est basée sur les articles suivants:

Chapitre	Article	Conférence / Journal	Atelier / Conférences invitées	Blogpost
2	Minimisation quadratique: du gradient conjugué à une méthode de type Heavy-ball adaptative avec des pas de Polyak (2022d)	En processus d'évaluation	OPT22	
3	Super-accélération avec des pas cycliques (2022b)	AISTATS22	MLOpt	Cyclical Step-sizes (2022)
4	PEPit: analyses assistées par ordinateur des pires cas des méthodes d'optimisation du premier ordre en Python (2022a)	MPC	TRADEOPT22, ICCOPT22, LOL22	
5	Sur les structures de preuve fondamentales en optimisation du premier ordre (2023b)		CDC23	
6	Méthodes optimales du premier ordre pour les fonctions convexes avec une borne supérieure quadratique (2022c)	En processus d'évaluation		
7	Contre-exemples en optimisation du premier ordre: une approche constructive (2023a)	CDC23, L-CSS	FoCM23, SIAMOP23	
8	Non-accélérations prouvables de la méthode Heavy-ball (2023c)	En processus d'évaluation		

Nous avons créé l'organisation GitHub PerformanceEstimation contenant à la fois PESTO et PEPit, ainsi que des exercices introductifs aux PEPs et le site web du workshop que nous avons organisé sur ce sujet.

J'ai également eu l'occasion de collaborer, avant et pendant ma thèse, sur les articles Goujaud et al. (2017); Aljundi et al. (2019); Guille-Escuret et al. (2021, 2022); Ferbach et al. (2023) qui ne sont pas abordés dans cette thèse.

1

Introduction

Contents

In this thesis, we consider optimization problems of the generic form

$$f_\star \triangleq \min_{x\in\mathbb{R}^d} f(x), \tag{OPT}$$

where the scalar-valued function $f : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ (proper, i.e., non-identically equals to $+\infty$) is assumed to have a non-empty compact set of global minimizers denoted by $\mathcal{X}_\star$.

While optimization is an old problem (Cauchy, 1847), it has found many applications over time. Indeed, many practical problems can be reduced into the form (OPT). *Machine Learning* and *Computer Vision* are examples of fields recently receiving a lot of interest (see Krenn et al., 2022, Figure 1) and heavily relying on optimization.

Therefore, understanding existing optimization algorithms and developing new ones has become a major challenge. The theory of optimization has already been successful in the past, in particular by introducing methods such as *Gradient descent* (GD) (Cauchy, 1847) or momentum-based methods such as the *Heavy-ball* (HB) (Polyak, 1963) and *Fast Gradient* (FGM) (Nesterov, 1983) methods. Interestingly, being so efficient, those methods are even used far beyond their domain of theoretical convergence, e.g. in Deep Learning.

This introduction describes the key notions that are used in the following chapters as well as the organisations of the different parts. Part I deals with quadratic optimization while we focus on non-quadratic optimization in Part II. Those two parts use different tools. We briefly describe the two settings in Section 1.2 and Section 1.3. Finally, we summarize our contributions in Section 1.4.

1.1 Oracles and algorithms performance

Convergence properties of first-order optimization are typically analyzed through worst-case analyzes under the *black-box* model (Nemirovskii and Yudin, 1983a).

Black box optimization and oracles. The black-box model assumes the procedure does not have access to a full analytical description of the objective function, but rather only to partial descriptive discrete pieces of information queried to the so-called oracles. In words, an algorithm proposes a first point (or iterate) $x_0 \in \mathbb{R}^d$ and queries information from the oracle $\mathcal{O}$ that sends $\mathcal{O}(x_0)$ back. Based on this knowledge, the algorithm chooses the next iterate, and so on. The most classical examples of oracles are the n^{th} order oracles defined as $\mathcal{O}(x) \triangleq (f(x), \nabla f(x), \cdots, \nabla^n f(x))$, especially zeroth, first and second orders oracles.

How to choose an oracle? The higher the order of the oracle is, the more information we get. Therefore, it seems natural to consider a high order oracle when possible. Second order methods are known to converge fast, at least locally. However, hessians may be hard or costly to compute, and sometimes simply does not exist. For instance, in large-scale optimization, hessians are too large to be handled properly, manipulated (e.g. inverted) and sometimes even to be stored in memory.

On the other hand, a zeroth order oracle is usually cheap and therefore convenient for large scale problems. They are also used to optimize non-differentiable functions and functions defined on a discrete set, encountered for example in hyperparameter optimization of machine learning models (Bergstra et al., 2011).

Therefore, a trade-off must be found between the efficiency of an oracle and its cost. High dimensional differentiable models are often well adapted for first-order oracle, espe-

cially when the model benefits from a graphical structure, suitable to apply a dynamically implemented chain rule (Rumelhart et al., 1986).

Due to the practical success of first-order methods in large-scale applications such as deep learning, their analyses for minimizing such functions in the black-box model (see e.g., (Nemirovskii and Yudin, 1983a; Polyak, 1987; Nesterov, 2003)) occupy a great deal of attention and is the main focus of this thesis. Note however that, in certain situations, other oracles can be considered, such as inexact gradient oracles (see De Klerk et al. (2020)), stochastic gradient oracles (see (Robbins and Monro, 1951)), or proximal oracles (see Rockafellar (1976); Combettes and Pesquet (2011); Chambolle and Pock (2011)).

Classical first-order methods. One of the first ever introduced algorithms is the *Gradient descent* (GD) method defined as

$$x_{t+1} = x_t - \gamma \nabla f(x_t). \tag{GD}$$

GD consists of moving along the locally steepest direction. This is therefore a greedy algorithm, always moving orthogonally to the last visited level sets. Figure 1.1 shows that this procedure can be very slow in some cases. This is due to the back-and-forth moves of the iterates. More consistent updates accelerate this procedure. This is the main idea behind momentum-based methods such as the *Heavy-ball* (HB) and *Fast Gradient* (FGM) methods defined as follows:

$$x_{t+1} = x_t - \gamma \nabla f(x_t) + \beta(x_t - x_{t-1}) \tag{HB}$$

$$x_{t+1} = x_t - \gamma \nabla f\left(x_t + \beta(x_t - x_{t-1})\right) + \beta(x_t - x_{t-1}) \tag{FGM}$$

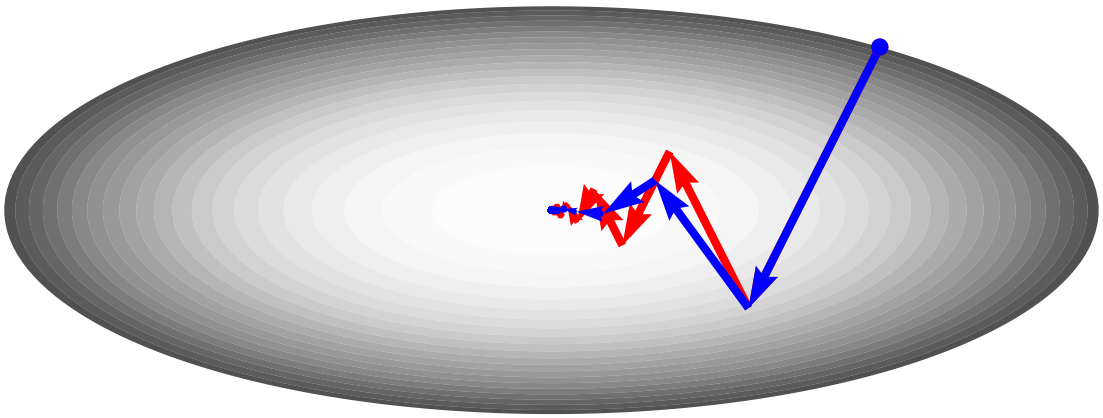


Figure 1.1: Comparison of the convergence speed of Gradient descent and Heavy-ball on a quadratic objective function. The Gradient descent method (red curve) oscillates while the Heavy-ball method (blue curve) makes more consistent moves.

Those three algorithms are the most mentioned in this thesis.

Algorithm performance and regularity. An optimization algorithm can be seen as a decision maker, that must output the next iterate, based on all the information it got from the oracle so far. In other words, after each oracle call, we obtain a *local* piece of knowledge about the objective function. Consequently, we reduce the set of possible functions that are compatible with the information gained at each step. Our ultimate goal is to approximately

find the minimizer of the objective function as fast as possible. However, without constraint on the function, local oracle knowledge is not sufficient to accurately perform this task. Indeed, one can easily convince oneself that if no assumption is made on f, whatever the values and gradients of f are on a finite set of points, its minimizers may still be anywhere else and the minimal value of f can still be any value smaller than the smallest observed one. Therefore, if we expect some result from an algorithm, we need some *regularity knowledge* about the objective function, that extends the local ones provided by the oracle up to the entire space.

Classes of functions. To make optimization possible, we assume $f \in \mathcal{F}$ for some set of functions $\mathcal{F}$. This regularity assumption must bring some global information spreading the local knowledge provided by the oracle. As an example, a classical class of functions is the class of L-gradient-Lipschitz (or L-smooth) functions, defined as

$$\{f \mid \forall x, y, \|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|\}\,.$$

If we assume that this relation holds, then any locally observed gradient contributes to control all the other unseen ones. Note that smaller L are more restrictive, that is, it corresponds to a stronger assumption, that brings more knowledge, in the sense that the bounds obtained on the gradients are tighter. The main classes we study in this thesis are

$$\mathcal{Q}_\Lambda \triangleq \{x \to \tfrac{1}{2}(x - x_\star)^T H(x - x_\star) \mid \mathrm{Sp}(H) \in \Lambda\}, \tag{1.1}$$

$$\mathcal{F}_{\mu,L} \triangleq \{f \mid \forall x, y,\ f(x) \geqslant f(y) + \langle \nabla f(y), x - y\rangle + \tfrac{\mu}{2}\|x - y\|^2 \quad \text{and} \quad f(x) \leqslant f(y) + \langle \nabla f(y), x - y\rangle + \tfrac{L}{2}\|x - y\|^2\}, \tag{1.2}$$

$$\mathrm{QG}^+(L) \triangleq \{f \mid \forall x, y,\ f(x) \leqslant f_\star + \tfrac{L}{2}\|x - x_\star\|^2\}, \tag{1.3}$$

where Λ is a subset of $\mathbb{R}_{\geqslant 0}$ and $-\infty \leqslant \mu \leqslant \max(\mu, 0) \leqslant L \leqslant +\infty$. In particular

- when $\mu > 0$, $\mathcal{F}_{\mu,L}$ is the set of L-smooth μ-strongly convex functions,
- when $\mu = 0$, $\mathcal{F}_{\mu,L} = \mathcal{F}_{0,L}$ is the set of L-smooth convex functions,
- when $\mu = -L$, $\mathcal{F}_{\mu,L} = \mathcal{F}_{-L,L}$ is the set of L-smooth functions,
- when $\mu > 0$ and $L = +\infty$, $\mathcal{F}_{\mu,L} = \mathcal{F}_{\mu,+\infty}$ is the set of μ-strongly convex functions,
- when $\mu = 0$ and $L = +\infty$, $\mathcal{F}_{\mu,L} = \mathcal{F}_{0,+\infty}$ is the set of closed convex proper functions.

Types of performance analysis. Let f be a function. Given a budget of computation, we can compare the performance of 2 optimization algorithms over this objective function f. Now, given a class of functions $\mathcal{F}$, *how can we compare two algorithms?* In some rare cases, we find an algorithm that is better than all the others on *every function of the class* $\mathcal{F}$. This is the case for some algorithms such as the one we propose in Chapter 2. However, most of the time, a given algorithm is better than another one on a specific function, and worse on another function. We therefore need to set a metric. Many types of analysis are used in optimization theory, among which the 2 following:

1. The worst-case analysis (see Nemirovskii (1992, 1994)) consists in defining the performance of an algorithm over a class as its worst performance over the functions of this class. Although the worst-case analysis is heavily used to explain the performance

of many optimization methods (see e.g., Nesterov, 1983), it can sometimes be unrepresentative of the observed complexity when the worst-case problem is never encountered in practice.

2. The average-case analysis (see Borgwardt (1977, 1980)) overcomes this issue by averaging the performance of an algorithm over all functions of the class of interest. While less studied, the average-case analysis has recently been employed in quadratic optimization (see e.g. Pedregosa and Scieur (2020); Scieur and Pedregosa (2020); Cunha et al. (2022)). However, this type of analysis depends on the probability distribution over the problems, that is arbitrarily chosen.

The worst-case analysis is the most common one, the first analysis that is usually made to tentatively explain the observed complexity, and the one we focus this thesis on.

Worst-case upper bounds. In worst-case analysis, an upper bound is therefore a guarantee that holds on every single function of the studied class. As examples, in this thesis, we prove the following:

1. Let $\mu_1 < L_1 < \mu_2 < L_2$ for positive real numbers with $L_1 - \mu_1 = L_2 - \mu_2$. We prove in Corollary 3.3.2 that a version of (HB) verifies
$$\forall f \in \mathcal{Q}_{[\mu_1,L_1]\cup[\mu_2,L_2]}, \quad d(x_t, \mathcal{X}_\star) \leqslant \Big(\frac{\sqrt{\rho^2 - R^2} - \sqrt{\rho^2 - 1}}{\sqrt{1 - R^2}}\Big)^t \Big(1 + t\sqrt{\frac{\rho^2 - 1}{\rho^2 - R^2}}\Big) d(x_0, \mathcal{X}_\star)$$
for t even, where $\rho = \frac{L_2+\mu_1}{L_2-\mu_1}$ and $R = \frac{\mu_2-L_1}{L_2-\mu_1}$.
2. We prove in Theorem 6.2.4 that a version of (HB) verifies
$$\forall f \in \mathrm{QG}^+(L) \cap \mathcal{F}_{0,+\infty}, \quad f(x_t) - f_\star \leqslant \frac{L}{2}\frac{1}{t+1} d(x_0, \mathcal{X}_\star)^2$$
for all t.

Worst-case lower bounds. On the other hand, a lower bound is effective as soon as one function of the class prevents the algorithm to perform faster than this bound. As examples, in this thesis, we prove the following:

1. Let $\mu_1 < L_1 < \mu_2 < L_2$ for positive real numbers with $L_1 - \mu_1 = L_2 - \mu_2$. We prove in Corollary 3.3.2 that for any first-order algorithm
$$\exists f \in \mathcal{Q}_{[\mu_1,L_1]\cup[\mu_2,L_2]}, \quad d(x_t, \mathcal{X}_\star) \geqslant \Big(\frac{\sqrt{\rho^2 - R^2} - \sqrt{\rho^2 - 1}}{\sqrt{1 - R^2}}\Big)^t \Big(1 + t\sqrt{\frac{\rho^2 - 1}{\rho^2 - R^2}}\Big) d(x_0, \mathcal{X}_\star)$$
for t even, where $\rho = \frac{L_2+\mu_1}{L_2-\mu_1}$ and $R = \frac{\mu_2-L_1}{L_2-\mu_1}$.
2. We prove in Theorem 6.2.4 that for any first-order algorithm
$$\exists f \in \mathrm{QG}^+(L) \cap \mathcal{F}_{0,+\infty}, \quad f(x_t) - f_\star \geqslant \frac{L}{2}\frac{1}{t+1} d(x_0, \mathcal{X}_\star)^2$$
for all t.
3. Let $0 < \mu < L < +\infty$. We prove in Theorem 8.3.6 that for any stationary version of (HB),
$$\exists f \in \mathcal{F}_{\mu,L}, \quad \|x_t - x_\star\| \geqslant \left(\frac{1 - C\kappa}{1 + C\kappa}\right)^t \|x_0 - x_\star\|$$
for all t.

Worst-case tight guarantees. A tight guarantee is both an upper bound and a lower bound. Corollary 3.3.2 and Theorem 6.2.4 therefore provide tight guarantees. Chapter 5 discusses the mathematical problem of finding the tight guarantee verified by an algorithm on a class. As an example, this problem on (GD) can be written as

$$\left| \begin{array}{ll} \underset{\substack{f\in\mathcal{F},d\geqslant 1\\ (x_t)_{t\leqslant T}\in\left(\mathbb{R}^d\right)^{T+1}}}{\text{maximize}} & f(x_T)-f_\star \\ \text{subject to} & \left\{ \begin{array}{l} d(x_0,\mathcal{X}_\star)\leqslant 1, \\ x_{t+1}=x_t-\gamma\nabla f(x_t). \end{array} \right. \end{array} \right. \tag{$\mathcal{P}$}$$

Constructive approaches. In recent years, two main approaches have been proposed to automate the search for upper, lower or tight worst-case bounds. The Integral Quadratic Constraints (IQC) formalism (see Lessard et al. (2016); Lessard (2022)) primarily focuses on searching for upper bounds, while the Performance Estimation Problems (PEP) formalism (see Drori and Teboulle (2014); Taylor et al. (2017c); Drori and Taylor (2020)) considers the problem ($\mathcal{P}$) and therefore searches for tight bounds (by solving exactly ($\mathcal{P}$)) or lower bounds (by finding a feasible point to ($\mathcal{P}$)). Both the IQC approach and the PEP approach have encountered several success in the design of some methods (see e.g. Van Scoy et al. (2017); Cyrus et al. (2018); Taylor and Drori (2022)), and have been combined in Taylor et al. (2018a). Chapter 5 describes the PEP approach to tackle this problem and provides a non-exhaustive list of insights the PEP literature contains on the proof structure of worst-case analyzes in optimization.

1.2 Quadratic first-order optimization

One particular class of interest is the class of quadratic convex functions $\mathcal{Q}$. Solving (OPT) over $\mathcal{Q}$ enables us to find the solutions to linear regressions. Moreover, any twice differentiable function can be approximated as a quadratic function around its optimizers. Therefore, studying the behavior of a method on quadratic functions provides good insights on the asymptotic behavior of this method on any twice differentiable functions. A good example of this is the Heavy-ball method, historically derived on $\mathcal{Q}$ (Polyak, 1963), it is now used on much larger classes, showing good practical results.

Formally, we can define F as $F(x)\triangleq\frac{1}{2}(x-x_\star)^TH(x-x_\star)+F_\star$ where H is the constant Hessian matrix of F. Its gradient at x is therefore $\nabla F(x)=H(x-x_\star)$. This parametrization of $\mathcal{Q}$ by the hessian matrix H helps express the classical algorithms in a simple way. For example:

1. (GD)'s update verifies $x_{t+1}-x_\star=(I-\gamma H)(x_t-x_\star)$,
2. (HB)'s update verifies $x_{t+1}-x_\star=(I-\gamma H)(x_t-x_\star)+\beta(x_t-x_{t-1})$,
3. (FGM)'s update verifies $x_{t+1}-x_\star=(I-\gamma H)(x_t-x_\star+\beta(x_t-x_{t-1}))$.

Then, ($\mathcal{P}$) writes as

$$\left|\begin{array}{ll} \underset{\substack{d\geqslant 1, H\in\mathcal{S}_d^+ \\ (x_t)_{t\leqslant T}\in\left(\mathbb{R}^d\right)^{T+1}}}{\text{maximize}} & \frac{1}{2}(x_T-x_\star)^T H(x_T-x_\star) \\ \text{subject to} & \begin{cases} d(x_0,\mathcal{X}_\star)\leqslant 1, \\ x_{t+1}-x_\star=(I-\gamma H)(x_t-x_\star). \end{cases} \end{array}\right.$$

To study (GD)'s convergence properties, we unroll the updates as $x_t - x_\star = (I - \gamma H)^t(x_0 - x_\star)$ and reduce the study of (GD)'s convergence properties to the study of the matrix $(I-\gamma H)^t$. Polyak (1964) proceeds similarly to study momentum methods. For example, by defining $X_t \triangleq \begin{pmatrix} x_t \\ x_{t-1} \end{pmatrix}$, (HB) verifies $X_{t+1} = MX_t$ with $M \triangleq \begin{pmatrix} (1+\beta)I - \gamma H & -\beta I \\ I & 0 \end{pmatrix}$. The study of (HB) reduces to the study of the matrix M, allowing to establish the convergence (see Figure 8.1 in Chapter 8).

More generally, any method whose update is of the very generic form $x_t = x_0 - \sum_{i=0}^{t-1} \gamma_i^{(t)} \nabla f(x_i)$, as suggested in (Nemirovskii, 1992, 1994), can also be explicitly written as $\forall t \geqslant 0, x_t - x_\star = P_t(H)(x_0 - x_\star)$ for some polynomial P_t. This strong connection between quadratic optimization and polynomials theory, first described in Fischer (2011), is detailed and used in Chapters 2 and 3.

Organisation of Part I on quadratic optimization. In Chapter 2, we use polynomials theory to derive an alternative to the conjugate gradient method under the form of a Heavy-ball update using the Polyak step-size. Then, in Chapter 3, still using polynomials theory, we show that the Heavy-ball method with cycling step-size is worst-case optimal on the class of quadratic functions in which eigenvalues lie in the union of 2 intervals. This method is actively used in practice and has been proven efficient in complex problems. Chapter 3 brings the first analysis of this method. Finally, while Chapter 8 focuses on the behavior of (HB) on a larger class of functions, results on quadratic are recalled and used to prove that the known convergence behavior of (HB) is specific to the class of quadratic functions.

1.3 Beyond quadratic optimization

Unlike the specific case of $\mathcal{Q}$, classes of functions typically do not inherently possess a parametric representation. Instead, they are defined as sets of functions verifying some inequalities. Typical examples of such restrictions, that have been mentioned above, are:

- convexity: (see e.g., Rockafellar (1997, §4), Nesterov (2003, Definition 2.1.1)) $\forall x, y \in \mathbb{R}^d, \quad f(y) \geqslant f(x) + \langle \nabla f(x), y - x \rangle$,
- strong convexity: (see e.g., Nesterov (2003, Definition 2.1.2)) $\forall x, y \in \mathbb{R}^d, \quad f(y) \geqslant f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2$,
- smoothness: (see e.g., Nesterov (2003, eq. 1.2.3)) $\forall x, y \in \mathbb{R}^d, \quad \|\nabla f(x) - \nabla f(y)\| \leqslant L\|x - y\|$,
- Lipschitz continuity: (see e.g., Rockafellar (1997, §10)) $\forall x, y \in \mathbb{R}^d, \quad |f(x) - f(y)| \leqslant L\|x - y\|$.

Combining some inequalities can lead to stronger ones. As an example, a function that is both convex and L-smooth verifies (see Nesterov, 2003, Theorem 2.1.5)

$$\forall x, y \in \mathbb{R}^d, \quad f(y) \geqslant f(x) + \langle \nabla f(x), y - x \rangle + \tfrac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2. \tag{1.4}$$

Given a class of functions and an algorithm, we can derive a worst-case guarantee of the algorithm over the class of functions. For example, (GD) over L-smooth convex functions verifies

$$\begin{aligned}\|x_{t+1} - x_\star\|^2 =& \|x_t - x_\star\|^2 - \tfrac{2}{L} \underbrace{\langle \nabla f(x_t), x_t - x_\star \rangle}_{\geqslant f(x_t) - f_\star + \frac{1}{2L}\|\nabla f(x_t)\|^2} + \tfrac{1}{L^2}\|\nabla f(x_t)\|^2 \\ \leqslant& \|x_t - x_\star\|^2 - \tfrac{2}{L}(f(x_t) - f_\star).\end{aligned}$$

Summing over t from 0 to some T, the telescopic summation gives

$$\|x_{T+1} - x_\star\|^2 \leqslant \|x_0 - x_\star\|^2 - \tfrac{2}{L}\sum_{t=0}^{T}(f(x_t) - f_\star),$$

proving in particular that $\min_{t \in [\![0,T]\!]} f(x_t) - f_\star \leqslant \frac{L}{2}\frac{\|x_0 - x_\star\|^2}{T+1}$.

This short proof raises quite a few questions: Is the obtained guarantee optimal? Or can we find a tighter one for the same algorithm on the same class? In particular, we used (1.4) instead of the classical smoothness and strong convexity inequalities. Would another proof work without it? Which inequalities should we consider in more general cases? How should we arrange them in a complex proof?

The answers to those questions resides in the rewriting of ($\mathcal{P}$) in a tractable form. This line of work, commonly referred to as *Performance estimation* and initiated by Drori and Teboulle (2014); Taylor et al. (2017c), has been used over the past years to derive optimization certificates and has been developed for further applications such as finding Lyapunov functions (Taylor et al., 2018a) or designing new algorithms (Drori and Taylor, 2020). A gentle introduction to this framework can be found in (Taylor, 2020).

Organisation of Part II beyond quadratic optimization. In Chapter 4, we describe a Python package we developed to largely ease the use of this framework and automate the search for numerical certificates and their proofs. In Chapter 5, we explain the general structure of those proofs. In Chapter 6, we use both the Python package and theoretical insights described in Chapter 5 to derive a complete theory of first-order optimization over a large class of functions. Finally, in Chapters 7 and 8, we complement the *Performance estimation* usual use-case by extending it to the search of cycles in the sequence of iterates produced by an algorithm, disproving its convergence. Moreover, we apply it to some instances of (HB), proving its non-acceleration over $\mathcal{F}_{\mu,L}$.

1.4 Contributions

In Part I, we focus on quadratic optimization. More precisely, we use the equivalence mentioned above between quadratic optimization and polynomial theory as proposed in Fischer (2011) to tackle two problems:

In Chapter 2, we design a Heavy-ball-like algorithm, tuned with Polyak step-sizes, that is instance-wise optimal on $\mathcal{Q}_{0,+\infty}$, that is it minimizes $\|x_t - x_\star\|$ for all t and every quadratic convex function. Interestingly, Polyak step-size is well-known as optimal 1-step tuning for (GD) on quadratic convex functions, and (HB) is known to accelerate over (GD) on quadratic convex functions. We show that the optimal strategy to adopt on $\mathcal{Q}_{0,+\infty}$ consists in combining the two approaches with the right momentum.

In Chapter 3, we make a theoretical bridge between two empirical observations. On the one hand, (Loshchilov and Hutter, 2017; Smith, 2017) showed state-of-the-art results of a cycling step-sizes strategy of (HB) on different deep learning benchmarks. On the other hand, (Sagun et al., 2017; Papyan, 2018; Ghorbani et al., 2019; Papyan, 2019) have empirically studied properties of the Hessians of deep neural networks showing their spectrum most often lie in the union of two intervals. Based on this last observation, we provide a necessary and sufficient condition that an algorithm must verify to be worst-case optimal on $\mathcal{Q}_\Lambda$, and in particular, exhibit that the worst-case optimal algorithm when Λ is a union of 2 intervals of the same length is (HB) with a cycling step-size strategy.

In Part II, we focus on optimization on large classes of functions, beyond the quadratic case. Our work is mainly based on a theoretical framework proposed in Drori and Teboulle (2014); Drori (2014); Taylor et al. (2017c,a) to study optimization over different classes. We ease its usage by providing a Python package and a tutorial summarizing a few of the essential strengths and insights provided by the frameworks. We also build upon this framework to develop non-convergence guarantees.

In Chapter 4, we describe the python package PEPit that we implemented. This package makes the use of the PEP framework very user-friendly. It can be seen as an extended open source PYTHON version of the MATLAB package PESTO (Taylor et al., 2017b).

More than simply automating the search for numerical proofs of optimization guarantees, the PEP framework brings a lot of insights about the structure of those proofs. In Chapter 5, we review the classical PEP derivation (Drori and Teboulle, 2014; Drori, 2014; Taylor et al., 2017c,a) and detail the consequences we learn on proof structure from the PEP approach, such as the minimal set of inequalities to be used and the right way to combine them. We also review techniques to find Lyapunov functions (Lessard et al., 2016; Taylor et al., 2018a). Finally, this framework has been used and extended to design new worst-case optimal algorithms (see Drori and Teboulle, 2014; Kim and Fessler, 2016, 2021; Sundararajan et al., 2020; Park and Ryu, 2022; Das Gupta et al., 2023; Jang et al., 2023; Barré et al., 2023). We review the approach proposed in (Drori and Taylor, 2020), inspired by Nemirovskii works (Nemirovskii, 1982; Nemirovskii and Yudin, 1983b) to originally design accelerated methods. Unfortunately, those works are difficult to find, so we refer to the review (Narkiss and Zibulevsky, 2005). Interestingly, this approach is compatible with the search for Lyapunov functions.

In Chapter 6, we study a particular class of non-smooth functions: $\mathrm{QG}^+(L) \cap \mathcal{F}_{0,+\infty}$. In this chapter, we derive tight worst-case guarantees of variants of (GD) and (HB) as well as the first-order lower bound of this class, showing that (HB) is worst-case optimal for this class of functions. We also describe a line-search parameter-free version of this algorithm achieving the same guarantee. We also extend this result to other classes of non-smooth functions.

In Chapter 7, we describe a PEP-based approach to automate the search for cycles of first-order stationary methods. This approach can enable to prove the non-convergence of

an algorithm and is therefore complementary to the classical PEP approach.

In Chapter 8, we improve the previous approach to simplify it, and draw conclusions about the generic form of the cycles. In particular, we prove the non-acceleration of (HB) on $\mathcal{F}_{\mu,L}$, a very long-term open problem.

Main algorithms appearances.

- (GD) is studied in Chapters 4, 6 and 7,
- (HB) is studied in Chapters 2, 3 and 6 to 8,
- (FGM) and the *Three operator splitting*(TOS) method are studied in Chapter 7.

Class of functions appearances.

- $\mathcal{Q}_\Lambda$ is studied in Chapters 2, 3 and 8,
- $\mathcal{F}_{\mu,L}$ is studied in Chapters 4, 7 and 8,
- $\mathrm{QG}^+(L)$ is studied in Chapter 6.

Publications. This thesis is based on the following papers:

Chapter	Paper	Conference / Journal	Workshop / Invited talks	Blogpost
2	Quadratic minimization: from conjugate gradient to an adaptive Heavy-ball method with Polyak step-sizes (2022d)	Under review	OPT22	
3	Super-acceleration with cyclical step-sizes (2022b)	AISTATS22	MLOpt	Cyclical Step-sizes (2022)
4	PEPit: computer-assisted worst-case analyses of first-order optimization methods in Python (2022a)	MPC	TRADEOPT22, ICCOPT22, LOL22	
5	On Fundamental Proof Structures in First-Order Optimization (2023b)		CDC23	
6	Optimal first-order methods for convex functions with a quadratic upper bound (2022c)	Under review		
7	Counter-examples in first-order optimization: a constructive approach (2023a)	CDC23, L-CSS	FoCM23, SIAMOP23	
8	Provable non-accelerations of the Heavy-ball method (2023c)	Under review		

We created the GitHub organization PerformanceEstimation containing both PESTO and PEPit, but also introductory exercises to PEPs and the website of the workshop we organized around this topic.

I have also had the opportunity to collaborate, both prior to and during my PhD, on the papers Goujaud et al. (2017); Aljundi et al. (2019); Guille-Escuret et al. (2021, 2022); Ferbach et al. (2023) which are not discussed in this thesis.

Part I

Tools for unconstrained quadratic optimization

2

Quadratic minimization: from conjugate gradient to an adaptive Heavy-ball method with Polyak step-sizes

In this work, we propose an adaptive variation on the classical Heavy-ball method for convex quadratic minimization. The adaptivity crucially relies on so-called "Polyak step-sizes", which consists of using the knowledge of the optimal value of the optimization problem at hand instead of problem parameters such as a few eigenvalues of the Hessian of the problem. This method happens to also be equivalent to a variation of the classical conjugate gradient method, and thereby inherits many of its attractive features, including its finite-time convergence, instance optimality, and its worst-case convergence rates.

The classical gradient method with Polyak step-sizes is known to behave very well in situations in which it can be used, and the question of whether incorporating momentum in this method is possible and can improve the method itself appeared to be open. We provide a definitive answer to this question for minimizing convex quadratic functions, an arguably necessary first step for developing such methods in more general setups.

This chapter is based on our work "Quadratic minimization: from conjugate gradient to an adaptive Heavy-ball method with Polyak step-sizes" (co-authored with A. Taylor, and A. Dieuleveut), currently under review.

Contents

2.1 Introduction

Consider the convex quadratic minimization problem in the form

$$\min_{x\in\mathbb{R}^d}\left\{f(x)\triangleq\frac{1}{2}\langle x,\,Hx\rangle+\langle h,\,x\rangle\triangleq\frac{1}{2}\langle x-x_\star,\,H(x-x_\star)\rangle+f_\star\right\}\tag{2.1}$$

where $H\succcurlyeq 0$ is a symmetric positive semi-definite matrix, and we denote $f_\star$ the minimum value of f. In the context of large-scale optimization (i.e. $d\gg 1$), we are often interested in using first-order iterative methods for solving equation (2.1). There are many known and celebrated iterative methods for solving such quadratic problems, including conjugate gradient, Heavy-ball methods (a.k.a., Polyak momentum), Chebyshev methods, and Gradient descent. Each of those methods having different specifications, the choice of the method largely depends on the application at hand. In particular, a typical drawback of momentum-based methods is that they generally require the knowledge of some problem parameters (such as extreme values of the spectrum of H). This problem is typically not as critical for simpler Gradient descent schemes with no momentum, although it generally still requires some knowledge on problem parameters if we want to avoid using linesearch-based strategies. This limitation motivates the search for adaptive strategies, fixing step-size using past observations about the problem at hand. In the context of (sub)gradient descent, a famous adaptive strategy is the so-called Polyak step-size, which relies on the knowledge of the optimal value $f_\star$:

$$x_{t+1}=x_t-\gamma_t\nabla f(x_t),\quad\text{with }\gamma_t=\frac{f(x_t)-f_\star}{\|\nabla f(x_t)\|^2}.\tag{2.2}$$

Polyak steps were originally proposed in Polyak (1987) for nonsmooth convex minimization; it is also discussed in Boyd et al. (2003) and a few variants are proposed by, e.g., Barré et al. (2020); Loizou et al. (2021); Gower et al. (2022) including for stochastic minimization. In terms of speed, this strategy (and variants) enjoy similar theoretical convergence properties as those for Gradient descent. This method appears to perform quite well in applications where $f_\star$ can be efficiently estimated—see, e.g., Hazan and Kakade (2019) for an adaptation of the method for estimating it online. Therefore, a remaining open question in this context is whether the performances of this method can be improved by incorporating momentum in it. A first answer to this question was provided by Barré et al. (2020), although it is not clear that it can match the same convergence properties as optimal first-order methods.

In this work, we answer this question for the class of quadratic problems. In short, it turns out that the following conjugate gradient-like iterative procedure

$$x_{t+1}=\arg\min_x\left\{\|x-x_\star\|^2\text{ s.t. }x\in x_0+\operatorname{span}\{\nabla f(x_0),\nabla f(x_1),\dots,\nabla f(x_t)\}\right\},\tag{2.3}$$

can be rewritten exactly as a Heavy-ball type method whose parameters are chosen adaptively using the value of $f_\star$. This might come as a surprise, as the iteration equation (2.3) might seem impractical due to its formulation relying on the knowledge of $x_\star$. More precisely, equation (2.3) is exactly equivalent to:

$$x_{t+1}=x_t-(1+m_t)h_t\nabla f(x_t)+m_t(x_t-x_{t-1}),\tag{2.4}$$

with parameters

$$\forall t \geqslant 0, \quad h_t \triangleq \frac{2(f(x_t) - f_\star)}{\|\nabla f(x_t)\|^2}, \tag{2.5}$$
$$m_0 \triangleq 0 \text{ and } \forall t \geqslant 1, \quad m_t \triangleq \frac{-(f(x_t) - f_\star)\langle \nabla f(x_t), \nabla f(x_{t-1})\rangle}{(f(x_{t-1}) - f_\star)\|\nabla f(x_t)\|^2 + (f(x_t) - f_\star)\langle \nabla f(x_t), \nabla f(x_{t-1})\rangle}.$$

In equation (2.4), m_t corresponds to the momentum coefficient and h_t to a step-size. With the tuning of equation (2.5), this step-size is twice the Polyak step-size in equation (2.2). This Heavy-ball momentum method with Polyak step-sizes is summarized in Algorithm 1 and illustrated in Figure 2.1. Due to its equivalence with equation (2.3), the Heavy-ball method equation (2.4) inherits nice advantageous properties of conjugate gradient-type methods, including:

(i) finite-time convergence: the problem equation (2.1) is solved exactly after at most d iterations,

(ii) instance optimality: for *all* $H \succcurlyeq 0$, no first-order method satisfying $x_{t+1} \in x_0 + \text{span}\{\nabla f(x_0), \dots, \nabla f(x_t)\}$ results in a smaller $\|x_t - x_\star\|$,

(iii) it inherits optimal worst-case convergence rates on quadratic functions.

Of course, a few of those points needs to be nuanced in practice due to finite precision arithmetic. The equivalence between equation (2.3) and equation (2.4) is formally stated in the following theorem.

Theorem 2.1.1. *Let $(x_t)_{t\in\mathbb{N}}$ be the sequence defined by the recursion equation (2.3), namely such that for any t, x_{t+1} is the Euclidean projection of $x_\star$ onto the affine subspace $x_0 + \text{span}\{\nabla f(x_0), \nabla f(x_1), \dots, \nabla f(x_t)\}$. Then $(x_t)_{t\in\mathbb{N}}$ is the sequence generated by Algorithm 1.*

Algorithm 1 Adaptive Heavy-ball algorithm

Input T and $f : x \mapsto f(x) \triangleq \frac{1}{2}\langle x - x_\star, H(x - x_\star)\rangle + f_\star$
Initialize $x_0 \in \mathbb{R}^d$, $m_0 = 0$
for $t = 0 \cdots T - 1$ **do**
  $h_t = \frac{2(f(x_t)-f_\star)}{\|\nabla f(x_t)\|^2}$
  $x_{t+1} = x_t - (1 + m_t)h_t \nabla f(x_t) + m_t(x_t - x_{t-1})$
  $m_{t+1} = \frac{-(f(x_{t+1})-f_\star)\langle \nabla f(x_{t+1}), \nabla f(x_t)\rangle}{(f(x_t)-f_\star)\|\nabla f(x_{t+1})\|^2 + (f(x_{t+1})-f_\star)\langle \nabla f(x_{t+1}), \nabla f(x_t)\rangle}$
end
Result: x_T

Theorem 2.1.1 turns out to be a particular case of a more general result stating that the iterates of any conjugate gradient-type method described with a polynomial Q as

$$x_{t+1} = \text{argmin}_x \left\{\langle x - x_\star, Q(H)(x - x_\star)\rangle \text{ s.t. } x \in x_0 + \text{span}\{\nabla f(x_0), \dots, \nabla f(x_t)\}\right\}, \tag{Q-minimization}$$

are equivalently written in terms of an adaptive Heavy-ball iteration. In particular, equation (2.3) corresponds to equation (Q-minimization) with $Q(x) = 1$. Similarly, classical conjugate gradient method corresponds to equation (Q-minimization) with $Q(x) = x$ (this fact is quite famous, see, e.g., Nocedal and Wright (1999)). In Section 2.2, we provide the adaptive Heavy-ball iteration equivalent to (Q-minimization). The key point of this work is that the equivalent Heavy-ball reformulation of equation (2.3) can be written in terms of $f_\star$, thereby obtaining a momentum-based Polyak step-size.

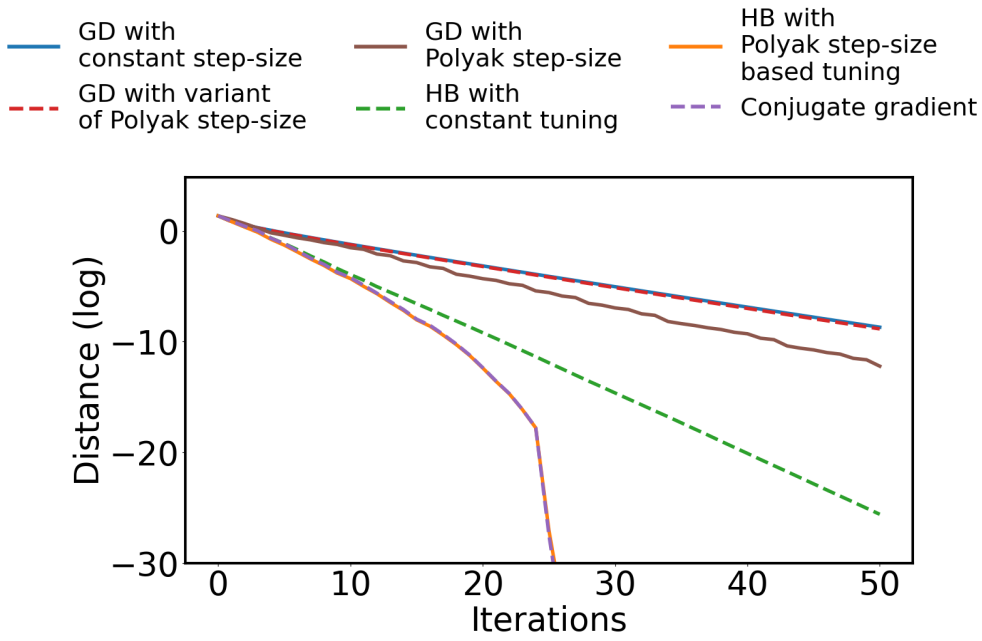


Figure 2.1: Comparison in semi-log scale over 50 iterations of different first-order methods applied on a 25-dimensional quadratic objective with condition number 10. **GD with constant step-size**, **GD with Polyak step-size** and **GD with variant of Polyak step-size** refer to the GD method tuned respectively with the step-size $\gamma = 2/(L+\mu)$, $\gamma_t = (f(x_t) - f_\star)/\|\nabla f(x_t)\|^2$ and $\gamma_t = 2(f(x_t) - f_\star)/\|\nabla f(x_t)\|^2$. **HB with constant tuning** is the HB method tuned with constant parameters $\gamma_t = (2/(\sqrt{L}+\sqrt{\mu}))^2$ and $m_t = ((\sqrt{L}-\sqrt{\mu})(\sqrt{L}+\sqrt{\mu}))^2$ while **HB with Polyak step-size based tuning** refers to Algorithm 1.

Notations. We denote $\preccurlyeq$ the order between symmetric matrices; $\mathrm{Sp}H$ the spectrum of the matrix H, namely its set of eigenvalues; $\mathbb{R}_d[X]$ the set of polynomials with degree at most d.

2.1.1 Preliminary material

Worst-case optimality. Solving equation (2.1) is a very important problem and several methods have been proposed to achieve this goal. They are compared with each other through notions of performance. This consists of evaluating the precision of an algorithm over the functions of a given class after a given number T of iterations. The main framework is *worst-case analysis* and the precision is the value of a given metric, e.g. the distance of the last iterate to the optimizer $\|x_T - x_\star\|$, the function value of the last iterate $f(x_T) - f(x_\star)$, or its gradient norm $\|\nabla f(x_T)\|$. The *worst-case analysis* framework consists of finding the guarantees of a method that hold for each and every function of a given class, as for instance the class of L-smooth μ-strongly convex quadratic functions described as quadratic functions verifying $\mu I \preccurlyeq H \preccurlyeq LI$ for given $0 < \mu \leqslant L$. The **Gradient descent (GD)** method characterized by the update

$$x_{t+1} = x_t - \gamma_t \nabla f(x_t) \tag{2.6}$$

therefore verifies $\|x_t - x_\star\| = O((\frac{L-\mu}{L+\mu})^t)$ on all such functions for $\gamma_t = \frac{2}{L+\mu}$. Thanks to a relationship with polynomial analysis, Golub and Varga (1961) proved that the **Chebyshev**

method, described as

$$x_{t+1} = x_t - \gamma_t \nabla f(x_t) + m_t(x_t - x_{t-1}), \tag{2.7}$$

for a well chosen tuning of the parameters γ_t and m_t ($m_t = (\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}})^2 \frac{1+((\sqrt{L}-\sqrt{\mu})/(\sqrt{L}+\sqrt{\mu}))^{2(t-1)}}{1+((\sqrt{L}-\sqrt{\mu})/(\sqrt{L}+\sqrt{\mu}))^{2(t+1)}}$, $\gamma_t = \frac{2}{L+\mu}(1+m_t)$), is *worst-case optimal* on this class of function, achieving the guarantee $\|x_t - x_\star\| = O((\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}})^t)$ (often referred to as "acceleration"). Methods based on this two-term recursion are called "Heavy-ball" or "Polyak momentum" (Polyak, 1964). In particular, the stationary regime of the Chebyshev method is the **Heavy-ball (HB)** method tuned with $m_t = (\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}})^2$ and $\gamma_t = \frac{2}{L+\mu}(1+m_t) = (\frac{2}{\sqrt{L}+\sqrt{\mu}})^2$ and achieves the worst-case guarantee $\|x_t - x_\star\| = O(t(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}})^t)$, close to the optimal one achieved by the Chebyshev method. Note that equation (2.7) is another formulation of equation (2.4) where $\gamma_t = (1+m_t)h_t$. In all the aforementioned tuning, h_t has the same value: $h_t = \frac{2}{L+\mu}$ (see Section 2.4 for more detailed discussion on this).

Span of gradients and Krylov subspaces. All methods described above can be defined using a recursion:

$$x_t = x_0 - \sum_{i=0}^{t-1} \gamma_i^{(t)} \nabla f(x_i) \tag{2.8}$$

for some sequence $(\gamma_i^{(t)})_{i\in[\![0,t-1]\!]}$ as suggested in (Nemirovskii, 1992, 1994). Note that the recursion equation (2.8) can also be explicitly written as $x_t = x_0 - H\sum_{i=0}^{t-1}\gamma_i^{(t)} x_i$, and therefore, $x_t - x_0 \in H\mathrm{span}(\{x_i\}_{i\in[\![0,t-1]\!]})$. We deduce by recursion that $x_t - x_0 \in H\mathcal{K}_t(H, x_0)$ where $\mathcal{K}_t(H, x_0) \triangleq \mathrm{span}(\{H^i x_0\}_{i\in[\![0,t-1]\!]})$ is called *order-t Krylov subspace generated by H and x_0*. This creates a link between first-order algorithms and polynomials, summarized in the following lemma (which is implicitly used in Golub and Varga (1961) and formally stated, e.g., in (Goujaud et al., 2022b, Proposition 4.1)).

Lemma 2.1.2. *Let f be quadratic convex (2.1). The iterates x_t satisfy*

$$x_t \in x_0 + \mathrm{span}\{\nabla f(x_0), \dots, \nabla f(x_{t-1})\}, \tag{2.9}$$

where x_0 is the initial approximation of $x_\star$, if and only if there exists a sequence of polynomials $(P_t)_{t\in\mathbb{N}}$, each of degree at most 1 more than the highest degree of all previous polynomials and P_0 of degree 0 (hence the degree of P_t is at most t), such that

$$\forall\, t, \quad x_t - x_\star = P_t(H)(x_0 - x_\star), \quad P_t(0) = 1\,. \tag{2.10}$$

Similar to the way we use this technique below, this lemma has already been extensively used to design methods; see, e.g., d'Aspremont et al. (2021, Chapter 1) or the blog post by Pedregosa (2021b) for gentle introductions to this technique. For instance, we can use this technique for optimizing the step-size of the gradient method, or to derive the Chebyshev method, which optimizes the worst-case on the class of smooth and strongly convex quadratic functions (see Fischer, 2011). More recently, Goujaud et al. (2022b) used it to derive a method which can take advantage of a possible gap in the spectrum of H. This approach has also been used for other applications such as accelerated gossip algorithms (Berthier et al., 2020).

Adaptive methods. In Lemma 2.1.2, while $P_t(H)$ is a polynomial evaluated on the matrix H, its scalar coefficients might or might not depend on H. If they depend on H, we say that the associated method is adaptive. Non-adaptive methods suffer from two main drawbacks: (i) they use the same parameters for all the functions within the class of problems, not taking advantage of the observed quantities; (ii) the underlying parameters must scale with the function class parameters, and therefore depends on the values of L and μ, which are generally difficult to estimate (and actually do not correspond to first-order information, as they rely on the Hessian of the function at hand). Ultimately, adaptive methods aim at solving those issues by choosing parameters (step-size, momentum, etc.) on the fly.

Polyak steps. It is straightforward that a Gradient descent update verifies on any convex function f that $\|x_{t+1} - x_\star\|^2 \leqslant \|x_t - x_\star\|^2 - 2\gamma_t(f - f_\star) + \gamma_t^2\|\nabla f(x_t)\|^2$. Polyak (1987) argues that, based on this inequality, the best guaranteed progression is then achieved for $\gamma_t = \frac{f(x_t)-f_\star}{\|\nabla f(x_t)\|^2}$. This choice is called "Polyak step-size" and has been studied intensively even recently Loizou et al. (2021); Gower et al. (2022).

Other variants of the latter have been proposed. For instance (Barré et al., 2020, Variant 1) suggested the step-size $\gamma_t = 2\frac{f(x_t)-f_\star}{\|\nabla f(x_t)\|^2}$. This also optimizes the exact progress of one Gradient descent update over quadratics realizing a projection of $x_t - x_\star$ over the orthogonal subspace of $\nabla f(x_t)$.

Therefore, the Polyak step-size strategy applied to the Gradient descent method achieves the same worst-case guarantee of the well tuned fixed step-size Gradient descent method, while not relying on Hessian information. Moreover, due to its adaptivity to each function, and since generic functions do not look like worst-cases, the Polyak step-size strategy applied to the Gradient descent method performs very well in practice (See Figure 2.1 and (Barré et al., 2020, Figure 1)), sometimes even beating the well tuned non-adaptive Heavy-ball method even if the worst-case guarantees are sorted in a different order.

Instance-optimality. While optimal worst-case method of the form of equation (2.8) have been found with predetermined parameters, it would be better to find a method under the form of equation (2.8) that is optimal (for some performance metric), not only on worst-case analysis, but on each function individually, taking advantage of the adaptivity of the parameters. The well-known conjugate gradient method achieves this goal when the performance metric is the function value of the last iterate. The MinRes method attacks the problem minimizing the gradient norm of the last iterate.

Contributions. In this work we derive iterative methods of the form of equation (2.8) (which iterates lie in the span of previously observed gradients) that are instance-optimal for a variety of performance metrics. All those methods updates are variations of the Heavy-ball two-term recursion equation (2.7) with only parameters γ_t and m_t changing from one method to another. Finally, we show (see Theorem 2.1.1) that for a well-chosen yet classical performance metric, this associated method Algorithm 1 is not relying on second-order information at all (not even L and μ). Instead, it uses a classical variant of the Polyak step-size $\frac{2(f(x_t)-f_\star)}{\|\nabla f(x_t)\|^2}$, providing an answer to the question "Can we accelerate methods with Polyak step-size?".

2.1.2 Related works

Polyak step-sizes were proposed in (Polyak, 1987). Despite the dependency on $f_\star$, the Polyak step-size is more studied theoretically and used in practice due to its efficiency when applied to real-world problems. Recent works (e.g. Loizou et al., 2021; D'Orazio et al., 2021) argue that this dependency is not a practical issue for many problems which we can assume verify $f_\star = 0$ (see Section 2.3). A few variants of the Polyak step-size strategy were proposed by Barré et al. (2020), including a version incorporating a Nesterov-type momentum Nesterov (1983), achieving a worst-case guarantee of $\|x_t - x_\star\|^2 = O((1 - 2(\mu/L)^{2/3})^{2t})$ over the class of (non-necessarily quadratic) L-smooth μ-strongly convex functions, thereby improving over previous works on adaptive first-order methods. However, the proposed method does not allow to remove the dependency on L and does not achieve the black-box complexity of smooth strongly convex minimization Nesterov (2003). In (Loizou et al., 2021), the authors study the stochastic Polyak step-size, whereas (D'Orazio et al., 2021) applies it to Mirror descent.

Many alternative adaptive methods have been proposed in the past. Among them, let us mention (Barzilai and Borwein, 1988) which introduced the so-called Barzilai-Borwein step-size rule, and the more recent (Malitsky and Mishchenko, 2020) which developed a step-size policy that adapts to the local geometry with convergence guarantees beyond quadratic minimization.

2.2 Main theorem

This section states and proves Theorem 2.1.1. In short, given a certain function f (characterized by H and $x_\star$ here) and a starting point x_0, we search for an iterative procedure, possibly adaptive, verifying the polynomial-based expression equation (2.10) such that x_t converges as fast as possible to $x_\star$ for some predefined performance metric. Most classical ways to measure the performance of such optimization schemes include the distance to optimum $\|x_t - x_\star\|^2$, the function accuracy gap $f(x_t) - f_\star$, the squared gradient norm $\|\nabla f(x_t)\|^2$, and linear combinations of the former. Let us abstract those notions by denoting the performance measure of choice by $\langle x_t - x_\star, Q(H)(x_t - x_\star)\rangle$ (with Q a predefined polynomial that is positive on $\mathbb{R}_{>0}$). Then, we consider the iterative scheme given by (Q-minimization).

$$x_{t+1} = \operatorname{argmin}_x \left\{ \langle x - x_\star, Q(H)(x - x_\star)\rangle \text{ s.t. } x \in x_0 + \operatorname{span}\{\nabla f(x_0), \dots, \nabla f(x_t)\} \right\}, \tag{Q-minimization}$$

The next theorem provides an explicit instance-optimal method to solve equation (Q-minimization).

Theorem 2.2.1 (Main)**.** *The unique solution to equation* (Q-minimization) *is given by the Heavy-ball procedure*

$$x_{t+1} = x_t - (1 + m_t)h_t \nabla f(x_t) + m_t(x_t - x_{t-1}) \tag{2.11}$$

where

$$\begin{cases} h_t &= \frac{\langle x_t - x_\star, HQ(H)(x_t - x_\star)\rangle}{\langle x_t - x_\star, H^2Q(H)(x_t - x_\star)\rangle}; \\ m_t &= \frac{-b_t h_t}{1 + b_t h_t}, \quad \text{with } b_t = \frac{\langle x_t - x_\star, H^2Q(H)(x_{t-1} - x_\star)\rangle}{\langle x_{t-1} - x_\star, HQ(H)(x_{t-1} - x_\star)\rangle}. \end{cases} \tag{2.12}$$

Remark that setting $Q(X) = X$ leads to a nice expression of the conjugate gradient method (see Polyak, 1987, Section 3.2.2). Indeed, setting $Q(X) = X$ corresponds to optimally minimizing the excess loss $f(x_t) - f_\star$.

As already known, the conjugate gradient method requires the knowledge of H (or a Hessian vector product) to proceed. This is also a priori the case for all other choices of $Q(\cdot)$. In the case of $Q(X) = 1$, which corresponds to minimizing the distance to the optimum (see equation (2.3)), we can use an alternate writing making use of $f_\star$:

$$\begin{cases} h_t &= \frac{2(f(x_t)-f_\star)}{\|\nabla f(x_t)\|^2} \\ m_t &= \frac{-b_t h_t}{1+b_t h_t}, \quad \text{with } b_t = \frac{\langle \nabla f(x_t), \nabla f(x_{t-1})\rangle}{2(f(x_{t-1})-f_\star)}. \end{cases} \tag{2.13}$$

Proof.

Designing methods from the polynomial point of view. As suggested by Lemma 2.1.2, we look for an iterative method that can be expressed in the form $x_t - x_\star = P_t(H)(x_0 - x_\star)$, where P_t is a t^{th} degree polynomial with $P_t(0) = 1$. Furthermore, as we look for an instance-optimal method, the latter polynomial must be instance-specific, and the coefficients of P_t should depend on H (and should describe the iterative procedure (Q-minimization)).

Recalling that H is real symmetric matrix, we denote by $\lambda \in \mathrm{Sp}(H)$ its eigenvalues and by v_λ the associated orthonormal basis of eigenvectors, leading to $H = \sum_{\lambda\in\mathrm{Sp}(H)} \lambda v_\lambda v_\lambda^T$. The quantity to be minimized can now be written as:

$$\begin{aligned} \langle x_t - x_\star,\, Q(H)(x_t - x_\star)\rangle &= \langle x_0 - x_\star,\, P_t(H)^T Q(H) P_t(H)(x_0 - x_\star)\rangle && (2.14)\\ &= \sum_{\lambda\in\mathrm{Sp}(H)} Q(\lambda) P_t(\lambda)^2 \langle x_0 - x_\star, v_\lambda\rangle^2 && (2.15)\\ &= \int_{\lambda\in\mathbb{R}^+} P_t(\lambda)^2 \,\mathrm{d}\lambda_Q(\lambda) && (2.16) \end{aligned}$$

with λ_Q the discrete measure $\sum_{\lambda\in\mathrm{Sp}(H)} Q(\lambda)\langle x_0 - x_\star, v_\lambda\rangle^2\, \delta_\lambda$ (we sometimes use the shorthand notation $\int P_t^2 \,\mathrm{d}\lambda_Q$ for equation (2.16) in what follows). It is clear that equation (2.16) is 0 if and only if $P_t(\lambda) = 0$ for all $\lambda \in \mathrm{Sp}(H)$. As a consequence, we conclude that (i) choosing the right sequence of polynomials leads to convergence in exactly $|\mathrm{Sp}(H)|$ iterations, and (ii) $\langle P^{(1)}, P^{(2)}\rangle_Q \triangleq \int P^{(1)} P^{(2)} \,\mathrm{d}\lambda_Q$ is an inner product on $\mathbb{R}_{|\mathrm{Sp}(H)|-1}[X]$. We therefore want to solve

$$\begin{cases} \underset{P_t\in\mathbb{R}_t[X]}{\text{minimize}} & \|P_t\|_Q^2 \\ \text{subject to} & P_t(0) = 1 \end{cases} \tag{2.17}$$

for any $t \leqslant |\mathrm{Sp}(H)| - 1$ where $\|P\|_Q^2 \triangleq \langle P, P\rangle_Q = \int P^2 \,\mathrm{d}\lambda_Q$ denotes the underlying norm of the inner product $\langle\cdot,\cdot\rangle_Q$. For $t \geqslant |\mathrm{Sp}(H)|$, we consider instead P_t as a multiple of the polynomial $\prod_{\lambda\in\mathrm{Sp}(H)}(X - \lambda)$ in X. The next steps are somewhat standard and follow a classical pattern for solving equation (2.17) (see, e.g. Berthier et al. (2020) and the references therein).

From minimal norm to orthogonality. The solution to equation (2.17) is the projection of the polynomial 0 over the affine space $\{P \in \mathbb{R}_t[X] \mid P(0) = 1\}$ with respect to the inner product $\langle\cdot,\cdot\rangle_Q$. A necessary and sufficient condition for P to be the solution of problem equation (2.17) is therefore to verify $\langle 0 - P, \Delta P\rangle_Q = 0$ for any ΔP in the vectorial

subspace $\{P \in \mathbb{R}_t[X] \mid P(0) = 0\} = X\mathbb{R}_{t-1}[X]$. Hence P_t solves problem equation (2.17) iff

$$\langle P_t, XR\rangle_Q = 0, \forall R \in \mathbb{R}_{t-1}[X]. \tag{2.18}$$

Note however, that for any $(P, R) \in \mathbb{R}[X]^2$,

$$\begin{aligned}\langle P, XR\rangle_Q &= \int_{\lambda\in\mathbb{R}^+} P(\lambda)\cdot\lambda R(\lambda)\,\mathrm{d}\lambda_{\mathrm{Q}}(\lambda)\\ &= \int_{\lambda\in\mathbb{R}^+} P(\lambda)\cdot R(\lambda)\,\mathrm{d}\lambda_{\mathrm{XQ}}(\lambda)\\ &\triangleq \langle P, R\rangle_{XQ}\end{aligned}$$

with $\mathrm{d}\lambda_{\mathrm{XQ}}(\lambda) \triangleq \lambda \mathrm{d}\lambda_{\mathrm{Q}}(\lambda) = \sum_{\lambda\in\mathrm{Sp}(H)} \lambda Q(\lambda)\langle x_0 - x_\star, v_\lambda\rangle^2\,\delta_\lambda$. Using the latter inner product, the condition for P_t to be the solution to problem equation (2.17) becomes:

$$P_t \in \mathbb{R}_{t-1}[X]^{\perp_{XQ}} \tag{2.19}$$

Hence, $(P_t)_{t\in\mathbb{N}}$ is a family of orthogonal polynomials for the inner product $\langle\cdot,\cdot\rangle_{XQ}$.

From orthogonality to recursion. We now focus on finding an explicit expression for the polynomials P_t. As for all families of orthogonal polynomials, $(P_t)_{t\in\mathbb{N}}$ can be obtained through a two-term recursion of the form:

$$P_{t+1}(X) = (a_t X + b_t)P_t(X) + c_t P_{t-1}(X), \quad \text{for some } (a_t, b_t, c_t) \in \mathbb{R}^3, \tag{2.20}$$

which is easy to verify by induction. Our goal is to find a_t, b_t and c_t. First, notice that $(a_t X + b_t)P_t(X) + c_t P_{t-1}(X)$ is orthogonal to $\mathbb{R}_{t-2}[X]$ independently of the values of a_t, b_t and c_t. Those three coefficients can be found via the following three conditions: (i) $\langle P_{t+1}, P_t\rangle_{XQ} = 0$, (ii) $\langle P_{t+1}, P_{t-1}\rangle_{XQ} = 0$, and (iii) $P_{t+1}(0) = 1$.

More precisely, it is clear that $a_t \neq 0$ for P_{t+1} to be of degree $t+1$. Therefore, one can factorize by a_t. Reparametrizing equation (2.20), one can write

$$P_{t+1}(X) = \frac{(\tilde{a}_t - X)P_t(X) + \tilde{b}_t P_{t-1}(X)}{\tilde{c}_t}, \text{ with } (\tilde{a}_t, \tilde{b}_t, \tilde{c}_t) \in \mathbb{R}^3.$$

Moreover, evaluation at $X = 0$ gives $\frac{\tilde{a}_t+\tilde{b}_t}{\tilde{c}_t} = 1$, thereby enforcing $\tilde{c}_t = \tilde{a}_t + \tilde{b}_t$. It remains to verify the two orthogonality conditions (independent of $\tilde{c}_t$):

$$\begin{aligned}\tilde{a}_t\langle P_t, P_t\rangle_{XQ} \quad &+\tilde{b}_t\langle P_{t-1}, P_t\rangle_{XQ} \quad &= \langle XP_t(X), P_t(X)\rangle_{XQ},\\ \tilde{a}_t\langle P_t, P_{t-1}\rangle_{XQ} \quad &+\tilde{b}_t\langle P_{t-1}, P_{t-1}\rangle_{XQ} \quad &= \langle XP_t(X), P_{t-1}(X)\rangle_{XQ}.\end{aligned}$$

Note that this system of equations is decoupled since $\langle P_{t-1}, P_t\rangle_{XQ} = 0$, and we finally arrive to

$$P_{t+1}(X) = \frac{(\tilde{a}_t - X)P_t(X) + \tilde{b}_t P_{t-1}(X)}{\tilde{a}_t + \tilde{b}_t}, \tag{2.21}$$

with

$$\begin{cases}\tilde{a}_t &= \frac{\langle XP_t(X), P_t(X)\rangle_{XQ}}{\langle P_t, P_t\rangle_{XQ}},\\ \tilde{b}_t &= \frac{\langle XP_t(X), P_{t-1}(X)\rangle_{XQ}}{\langle P_{t-1}, P_{t-1}\rangle_{XQ}}.\end{cases} \tag{2.22}$$

From a polynomial recursion to an iterative optimization method. For reaching the final desired result, we simply multiply equation (2.21) (evaluated in H) by $x_0 - x_\star$:

$$\begin{aligned} x_{t+1} - x_\star &= \frac{\tilde{a}_t(x_t - x_\star) - H(x_t - x_\star) + \tilde{b}_t(x_{t-1} - x_\star)}{\tilde{a}_t + \tilde{b}_t}, \\ &= x_t - x_\star - \frac{1}{\tilde{a}_t + \tilde{b}_t}\nabla f(x_t) + \frac{-\tilde{b}_t}{\tilde{a}_t + \tilde{b}_t}(x_t - x_{t-1}), \end{aligned}$$

thereby reaching the desired

$$x_{t+1} = x_t - (1 + m_t)h_t\nabla f(x_t) + m_t(x_t - x_{t-1}) \tag{2.23}$$

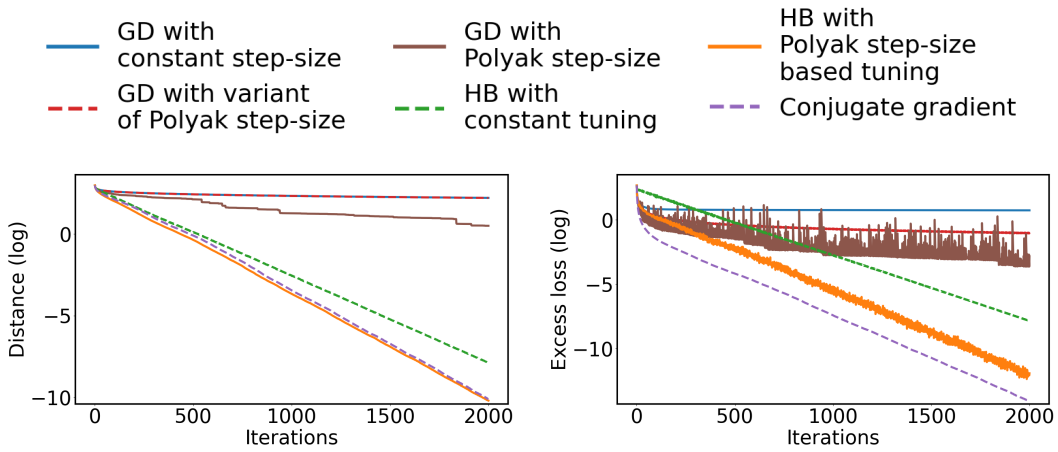
$$\text{with} \quad (1 + m_t)h_t = \frac{1}{\tilde{a}_t + \tilde{b}_t} \text{ and } m_t = \frac{-\tilde{b}_t}{\tilde{a}_t + \tilde{b}_t}, \qquad \text{hence,} \qquad h_t = \frac{1}{\tilde{a}_t} \text{ and } m_t = \frac{-\tilde{b}_t h_t}{1 + \tilde{b}_t h_t}. \tag{2.24}$$

From equations (2.23) and (2.24), we recognize a Heavy-ball method with some variable step-size h_t and momentum term m_t corresponding to the theorem statement, thereby concluding the proof. ■

Remark 2.2.2 (Step-size parametrization.)**.** *While the γ_t plays a different role in equation (2.7) and equation (2.6), they both usually are called "step-size" by default. But we noticed that both in the Chebyshev method and the Heavy-ball method (optimally tuned), $h_t = \frac{\gamma_t}{1+m_t}$ is exactly $\frac{2}{L+\mu}$, value of the optimal step-size for Gradient descent. In equation (2.5), we notice again that the value of h_t is the optimal step-size for a single step of Gradient descent. For this reason, we believe that the natural parametrization of the Heavy-ball methods should be $x_{t+1} = x_t - (1 + m_t)h_t\nabla f(x_t) + m_t(x_t - x_{t-1})$ and that h_t should be referred to as the "natural" step-size. Indeed, when one thinks of the Heavy-ball method with Polyak step-sizes, they would set γ_t to the Polyak step-size, not $h_t = \frac{\gamma_t}{1+m_t}$. We therefore provide a novel view on what should be tested.*

2.3 Numerical experiments

In this section, we compare Gradient descent, Heavy-ball, and conjugate gradient method in an adaptive setting or not. Figure 2.2 shows the performance of all these methods on a quadratic objective with known minimal value $f_\star$. The hessian of this quadratic objective has been generated from a sequence of eigenvalues with geometric increase, and a random orthogonal transformation. The difference between Figure 2.1 and Figure 2.2 is the dimension of the problem as well as the condition number of the objective function. Due to finite precision arithmetic, the finite-time convergence is not visible when the condition number is too large. However, both figures show that our method and the conjugate gradient algorithm behave similarly and faster than the other methods. The code can be found on this GitHub repository.



(a) Comparison of distances to optimum

(b) Comparison of excess losses

Figure 2.2: Comparison in semi-log scale over 2000 iterations of different first-order methods applied on a 1000-dimensional quadratic objective with condition number 10^5. **GD with constant step-size**, **GD with Polyak step-size** and **GD with variant of Polyak step-size** refer to the GD method tuned respectively with the step-size $\gamma = 2/(L+\mu)$, $\gamma_t = (f(x_t) - f_\star)/\|\nabla f(x_t)\|^2$ and $\gamma_t = 2(f(x_t) - f_\star)/\|\nabla f(x_t)\|^2$. **HB with constant tuning** is the HB method tuned with constant parameters $\gamma_t = (2/(\sqrt{L} + \sqrt{\mu}))^2$ and $m_t = ((\sqrt{L} - \sqrt{\mu})(\sqrt{L} + \sqrt{\mu}))^2$ while **HB with Polyak step-size based tuning** refers to Algorithm 1.

2.4 Concluding remarks and discussion

Polyak step-sizes are known for their general good working performances when the optimal value to the optimization problem at hand is known. Whether Polyak step-sizes can be used together with momentum for obtaining accelerated first-order methods appears to be an open question (Barré et al., 2020), which we answer in the simpler case of convex quadratic minimization. In this context, we argue that not only this tuning works well, but also it pops up naturally when investigating instance-optimal first-order iterative methods. Furthermore, we believe it is a necessary step for being able to understand more general optimization settings beyond quadratics. As our method does not seem to work well beyond quadratics, we leave further investigations on this topic for future work.

Among our competitors, we note that the celebrated conjugate gradient (CG) method is another instance-optimal algorithm for quadratics. Whereas our method minimizes the distance to the solution at each iteration, CG is instance-optimal for minimizing function values at each iteration. Perhaps interestingly, the two methods appeared to behave similarly in our numerical experiments. That being said, the main practical differences between the two methods are that CG Heavy-ball-like formulation naturally relies on higher order information while Polyak step-sizes do require knowledge of $f_\star$. In typical optimization problems, this value is not known. However, there are a few settings where this value is actually well-known, typically when $f_\star = 0$ generically (in machine learning, this setting is known as the "interpolation" regime; an alternative could be to use Polyak-steps as a competitor to MinRes). Finally, let us mention that a few generalizations of CG, often referred to as nonlinear conjugate gradient, were studied in the literature (see, e.g., (Bonnans et al., 2006; Nocedal and Wright, 1999; Hager and Zhang, 2006)). A compelling direction for future research would involve expanding our proposed method to a class of non-quadratic objectives.

3

Super-Acceleration with Cyclical Step-sizes

We develop a convergence-rate analysis of momentum with cyclical step-sizes. We show that under some assumption on the spectral gap of Hessians in machine learning, cyclical step-sizes are provably faster than constant step-sizes. More precisely, we develop a convergence rate analysis for quadratic objectives that provides optimal parameters and shows that cyclical learning rates can improve upon traditional lower complexity bounds. We further propose a systematic approach to design optimal first-order methods for quadratic minimization with a given spectral structure. Finally, we provide a local convergence rate analysis beyond quadratic minimization for the proposed methods and illustrate our findings through benchmarks on least squares and logistic regression problems.

This chapter is based on our work "Super-Acceleration with Cyclical Step-sizes" (co-authored with D. Scieur, A. Dieuleveut, A. Taylor, and F. Pedregosa), published at AISTATS, 2022. We also wrote a blogpost (see Goujaud and Pedregosa (2022)) for a friendly introduction to this work.

Contents

Algorithm 2
Cyclical Heavy-ball $\mathrm{HB}_K(h_0, \ldots, h_{K-1}; m)$

Input: Initialization x_0, momentum $m \in (0, 1)$, step-sizes $\{h_0, \ldots, h_{K-1}\}$
$x_1 = x_0 - \frac{h_0}{1+m}\nabla f(x_0)$
for $t = 1, 2, \ldots$ **do**

$$x_{t+1} = x_t - h_{\mathrm{mod}(t,K)}\nabla f(x_t) + m(x_t - x_{t-1})$$

end

3.1 Introduction

One of the most iconic methods in first-order optimization is Gradient descent with momentum, also known as the Heavy-ball method (Polyak, 1964). This method enjoys widespread popularity both in its original formulation and in a stochastic variant that replaces the gradient by a stochastic estimate, a method that is behind many of the recent breakthroughs in deep learning (Sutskever et al., 2013).

A variant of the stochastic Heavy-ball where the step-sizes are chosen in *cyclical* order has recently come to the forefront of machine learning research, showing state-of-the-art results on different deep learning benchmarks (Loshchilov and Hutter, 2017; Smith, 2017). Inspired by this empirical success, we aim to study the convergence of the Heavy-ball algorithm where step-sizes $h_0, h_1, \ldots$ are not fixed or decreasing but instead chosen in cyclical order, as in Algorithm 2.

The Heavy-ball method with constant step-sizes enjoys a mature theory, where it is known for example to achieve optimal black-box worst-case complexity of quadratic convex optimization (Nemirovskii, 1992). In stark contrast, little is known about the convergence of the above variant with cyclical step-sizes. Our main motivating question is

> Do cyclical step-sizes improve convergence of Heavy-ball?

Our **main contribution** provides a positive answer to this question and, more importantly, *quantifies* the speedup under different assumptions. In particular, we show that for quadratic problems, whenever Hessian's spectrum belongs to two or more disjoint intervals, the Heavy-ball method with cyclical step-sizes achieves a faster worst-case convergence rate. Recent works have shown that this assumption on the spectrum is quite natural and occurs in many machine learning problems, including deep neural networks (Sagun et al., 2017; Papyan, 2018; Ghorbani et al., 2019; Papyan, 2019). The concurrent work of Oymak (2021) analyzes Gradient descent (without momentum, see extended comparison in Appendix 3.G) under this assumption. More precisely, we list our main contributions below.

- In sections 3.3 and 3.4, we provide a **tight convergence rate analysis** of the cyclical Heavy-ball method (Theorems 3.3.1 and 3.3.2 for two step-sizes, and Theorem 3.4.8 for the general case). This analysis highlights a regime under which this method achieves a faster worst-case rate than the accelerated rate of Heavy-ball, a phenomenon we refer to as *super-acceleration*. Theorem 3.5.1 extends the (local) convergence rate analysis results to non-quadratic objectives.

- As a byproduct of the convergence-rate analysis, we obtain an explicit expression for the **optimal parameters** in in the case of cycles of length two (Algorithm 3) and an implicit expression in terms of a system of K equations in the general case.
- Section 3.6 presents **numerical benchmarks** illustrating the improved convergence of the cyclical approach on 4 problems involving quadratic and logistic losses on both synthetic and a handwritten digits recognition dataset.
- Finally, we conclude in Section 3.7 with a discussion of this work's **limitations**.

3.2 Notation and Problem Setting

Throughout the paper (except in Section 3.5), we consider the problem of minimizing a quadratic function:

$$\min_{x\in\mathbb{R}^d} f(x)\,, \text{ with } \quad f\in\mathcal{C}_\Lambda, \tag{OPT}$$

where $\mathcal{C}_\Lambda$ is the class of quadratic functions with Hessian matrix H and whose Hessian spectrum $\mathrm{Sp}(H)$ is localized in $\Lambda\subseteq[\mu,L]\subseteq\mathbb{R}_{>0}$:

$$\mathcal{C}_\Lambda \triangleq \left\{f(x)=(x-x_*)^\top\tfrac{H}{2}(x-x_*)+f_*,\ \mathrm{Sp}(H)\subseteq\Lambda\right\}$$

The condition $\Lambda\subseteq[\mu,L]$ implies all quadratic functions under consideration are L-smooth and μ-strongly convex. For this function class, we define κ, the (inverse) condition number, and ρ, the ratio between the center of Λ and its radius, as

$$\kappa \triangleq \frac{\mu}{L}, \qquad \rho \triangleq \frac{L+\mu}{L-\mu} = \left(\frac{1+\kappa}{1-\kappa}\right). \tag{3.1}$$

Finally, for a method solving (OPT) that generates a sequence of iterates $\{x_t\}$, we define its worst-case rate r_t and its asymptotic rate factor τ as

$$r_t \triangleq \sup_{x_0\in\mathbb{R}^d,\, f\in\mathcal{C}_\Lambda} \frac{\|x_t-x_*\|}{\|x_0-x_*\|},\ 1-\tau \triangleq \limsup_{t\to\infty}\sqrt[t]{r_t}\,. \tag{3.2}$$

3.3 Super-acceleration with Cyclical Step-sizes

In this section we develop one of our main contributions, a convergence rate analysis of the cyclical Heavy-ball method with cycles of length 2. This analysis crucially depends on the location of the Hessian's eigenvalues; we assume that these are contained in a set Λ that is the union of 2 intervals *of the same size*

$$\Lambda = [\boldsymbol{\mu}_1,\,\boldsymbol{L}_1]\cup[\boldsymbol{\mu}_2,\,\boldsymbol{L}_2]\,,\ \boldsymbol{L}_1-\boldsymbol{\mu}_1=\boldsymbol{L}_2-\boldsymbol{\mu}_2\,. \tag{3.3}$$

By symmetry, this set is alternatively described by

$$\mu\triangleq\boldsymbol{\mu}_1,\quad L\triangleq\boldsymbol{L}_2 \quad\text{and}\quad R\triangleq\frac{\boldsymbol{\mu}_2-\boldsymbol{L}_1}{\boldsymbol{L}_2-\boldsymbol{\mu}_1}\,, \tag{3.4}$$

where R is the relative length of the gap $\boldsymbol{\mu}_2-\boldsymbol{L}_1$ with respect to the diameter $\boldsymbol{L}_2-\boldsymbol{\mu}_1$ (see Figure 3.1). This parametrization is convenient since the relative gap plays a crucial role in our convergence analysis. Our results allow $R=0$, therefore recovering the classical setting of Hessian eigenvalues contained in an interval.

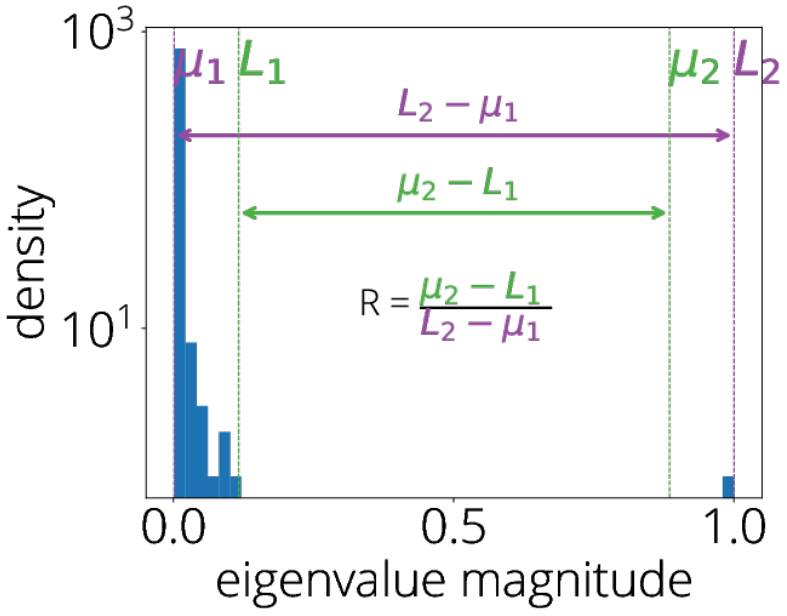


Figure 3.1: Hessian eigenvalue histogram for a quadratic objective on MNIST. The outlier eigenvalue at L_2 generates a non-zero relative gap $R = 0.77$. In this case, the 2-cycle Heavy-ball method has a faster asymptotic rate than the single-cycle one (see Section 3.3.2).

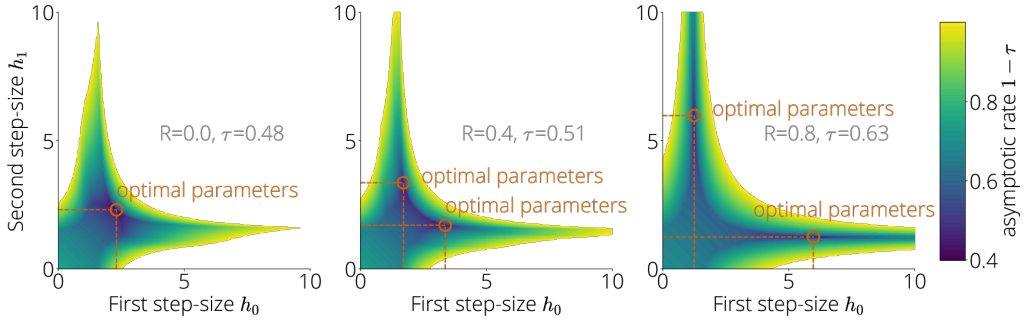


Figure 3.2: **Asymptotic rate of cyclical ($K = 2$) Heavy-ball** in terms of its step-sizes h_0, h_1 across 3 different values of the relative gap R. In the **left** plot, the relative gap is zero, and so the step-sizes with the smallest rate coincide ($h_0 = h_1$). For non-zero values of R (**center and right**), the optimal method instead alternates between two *different* step-sizes. In all plots the momentum parameter m is set according to Algorithm 3.

Through a correspondence between optimization methods and polynomials (see Section 3.4), we can derive a worst-case analysis for the cyclical Heavy-ball method. The outcome of this analysis is in the following theorem, that provides the asymptotic convergence rate of Algorithm 2 for cycles of length two. All proofs of results in this section can be found in Appendix 3.D.3.

Theorem 3.3.1 (Rate factor of $\mathrm{HB}_2(h_0, h_1; m)$)**.** *Let $f \in \mathcal{C}_\Lambda$ and consider the cyclical Heavy-ball method with step-sizes h_0, h_1 and momentum parameter m. The asymptotic rate factor of*

Algorithm 2 with cycles of length two is

$$1-\tau = \begin{cases} \sqrt{m} & \text{if } \sigma_* \leqslant 1, \\ \sqrt{m}\left(\sigma_* + \sqrt{\sigma_*^2 - 1}\right)^{\frac{1}{2}} & \text{if } \sigma_* \in \left(1,\ \frac{1+m^2}{2m}\right), \\ \geqslant 1 \text{ (no convergence)} & \text{if } \sigma_* \geqslant \frac{1+m^2}{2m}, \end{cases}$$

$$\text{with} \quad \sigma_* = \max_{\lambda \in \left\{\mu_1, L_1, \mu_2, L_2, (1+m)\frac{h_0+h_1}{2h_0h_1}\right\} \cap \Lambda} |\sigma_2(\lambda)|$$

$$\text{and} \quad \sigma_2(\lambda) = 2\left(\frac{1+m-\lambda h_0}{2\sqrt{m}}\right)\left(\frac{1+m-\lambda h_1}{2\sqrt{m}}\right) - 1\,.$$

3.3.1 Optimal algorithm

The previous theorem gives the convergence rate for all triplets (h_0, h_1, m). This allows us for instance to map out the associated convergence rate for every pair of step-sizes. As we illustrate in Figure 3.2, as we increase the relative gap (R), the optimal step-sizes become further apart.

Another application of the previous theorem is to find the parameters that minimize the asymptotic convergence rate. Although the process rather tedious and relegated to Appendix 3.D.3, the resulting momentum (m) and step-size parameters (h_0, h_1) are remarkably simple, and given by the expressions

$$m = \left(\frac{\sqrt{\rho^2 - R^2} - \sqrt{\rho^2 - 1}}{\sqrt{1 - R^2}}\right)^2 \tag{3.5}$$

$$h_0 = \frac{1+m}{\boldsymbol{L_1}} \qquad h_1 = \frac{1+m}{\boldsymbol{\mu_2}}\,. \tag{3.6}$$

Being one of our main contributions, this algorithm is also described in pseudocode in Algorithm 3. By construction, this method has an *asymptotically optimal* convergence rate which we detail in the next Corollary:

Algorithm 3 Cyclical $(K = 2)$ Heavy-ball with optimal parameters

Input: Initial iterate x_0, $\boldsymbol{\mu_1} < \boldsymbol{L_1} \leqslant \boldsymbol{\mu_2} < \boldsymbol{L_2}$ (where $\boldsymbol{L_1} - \boldsymbol{\mu_1} = \boldsymbol{L_2} - \boldsymbol{\mu_2}$)

Set: $\rho = \frac{\boldsymbol{L_2}+\boldsymbol{\mu_1}}{\boldsymbol{L_2}-\boldsymbol{\mu_1}}$, $R = \frac{\boldsymbol{\mu_2}-\boldsymbol{L_1}}{\boldsymbol{L_2}-\boldsymbol{\mu_1}}$,

$m = \left(\frac{\sqrt{\rho^2-R^2}-\sqrt{\rho^2-1}}{\sqrt{1-R^2}}\right)^2$

$x_1 = x_0 - \frac{1}{\boldsymbol{L_1}}\nabla f(x_0)$

for $t = 1, 2, \ldots$ **do**

$$h_t = \tfrac{1+m}{\boldsymbol{L_1}} \text{ (if } t \text{ is even)}, \quad h_t = \tfrac{1+m}{\boldsymbol{\mu_2}} \text{ (if } t \text{ is odd)}$$

$$x_{t+1} = x_t - h_t\nabla f(x_t) + m(x_t - x_{t-1})$$

end

Corollary 3.3.2. *The non-asymptotic and asymptotic worst-case rates $r_t^{\text{Alg. 2}}$ and $1 - \tau^{\text{Alg. 2}}$ of Algorithm 3 over $\mathcal{C}_\Lambda$ for even iteration number t are*

$$r_t^{\text{Alg. 2}} = \left(\frac{\sqrt{\rho^2-R^2}-\sqrt{\rho^2-1}}{\sqrt{1-R^2}}\right)^t \left(1 + t\sqrt{\frac{\rho^2-1}{\rho^2-R^2}}\right),$$

$$1 - \tau^{\text{Alg. 2}} = \frac{\sqrt{\rho^2 - R^2} - \sqrt{\rho^2 - 1}}{\sqrt{1 - R^2}}\,.$$

Note that this result also holds if we swap the 2 step-sizes in Algorithm 3.

Eigengap and accelerated cyclical step-sizes While Corollary 3.3.2 focuses on the optimal tuning of Algorithm 3, Theorem 3.D.1 provides general convergence analysis for non-optimal parameters. In the case of the existence of an eigengap, a range of cyclical step-sizes leads to an accelerated convergence rate (compared to the optimal constant step-size strategy) and therefore, an inexact parameters search can lead to such an acceleration.

3.3.2 Comparison with Polyak Heavy-ball

In the absence of eigenvalue gap ($R = 0$ and $\Lambda = [\mu, L]$), Algorithm 3 reduces to Polyak Heavy-ball (PHB) (Polyak, 1964), whose worst-case rate is detailed in Appendix 3.B. Since the asymptotic rate of Algorithm 3 is monotonically decreasing in R, the convergence rate of the cyclical variant is always better than PHB. Furthermore, in the ill-conditioned regime (small κ), the comparison is particularly simple: the optimal 2-cycle algorithm has a $\sqrt{1-R^2}$ relative improvement over PHB, as provided by the next proposition. A more thorough comparison for different support sets Λ is discussed in Table 3.1.

Proposition 3.3.3. *Let $R \in [0, 1)$. The rate factors of respectively Algorithm 3 and PHB verify*

$$\begin{aligned} 1 - \tau^{\text{Alg. 2}} &\underset{\kappa \to 0}{=} 1 - \tfrac{2\sqrt{\kappa}}{\sqrt{1-R^2}} + o(\sqrt{\kappa})\,, \\ 1 - \tau^{\text{PHB}} &\underset{\kappa \to 0}{=} 1 - 2\sqrt{\kappa} + o(\sqrt{\kappa})\,. \end{aligned} \tag{3.7}$$

Relative gap R	Set Λ	Rate factor τ	Speedup τ/τ^{PHB}
$R \in [0, 1)$	$[\mu, \mu + \frac{1-R}{2}(L-\mu)] \cup [L - \frac{1-R}{2}(L-\mu), L]$	$\frac{2\sqrt{\kappa}}{\sqrt{1-R^2}}$	$(1-R^2)^{-\frac{1}{2}}$
$R = 1 - \sqrt{\kappa}/2$	$[\mu, \mu + \frac{\sqrt{\mu L}}{4}] \cup [L - \frac{\sqrt{\mu L}}{4}, L]$	$2\sqrt[4]{\kappa}$	$\kappa^{-\frac{1}{4}}$
$R = 1 - 2\gamma\kappa$	$[\mu, (1+\gamma)\mu] \cup [L - \gamma\mu, L]$	indep. of κ	$O(\kappa^{-\frac{1}{2}})$

Table 3.1: Case study of the convergence of Algorithm 3 as a function of R, in the regime $\kappa \to 0$. The **first line** corresponds to the regime where R is independent of κ, and we observe a constant gain w.r.t. PHB. The **second line** considers a setting in which R depends on $\sqrt{\kappa}$, that is, the two intervals in Λ are relatively small. The asymptotic rate reads $(1 - 2\sqrt[4]{\kappa})^t$, improving over the $(1 - 2\sqrt{\kappa})^t$ rate of Polyak Heavy-ball, unimprovable when $R = 0$. Finally, in the **third line**, R depends on κ, the two intervals in Λ are so small that the convergence becomes $O(1)$, i.e., is independent of κ.

3.4 A constructive Approach: Minimax Polynomials

This section presents a generic framework that allows designing optimal momentum and step-size cycles for given sets Λ and cycle length K.

We first recall classical results that link optimal first-order methods on quadratics and Chebyshev polynomials. Then, we generalize the approach by showing that optimal methods can be viewed as combinations of Chebyshev polynomials, and minimax polynomials σ_K^Λ of degree K over the set Λ. Finally, we show how to recover the step-size schedule from σ_K^Λ and present the general algorithm (Algorithm 4).

3.4.1 First-Order Methods on Quadratics and Polynomials

A key property that we will use extensively in the analysis is the following link between first-order methods and polynomials.

Proposition 3.4.1. *Let $f \in \mathcal{C}_\Lambda$. The iterates x_t satisfy*

$$x_{t+1} \in x_0 + \operatorname{span}\{\nabla f(x_0), \dots, \nabla f(x_t)\}\,, \tag{3.8}$$

where x_0 is the initial approximation of x_*, if and only if there exists a sequence of polynomials $(P_t)_{t\in\mathbb{N}}$, each of degree at most 1 more than the highest degree of all previous polynomials and P_0 of degree 0 (hence the degree of P_t is at most t), such that

$$\forall\, t \quad x_t - x_* = P_t(H)(x_0 - x_*), \quad P_t(0) = 1\,. \tag{3.9}$$

Example 3.4.2 (Gradient descent)**.** *Consider the Gradient descent algorithm with fixed step-size h, applied to problem* (OPT)*. Then, after unrolling the update, we have*

$$\begin{aligned} x_{t+1} - x_* &= x_t - x_* - h\nabla f(x_t) \\ &= x_t - x_* - hH(x_t - x_*) \\ &= (I - hH)^{t+1}(x_0 - x_*)\,. \end{aligned} \tag{3.10}$$

In this case, the polynomial associated to Gradient descent is $P_t(\lambda) = (1 - h\lambda)^t$.

The above proposition can be used to obtain worst-case rates for first-order methods by bounding their associated polynomials. Indeed, using the Cauchy-Schwartz inequality in (3.9) leads to

$$\begin{aligned} &\|x_t - x_*\| \leqslant \sup_{\lambda\in\Lambda} |P_t(\lambda)|\, \|x_0 - x_*\| \\ \implies \quad & r_t = \sup_{\lambda\in\Lambda} |P_t(\lambda)|, \quad \text{where } P_t(0) = 1\,. \end{aligned} \tag{3.11}$$

Therefore, finding the algorithm with the fastest worst-case rate can be equivalently framed as the problem of finding the polynomial with smallest value on the eigenvalue support Λ, subject to the normalization condition $P_t(0) = 1$. Such polynomials are referred to as **minimax**. Throughout the paper, we use this polynomial-based approach to find methods with optimal rates.

An important property of minimax polynomials is their *equioscillation* on Λ (see Theorem 3.C.1 and its proof for a formal statement).

Definition 3.4.3. *(Equioscillation) A polynomial P_t of degree t equioscillates on Λ if it verifies $P_t(0) = 1$ and there exist $\lambda_0 < \lambda_1 < \ldots < \lambda_t \in \overline{\Lambda}$ such that*

$$P_t(\lambda_i) = (-1)^i \max_{\lambda\in\Lambda} |P_t(\Lambda)|\,. \tag{3.12}$$

Example 3.4.4 (Λ is an interval)**.** *The t-th order Chebyshev polynomials of the first kind T_t satisfy the* equioscillation *property on $[-1, 1]$. It follows that minimax polynomials on $\Lambda = [\mu, L]$ can be obtained by composing the Chebyshev polynomial T_t with the linear transformation σ_1^Λ:*

$$\frac{T_t\left(\sigma_1^\Lambda(\lambda)\right)}{T_t\left(\sigma_1^\Lambda(0)\right)} = \underset{P\in\mathbb{R}_t[X],P(0)=1}{\arg\min} \sup_{\lambda\in\Lambda}|P(\lambda)|\,, \qquad (3.13)$$
$$\text{with}\quad \sigma_1^\Lambda(\lambda) = \frac{L+\mu}{L-\mu} - \frac{2}{L-\mu}\lambda\,,$$

where σ_1^Λ maps the interval $[\mu, L]$ to $[-1, 1]$. The optimization method associated with this minimax polynomial is the Chebyshev semi-terative method (Flanders and Shortley, 1950; Golub and Varga, 1961), described also in Appendix 3.B.1. This method achieves the lower complexity bound for smooth strongly convex quadratic minimization (Nemirovskii, 1994, Chapter 12) or (Nemirovskii, 1992; Nesterov, 2003).

The next proposition provides the main results in this subsection, which is key for obtaining Algorithm 3. It characterizes the even degree minimax polynomial in the setting of Section 3.3, that is, when Λ is the union of 2 intervals of same size. In this case, the minimax solution is also based on Chebyshev polynomials, but composed with a degree-two polynomial σ_2^Λ.

Proposition 3.4.5. *Let $\Lambda = [\boldsymbol{\mu}_1, \boldsymbol{L}_1] \cup [\boldsymbol{\mu}_2, \boldsymbol{L}_2]$ be an union of two intervals of the same size ($\boldsymbol{L}_1 - \boldsymbol{\mu}_1 = \boldsymbol{L}_2 - \boldsymbol{\mu}_2$) and let m, h_0, h_1 be as defined in Algorithm 3. Then the minimax polynomial (solution to (3.12)) is, for all $t = 2n$, $n \in \mathbb{N}_0^+$,*

$$\frac{T_n\left(\sigma_2^\Lambda(\lambda)\right)}{T_n\left(\sigma_2^\Lambda(0)\right)} = \underset{\substack{P\in\mathbb{R}_t[X],\\ P(0)=1}}{\arg\min} \sup_{\lambda\in\Lambda}|P(\lambda)|\,,$$
$$\text{with}\quad \sigma_2^\Lambda(\lambda) = \frac{1}{2m}\left(1+m-\lambda h_0\right)\left(1+m-\lambda h_1\right) - 1\,.$$

3.4.2 Generalization to Longer Cycles

The polynomial in Example 3.4.4 uses a linear link function σ_1^Λ to map Λ to $[-1, 1]$. In Proposition 3.4.5, we see that a degree *two* link function σ_2^Λ can be used to find the minimax polynomial when Λ is the union of two intervals. This section generalizes this approach and considers higher-order polynomials for σ_K.

We start with the following parametrization, with an arbitrary polynomial σ_K of degree K,

$$P_t(\lambda;\sigma_K) \triangleq \frac{T_n\left(\sigma_K(\lambda)\right)}{T_n\left(\sigma_K(0)\right)}, \quad \forall t = Kn,\, n\in\mathbb{N}_0^+\,. \qquad (3.14)$$

As we will see in the next subsection, this parametrization allows considering cycles of step-sizes. Our goal now is to find the σ_K that obtains the fastest convergence rate possible. The next proposition quantifies its impact on the asymptotic rate and its proof can be found in Subsection 3.D.1.

Proposition 3.4.6. *For a given σ_K such that $\sup_{\lambda\in\Lambda}|\sigma_K(\lambda)| = 1$, the asymptotic rate factor τ^{σ_K} of the method associated to the polynomial (3.14) is*

$$1 - \tau^{\sigma_K} = \lim_{t\to\infty} \sqrt[t]{\sup_{\lambda\in\Lambda}|P_t(\lambda;\sigma_K)|} = \left(\sigma_0 - \sqrt{\sigma_0^2 - 1}\right)^{\frac{1}{K}}, \quad \text{with } \sigma_0 \triangleq \sigma_K(0)\,. \tag{3.15}$$

For a fixed K, the asymptotic rate (3.15) is a decreasing function of σ_0. This motivates the introduction of the "optimal" degree K polynomial σ_K^Λ as the one that solves

$$\sigma_K^\Lambda \triangleq \underset{\sigma\in\mathbb{R}_K[X]}{\arg\max}\, \sigma(0) \quad \text{s.t.} \quad \sup_{\lambda\in\Lambda}|\sigma(\lambda)| \leqslant 1\,. \tag{3.16}$$

Using the above definition, we recover the σ_1^Λ and σ_2^Λ from Example 3.4.4 and Proposition 3.4.5.

Finding the polynomial. Finding an exact and explicit solution for the general K and Λ case is unfortunately out of reach, as it involves solving a system of K non-linear equations. Here we describe an approximate approach. Let $\sigma_K^\Lambda(x) = \sum_{i=0}^K \sigma_i x^i$. We propose to discretize Λ into N different points $\{\lambda_j\}$, then solve the linear problem

$$\max_{\sigma_i}\ \sigma_0 \quad \text{s.t.} \quad -1 \leqslant \textstyle\sum_{i=0}^K \sigma_i \lambda_j^i \leqslant 1, \quad \forall j = 1,\dots,N\,. \tag{3.17}$$

To check the optimality, it suffices to verify that the polynomial σ_K^Λ satisfies the *equioscillation* property (Definition 3.4.3), as depicted in Figure 3.3.

Remark 3.4.7 (Relationship between optimal and minimax polynomials)**.** *For later reference, we note that the optimal polynomial σ_K^Λ is equivalent to finding a minimax polynomial on Λ and to rescale it. More precisely, σ_K^Λ is optimal if and only if $\sigma_K^\Lambda/\sigma_K^\Lambda(0)$ is minimax.*

3.4.3 Cyclical Heavy-ball and (Non-)asymptotic Rates of Convergence

We now describe the link between σ_K^Λ and Algorithm 4. Using the recurrence for Chebyshev polynomials of the first kind in (3.14), we have $\forall t = Kn$, $n \in \mathbb{N}_0^+$,

$$\frac{T_{n+1}(\sigma_K^\Lambda(\lambda))}{T_{n+1}(\sigma_K^\Lambda(0))} = 2\sigma_K^\Lambda(\lambda)\left[\frac{T_n(\sigma_K^\Lambda(\lambda))}{T_n(\sigma_K^\Lambda(0))}\right]\underbrace{\left[\frac{T_n(\sigma_K^\Lambda(0))}{T_{n+1}(\sigma_K^\Lambda(0))}\right]}_{=a_n} - \left[\frac{T_{n-1}(\sigma_K^\Lambda(\lambda))}{T_{n-1}(\sigma_K^\Lambda(0))}\right]\underbrace{\left[\frac{T_{n-1}(\sigma_K^\Lambda(0))}{T_{n+1}(\sigma_K^\Lambda(0))}\right]}_{=b_n}.$$

It still remains to find an algorithm associated with this polynomial. To obtain one in the form of Algorithm 2, one can use the stationary behavior of the recurrence. From (Scieur and Pedregosa, 2020), the coefficients a_n and b_n converge as $n \to \infty$ to their fixed points a_∞ and b_∞. We therefore consider here an asymptotic polynomial $\bar{P}_t(\lambda;\sigma_K^\Lambda)$, whose recurrence satisfies

$$\bar{P}_t(\lambda;\sigma_K^\Lambda) = 2a_\infty\sigma_K^\Lambda(\lambda)\bar{P}_{t-K}(\lambda;\sigma_K^\Lambda) - b_\infty\bar{P}_{t-2K}(\lambda;\sigma_K^\Lambda)\,. \tag{3.18}$$

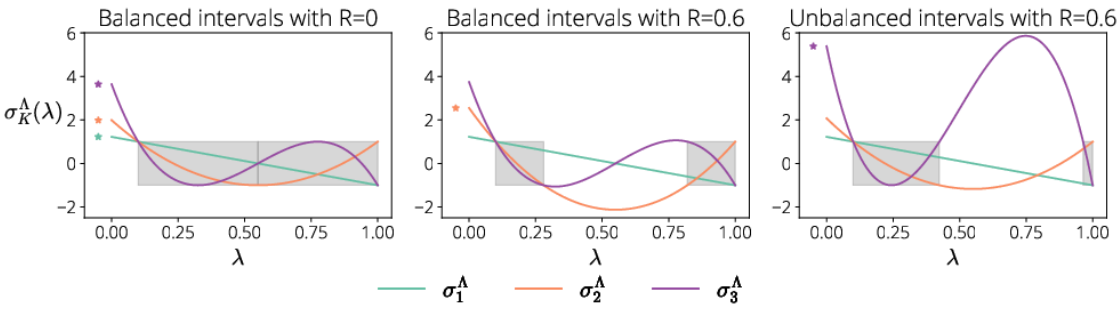


Figure 3.3: Examples of optimal polynomials σ_K^Λ from (3.16), all of them verifying the equioscillation property (Definition 3.4.3). The "$\star$" symbol highlights the degree of σ_K^Λ that achieves the best asymptotic rate $\tau^{\sigma_K^\Lambda}$ in (3.15) amongst all K (see Section 3.4.4). **(Left)** When Λ is a unique interval, all 3 polynomials are equivalently optimal $\tau^{\sigma_1^\Lambda} = \tau^{\sigma_2^\Lambda} = \tau^{\sigma_3^\Lambda}$. **(Center)** When Λ is the union of two intervals of the same size, the degree 2 polynomial is optimal $\tau^{\sigma_2^\Lambda} > \tau^{\sigma_3^\Lambda} > \tau^{\sigma_1^\Lambda}$. This is expected given the result in Proposition 3.4.5. **(Right)** When Λ is the union of two unbalanced intervals, the degree 3 polynomial instead achieves the best asymptotic rate $\tau^{\sigma_3^\Lambda} > \tau^{\sigma_2^\Lambda} > \tau^{\sigma_1^\Lambda}$ (see Section 3.4.4).

Similarly to $K = 1$, where this limit recursion corresponds to PHB, this recursion corresponds to an instance of Algorithm 4 (see Proposition 3.4.9 below), further motivating the cyclical Heavy-ball algorithm.

The following theorem is the main result of this section and characterizes the convergence rate of Algorithm 2 for arbitrary momentum and step-size sequence $\{h_i\}_{i \in [\![1,K]\!]}$.

Theorem 3.4.8. *With an arbitrary momentum m and an arbitrary sequence of step-sizes $\{h_i\}$, the worst-case convergence rate $1 - \tau$ of Algorithm 2 on $\mathcal{C}_\Lambda$ is*

$$\begin{cases} \sqrt{m} & \text{if } \sigma_* \leqslant 1 \\ \sqrt{m}\left(\sigma_* + \sqrt{\sigma_*^2 - 1}\right)^{K-1} & \text{if } \sigma_* \in \left(1, \dfrac{1+m^K}{2\left(\sqrt{m}\right)^K}\right) \\ \geqslant 1 \text{ (no convergence)} & \text{if } \sigma_* \geqslant \dfrac{1+m^K}{2\left(\sqrt{m}\right)^K}, \end{cases} \tag{3.19}$$

where $\sigma_* \triangleq \sup_{\lambda \in \Lambda} |\sigma(\lambda; \{h_i\}, m)|$, $\sigma(\lambda; \{h_i\}, m)$ is the K-degree polynomial

$$\sigma(\lambda; \{h_i\}, m) \triangleq \frac{1}{2} \mathrm{Tr}\left(M_1 M_2 \dots M_K\right), \tag{3.20}$$

and $M_i = \begin{bmatrix} \frac{1+m-h_{K-i}\lambda}{\sqrt{m}} & -1 \\ 1 & 0 \end{bmatrix}$.

By optimizing over these parameters, we obtain the Algorithm 4, a method associated to (3.18), whose rate is described in Proposition 3.4.9. All proofs can be found in Appendix 3.D.2.

Proposition 3.4.9. *Let $\sigma(\lambda; \{h_i\}, m)$ be the polynomial defined by (3.20), and σ_K^Λ be the optimal link function of degree K defined by (3.16). If the momentum m and the sequence of step-sizes $\{h_i\}$ satisfy*

$$\sigma(\lambda; \{h_i\}, m) = \sigma_K^\Lambda(\lambda), \tag{3.21}$$

Algorithm 4 Cyclical (arbitrary K) heavy-ball with optimal parameters

Input: Eigenvalue localization Λ, cycle length K, initialization x_0.

Preprocessing:

1. Find the polynomial σ_K^Λ such that it satisfies (3.16).
2. Set step-sizes $\{h_i\}_{i=0,\dots,K-1}$ and momentum m that satisfy resp. equations (3.21) and (3.22).

Set $x_1 = x_0 - \frac{h_0}{1+m}\nabla f(x_0)$

for $t = 1, 2, \dots$ **do**

$$x_{t+1} = x_t - h_{\mathrm{mod}(t,K)}\nabla f(x_t) + m(x_t - x_{t-1})$$

end

*then **1)** the parameters are optimal, in the sense that they minimize the asymptotic rate factor from Theorem 3.4.8, **2)** the optimal momentum parameter is*

$$m = (\sigma_0 - \sqrt{\sigma_0^2 - 1})^{2/K}, \quad \text{where } \sigma_0 = \sigma_K^\Lambda(0)\,, \tag{3.22}$$

***3)** the iterates from Algo. 4 with parameters $\{h_i\}$ and m form a polynomial with recurrence (3.18), and **4)** Algorithm 4 achieves the worst-case rate $r_t^{\text{Alg. 3}}$ and the asymptotic rate factor $1 - \tau^{\text{Alg. 3}}$*

$$r_t^{\text{Alg. 3}} = O\left(t\left(\sigma_0 - \sqrt{\sigma_0^2 - 1}\right)^{t/K}\right), \tag{3.23}$$
$$1 - \tau^{\text{Alg. 3}} = \left(\sigma_0 - \sqrt{\sigma_0^2 - 1}\right)^{1/K}.$$

Solving the system (3.21) The system is constructed by identification of the coefficients in both polynomials σ_K^Λ and $\sigma(\lambda; \{h_i\}, m)$, which can be solved using a naive grid-search for instance. We are not aware of any efficient algorithm to solve this system exactly, although it is possible to use iterative methods such as steepest descent or Newton's method.

3.4.4 Best Achievables Worst-case Guarantees on $\mathcal{C}_\Lambda$

This section discusses the (asymptotic) optimality of Algorithm 4. In Section 3.4.2, the polynomial $P_t(\,\cdot\,; \sigma_K^\Lambda)$ was written as a composition of Chebyshev polynomials with σ_K^Λ, defined in (3.16). The best K is chosen as follows: we solve (3.16) for several values of K, then pick the smallest K among the minimizers of (3.15). However, following such steps does not guarantee that the polynomial $P_{t,K}^\Lambda$ is *minimax*, as it is not guaranteed to minimize the worst-case rate $\sup_{\lambda\in\Lambda} |P_t(\lambda)|$ (see (3.11)).

We give here an optimality certificate, linked to a generalized version of *equioscillation*. In short, if we can find K non overlapping intervals (more formally, whose interiors are disjoint) Λ_i in Λ such that $\sigma_K^\Lambda(\Lambda_i) = [-1, 1]$ then $P_{t,K}^\Lambda$ is minimax for $t = nK$, $n \in \mathbb{N}_0^+$. The precise result is in Theorem 3.C.2. A direct consequence is the asymptotic optimality of Algorithm 4.

We note that σ_K^Λ might not exist for a given Λ. A complete characterization of the set Λ for which σ_K^Λ exists is out of the scope of this paper. A partial answer is given in (Fischer, 2011) when Λ is the union of two intervals but the general case remains open.

3.5 Local Convergence for Non-Quadratic Functions

When f is twice-differentiable, we show local convergence rates of Algorithm 2 (see proof in Section 3.E). As with Polyak heavy-ball acceleration, these results are local, as the only known convergence results for Polyak heavy-ball beyond quadratic objectives do not lead to an acceleration with respect to Gradient descent without momentum (See Ghadimi et al., 2015, Theorem 4). Moreover, it is possible to find pathological counter-examples and a specific initialization for which the method does not converge globally. Lessard et al. (2016, Figure 7) provides such a counter-example. Note the latest is *not* twice differentiable, but that a twice differentiable counter-example can be derived from the latest, using for instance convolutions.

Theorem 3.5.1 (Local convergence)**.** *Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ be a twice continuously differentiable function, x_* a local minimizer, and H be the Hessian of f at x_* with $Sp(H) \subseteq \Lambda$. Let x_t denote the result of running Algorithm 2 with parameters $h_1, h_2, \cdots, h_K, m$, and let $1 - \tau$ be the linear convergence rate on the quadratic objective* (OPT)*. Then we have*

$$\begin{aligned} &\forall \varepsilon > 0, \exists \text{ open set } V_\varepsilon : x_0,\, x_* \in V_\varepsilon \\ \implies &\|x_t - x_*\| = O((1 - \tau + \varepsilon)^t)\|x_0 - x_*\|. \end{aligned} \tag{3.24}$$

where $\|\cdot\|$ denotes the Euclidean norm.

In short, when Algorithm 2 is guaranteed to converge at rate $1 - \tau$ on (OPT), then the convergence rate on a nonlinear functions can be arbitrary close to $1 - \tau$ when x_0 is sufficiently close to x_*.

3.6 Experiments

In this section we present an empirical comparison of the cyclical heavy-ball method for different length cycles across 4 different problems. We consider two different problems, quadratic and logistic regression, each applied on two datasets, the MNIST handwritten digits (Le Cun et al., 2010) and a synthetic dataset. The results of these experiments, together with a histogram of the Hessian's eigenvalues are presented in Figure 3.4 (see caption for a discussion).

Dataset description. The MNIST dataset consists of a data matrix A with 60000 images of handwritten digits each one with $28 \times 28 = 784$ pixels. The *synthetic* dataset is generated according to a spiked covariance model (Johnstone, 2001), which has been shown to be an accurate model of covariance matrices arising for instance in spectral clustering (Couillet and Benaych-Georges, 2016) and deep networks (Pennington and Worah, 2017; Granziol et al., 2020). In this model, the data matrix $A = XZ$ is generated from a $m \times n$ random Gaussian matrix X and an $m \times m$ deterministic matrix Z. In our case, we take $n = 1000, m = 1200$ and Z is the identity where the first three entries are multiplied by 100 (this will lead to three outlier eigenvalues). We also generate an n-dimensional target vector b as $b = Ax$ or $b = \text{sign}(Ax)$ for the quadratic and logistic problem respectively.

Objective function For each dataset, we consider a quadratic and a logistic regression problem, leading to 4 different problems. All problems are of the form $\min_{x \in \mathbb{R}^p} \frac{1}{n} \sum_{i=1}^n \ell(A_i^\top x, b_i) + \lambda \|x\|^2$, where ℓ is a quadratic or logistic loss, A is the data matrix and b are the target values. We set the regularization parameter to $\lambda = 10^{-3}\|A\|^2$. For logistic regression, since

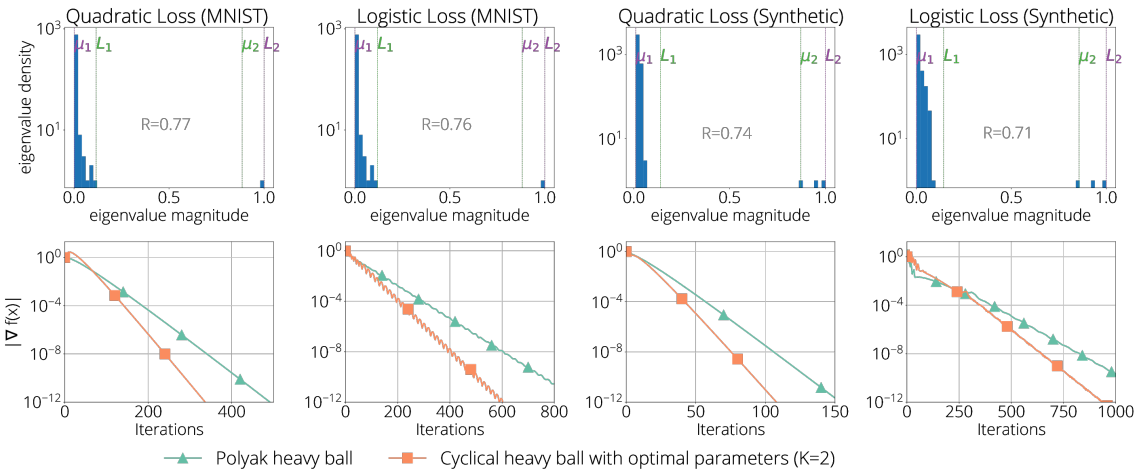


Figure 3.4: *Hessian Eigenvalue histogram (top row) and Benchmarks (bottom row)*. The **top row** shows the Hessian eigenvalue histogram at optimum for the 4 considered problems, together with the interval boundaries $\mu_1 < L_1 < \mu_2 < L_2$ for the two-interval split of the eigenvalue support described in Section 3.3. In all cases, there's a non-zero gap radius R. This is shown in the **bottom row**, where we compare the suboptimality in terms of gradient norm as a function of the number of iterations. As predicted by the theory, the non-zero gap radius translates into a faster convergence of the cyclical approach, compared to PHB in all cases. The improvement is observed on both quadratic and logistic regression problems, even through the theory for the latter is limited to *local* convergence.

guarantees only hold at a neighborhood of the solution (even for the 1-cycle algorithm), we initialize the first iterate as the result of 100 iteration of Gradient descent. In the case of logistic regression, the Hessian eigenvalues are computed at the optimum.

3.7 Conclusion

This work is motivated by two recent observations from the optimization practice of machine learning. First, cyclical step-sizes have been shown to enjoy excellent empirical convergence (Loshchilov and Hutter, 2017; Smith, 2017). Second, *spectral gaps* are pervasive in the Hessian spectrum of deep learning models (Sagun et al., 2017; Papyan, 2018; Ghorbani et al., 2019; Papyan, 2019). Based on the simpler context of quadratic convex minimization, we develop a convergence-rate analysis and optimal parameters for the heavy-ball method with cyclical step-sizes. This analysis highlights the regimes under which cyclical step-sizes have faster rates than classical accelerated methods. Finally, we illustrate these findings through numerical benchmarks.

Main Limitations. In Section 3.3 we gave explicit formulas for the optimal parameters in the case of the 2-cycle heavy-ball algorithm. These formulas depend not only on extremal eigenvalues—as is usual for accelerated methods—but also on the spectral gap R. The gap can sometimes be estimated after computing the top eigenvalues (e.g. top-2 eigenvalue for MNIST). However, in general, there is no guarantee on how many eigenvalues are needed to estimate it and it must sometimes be seen as hyperparameter. Note Theorem 3.3.1 provides a convergence analysis also for non-optimal parameters, which would give accelerated

convergence rates when doing a coarse grid-search over parameters as it is often done in empirical works.

Another limitation is the fact global convergence results rely heavily on the quadratic assumption which is quite different from our motivation, namely optimizing neural networks. Even if we provide local convergence guarantee in Section 3.5, we are not able to estimate the size of the optimum neighborhood for which Theorem 3.5.1 holds.

Another limitation regards long cycles. For cycles longer than 2, we gave an implicit formula to set the optimal parameters (Proposition 3.4.9). This involves solving a set of non-linear equations whose complexity increases with the cycle length. That being said, cyclical step-sizes might significantly enhance convergence speeds both in terms of worst-case rates and empirically, and this work advocates that new tuning practices involving different cycle lengths might be relevant.

Organization of the appendix

The appendix contains all proofs that were not presented in the main core of the paper. We also detail all examples, and provide some complementary elements.

Section 3.A details the existing link between first-order methods and family of "residual polynomials". This term refers in all the appendix to the polynomials which value in 0 is 1.

In Section 3.B, we recall some well known optimal methods for L-smooth μ-strongly convex quadratic minimization (i.e., when the spectrum is contained in a single interval $\Lambda = [\mu, L]$). Its purpose is exclusively to recall well-known foundation of optimization that are those algorithms and their construction.

In Section 3.C, we recall the polynomial formulation of the optimal method design problem, as well as a fundamental property, called "equioscillation", to characterize the solution of this problem.

In Appendix 3.D, we provide all proofs related to cyclic step-sizes. In particular,

- In Subsection 3.D.1, we derive the optimal algorithm in a case where Λ is the union of 2 intervals of the same size (See (3.3)). This leads to the use of alternating step-sizes. The resulting algorithm has a stationary form which is Algorithm 3.
- Therefore, in Subsection 3.D.2, we study the Heavy-ball with cycling step-sizes (Algorithm 2).
- In Subsection 3.D.3 and Subsection 3.D.4, we use our results to design methods with cycles of lengths $K = 2$ and $K = 3$. For those cases, we provide a more elegant formulation of the results.

In Section 3.E, we provide a proof of Theorem 3.5.1 (local behavior beyond quadratics) and in Section 3.F, we provide some information about the code we used for the experiments in quadratic and non quadratic settings.

Finally, in Section 3.G we discuss similarities and differences with Oymak (2021).

3.A Relationship between first-order methods and polynomials

In this section we prove some results on the relationship between polynomials and first-order methods for quadratic minimization, which is the starting point for our theoretical framework. This relationship is classical and was exploited by Rutishauser (1959); Nemirovskii (1992, 1994)), to name a few. The following proposition makes this relationship precise:

Proposition 3.4.1. *Let $f \in \mathcal{C}_\Lambda$. The iterates x_t satisfy*

$$x_{t+1} \in x_0 + \operatorname{span}\{\nabla f(x_0), \dots, \nabla f(x_t)\}\,, \tag{3.8}$$

where x_0 is the initial approximation of x_*, if and only if there exists a sequence of polynomials $(P_t)_{t\in\mathbb{N}}$, each of degree at most 1 more than the highest degree of all previous polynomials and P_0 of degree 0 (hence the degree of P_t is at most t), such that

$$\forall\, t \quad x_t - x_* = P_t(H)(x_0 - x_*), \quad P_t(0) = 1\,. \tag{3.9}$$

Proof. We successively prove both directions of the equivalence.

($\Longrightarrow$) *Given a first-order method, we can find a sequence of polynomials $(P_t)_{t\in\mathbb{N}}$ such that, for a given quadratic function f of Hessian H and a given starting point x_0, the iterates x_t verify*

$$x_t - x_* = P_t(H)(x_0 - x_*).$$

Moreover, The polynomials sequence $(P_t)_{t\in\mathbb{N}}$ verifies the relations

$$\deg(P_{t+1}) \leqslant \max_{k\leqslant t} \deg(P_k) + 1 \quad \textit{and} \quad P_t(0) = 1.$$

We proceed by induction:

Initial case. Let $t = 0$. Then for any first-order method we have the trivial relationship

$$x_0 - x_* = P_0(H)(x_0 - x_*) \quad \text{with} \quad P_0 = 1.$$

This proves the implication for $t = 0$, as P_0 is a degree 0 polynomial satisfying $P_0(0) = 1$.

Recursion. Let $t \in \mathbb{N}$. We assume the following statement true,

$$\forall\, k \leqslant t, \quad x_k - x_* = P_k(H)(x_0 - x_*) \quad \text{with} \quad P_k(0) = 1.$$

We now prove this statement is also true for $t+1$. Since $x_{t+1} \in x_0 + \text{span}\{\nabla f(x_0), \dots, \nabla f(x_t)\}$, there exists a family $(\gamma_{t+1,k})_{k\in[\![0,t]\!]}$ such that

$$x_{t+1} = x_0 - \gamma_{t+1,0}\nabla f(x_0) - \cdots - \gamma_{t+1,t}\nabla f(x_t). \tag{3.25}$$

Then, by the induction hypothesis we have:

$$\begin{aligned} x_{t+1} - x_* &= x_0 - x_* - \gamma_{t+1,0}H(x_0 - x_*) - \cdots - \gamma_{t+1,t}H(x_t - x_*) \\ &= x_0 - x_* - \gamma_{t+1,0}HP_0(H)(x_0 - x_*) - \cdots - \gamma_{t+1,t}HP_t(H)(x_0 - x_*) \\ &\triangleq P_{t+1}(H)(x_0 - x_*)\,. \end{aligned}$$

We observe that the latest polynomial has a degree at most 1 plus the highest degree of $(P_k)_{k\leqslant t}$ and that $P_{t+1}(0) = 1$ (since P_{t+1} is defined as 1 plus some polynomial multiple of the polynomial X), which concludes the proof.

($\Longleftarrow$): *From a family of polynomials $(P_t)_{t\in\mathbb{N}}$, with*

$$\deg(P_{t+1}) \leqslant \max_{k\leqslant t} \deg(P_k) + 1 \quad \textit{and} \quad P_t(0) = 1, \tag{3.26}$$

we can obtain a first-order method such that, for any quadratic f (and its Hessian H) and any starting point x_0, we verify

$$\forall t \in \mathbb{N}, x_t - x_* = P_t(H)(x_0 - x_*).$$

Let the sequence $(P_t)_{t\in\mathbb{N}}$ verifies (3.26) for all $t \in \mathbb{N}$. Let

$$d = \max_{t'\leqslant t} \deg(P_{t'}).$$

A gap in the sequence of degrees would stand in contradiction with our assumptions.

Since, there is no gap in degree, for any $d' \leqslant d$ there exists $t' \leqslant t$ such that $\deg(P_{t'}) = d'$, and therefore $\mathrm{Span}((P_k)_{k\leqslant t}) = \mathbb{R}_d[X]$.

Moreover, we know P_{t+1} has a degree at most $d+1$ and $P_{t+1}(0) = 1$, so $\frac{1-P_{t+1}(X)}{X} \in \mathbb{R}_d[X]$.

This proves the existence of $(\gamma_{t+1,k})_{k\in[\![0,t]\!]}$ such that

$$\frac{1-P_{t+1}(X)}{X} = \gamma_{t+1,0}P_0(X) + \cdots + \gamma_{t+1,t}P_t(X). \tag{3.27}$$

Then, defining

$$x_{t+1} = x_0 - \gamma_{t+1,0}\nabla f(x_0) - \cdots - \gamma_{t+1,t}\nabla f(x_t)\,, \tag{3.28}$$

we have

$$\begin{aligned} x_{t+1} - x_* &= x_0 - x_* - H\left(\gamma_{t+1,0}(x_0 - x_*) + \cdots + \gamma_{t+1,t}(x_t - x_*)\right) && (3.29)\\ &= \left(1 - X\left(\gamma_{t+1,0}P_0(X) + \cdots + \gamma_{t+1,t}P_t(X)\right)\right)(H)(x_0 - x_*) && (3.30)\\ &= P_{t+1}(H)(x_0 - x_*)\,. && (3.31) \end{aligned}$$

Defining x_t for all t according to (3.28) gives an algorithm that has as associated residual polynomials $(P_t)_{t\in\mathbb{N}}$. ■

The above proposition can be used to obtain worst-case rates for first-order methods by bounding their associated polynomials. Indeed, using the Cauchy-Schwartz inequality in (3.9) leads to

$$\|x_t - x_*\| \leqslant \sup_{\lambda\in\Lambda}|P_t(\lambda)|\,\|x_0 - x_*\| \quad\Longrightarrow\quad r_t = \sup_{\lambda\in\Lambda}|P_t(\lambda)|, \quad \text{where } P(0) = 1\,. \tag{3.32}$$

Therefore, finding the algorithm with the fastest worst-case rate can be equivalently framed as the problem of finding the residual polynomial with smallest value on the eigenvalue support Λ.

Then, finding the fastest algorithm is equivalent of finding, for each $t \geqslant 0$, the polynomial of degree t that reaches the smallest infinite norm on the set Λ. Therefore we introduce the notion of *minimax polynomial* (Definition 3.A.1) over a set Λ as the one that reaches the smallest maximal value over Λ among a set of polynomial of fixed degree and $P(0) = 1$.

Definition 3.A.1 (Minimax polynomial of degree t over Λ)**.** *For any, $t \geqslant 0$, and any relatively compact (i.e. bounded) set $\Lambda \subset \mathbb{R}$, the minimax polynomial of degree t over Λ, written Z_t^Λ, is defined as*

$$Z_t^\Lambda \triangleq \operatorname*{argmin}_{P\in\mathbb{R}_t[X]} \sup_{\lambda\in\Lambda}|P(\lambda)|, \quad \text{subject to} \quad P(0) = 1\,. \tag{3.33}$$

3.B Optimal methods for strongly convex and smooth quadratic objective

In this section, for sake of completness, we revisit some classical methods, described in e.g. (Polyak, 1964; Goh, 2017; Pedregosa, 2020, 2021a), that are optimal when the Hessian eigenvalues are contained in a single interval of the form $\Lambda = [\mu, L]$. To make this setup explicit, we will denote the optimal polynomials σ_1^Λ and Z_t^Λ (respectively defined in Equation (3.16) and Equation (3.33)) by $\sigma_1^{[\mu,L]}$, and $Z_t^{[\mu,L]}$.

As mentioned in Example 3.4.4, the minimax polynomial $Z_t^{[\mu,L]}$ is

$$Z_t^{[\mu,L]}(\lambda) = \frac{T_t(\sigma_1^{[\mu,L]}(\lambda))}{T_t(\sigma_1^{[\mu,L]}(0))},$$

where T_t denotes the t^{th} Chebyshev polynomial (See e.g. Chebyshev (1853)) and $\sigma_1^{[\mu,L]}$ the affine function $\sigma(\lambda) \triangleq \frac{L+\mu}{L-\mu} - \frac{2}{L-\mu}\lambda$ that maps $[\mu, L]$ onto $[-1, 1]$. This can be seen a consequence of the more general *equioscillation* discussed in Appendix 3.C. The next section presents one method which has $Z_t^{[\mu,L]}$ as associated residual polynomial. This method is known as the Chebyshev semi-iterative method.

3.B.1 Chebyshev semi-iterative method

The algorithm follows the three terms pattern from Equation (3.13) to iteratively form $Z_1^\Lambda, \dots, Z_t^\Lambda$.

Algorithm 5 Chebyshev semi-iterative method (Golub and Varga, 1961)

Input: x_0

Initialize: $\omega_0 = 2$

$x_1 = x_0 - \frac{2}{L+\mu}\nabla f(x_0)$

for $t = 1, \dots$ **do**

$\omega_{t+1} = \left(1 - \frac{1}{4}\left(\frac{1-\kappa}{1+\kappa}\right)^2 \omega_t\right)^{-1} \quad x_{t+1} = x_t - \frac{2}{L+\mu}\omega_t \nabla f(x_t) + (\omega_t - 1)(x_t - x_{t-1})$

end

Theorem 3.B.1. *The iterates produced by the Chebyshev semi-iterative method verify*

$$x_t - x_* = \frac{T_t(\sigma_1^{[\mu,L]}(H))}{T_t(\sigma_1^{[\mu,L]}(0))}(x_0 - x_*) \quad \textit{for all} \quad t \in \mathbb{N}. \tag{3.34}$$

Furthermore, this method enjoys a worst-case rate of the form

$$\|x_t - x_*\| \leqslant \frac{1}{T_t(\sigma_1^{[\mu,L]}(0))}\|x_0 - x_*\| = O\left(\left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^t\right). \tag{3.35}$$

Proof. Consider first an algorithm whose iterates verify (3.34). Then using the Cauchy-Schwartz inequality and known bounds of Chebyshev polynomials, we can show the following rate

$$\begin{aligned}
\|x_t - x_*\| &\leqslant \frac{\sup_{\lambda\in[\mu,L]} |T_t(\sigma_1^{[\mu,L]}(\lambda))|}{T_t(\sigma_1^{[\mu,L]}(0))}\|x_0 - x_*\| \\
&= \frac{1}{T_t\left(\frac{1+\kappa}{1-\kappa}\right)}\|x_0 - x_*\| && \text{since} \sup_{x\in[-1,1]} |T_t(x)| = 1 \\
&\leqslant 2\left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^t \|x_0 - x_*\| && \text{since} T_t(x) \geqslant \frac{\left(x + \sqrt{x^2-1}\right)^t}{2}, \forall x \notin (-1, 1).
\end{aligned}$$

It remains to prove that Algorithm 5 is the one that achieves the property (3.34). Using the recursion verified by Chebyshev polynomials

$$T_{t+1}(x) = 2xT_t(x) - T_{t-1}(x), \tag{3.36}$$

we have

$$\begin{aligned}
x_{t+1} - x_* &= \frac{T_{t+1}(\sigma_1^{[\mu,L]}(H))}{T_{t+1}(\sigma_1^{[\mu,L]}(0))}(x_0 - x_*) \\
&= \frac{2\sigma_1^{[\mu,L]}(H)T_t(\sigma_1^{[\mu,L]}(H))(x_0 - x_*) - T_{t-1}(\sigma_1^{[\mu,L]}(H))(x_0 - x_*)}{T_{t+1}(\sigma_1^{[\mu,L]}(0))} \\
&= \frac{2\sigma_1^{[\mu,L]}(H)T_t(\sigma_1^{[\mu,L]}(0))}{T_{t+1}(\sigma_1^{[\mu,L]}(0))}(x_t - x_*) - \frac{T_{t-1}(\sigma_1^{[\mu,L]}(0))}{T_{t+1}(\sigma_1^{[\mu,L]}(0))}(x_{t-1} - x_*) \\
&= \frac{2\sigma_1^{[\mu,L]}(0)T_t(\sigma_1^{[\mu,L]}(0))}{T_{t+1}(\sigma_1^{[\mu,L]}(0))}\left(I - \frac{2}{L+\mu}H\right)(x_t - x_*) - \frac{T_{t-1}(\sigma_1^{[\mu,L]}(0))}{T_{t+1}(\sigma_1^{[\mu,L]}(0))}(x_{t-1} - x_*)\,.
\end{aligned}$$

Let's introduce $\omega_t \triangleq \frac{2\sigma_1^{[\mu,L]}(0)T_t(\sigma_1^{[\mu,L]}(0))}{T_{t+1}(\sigma_1^{[\mu,L]}(0))}$. Then $\omega_0 = 2$ and by Chebyshev recursion (Equation (3.36)), $\omega_t - 1 = \frac{T_{t-1}(\sigma_1^{[\mu,L]}(0))}{T_{t+1}(\sigma_1^{[\mu,L]}(0))}$. With this notation we can write the above identity more compactly as

$$\begin{aligned}
x_{t+1} - x_* &= \omega_t\left(I - \frac{2}{L+\mu}H\right)(x_t - x_*) - (\omega_t - 1)(x_{t-1} - x_*) \\
&= x_t - \frac{2}{L+\mu}\omega_t\nabla f(x_t) + (\omega_t - 1)(x_t - x_{t-1})\,.
\end{aligned}$$

It remains to find a recursion on ω_t to make its use tractable. Using one more time the Chebyshev recursion Equation (3.36),

$$\begin{aligned}
\omega_t^{-1} &= \frac{T_{t+1}(\sigma_1^{[\mu,L]}(0))}{2\sigma_1^{[\mu,L]}(0)T_t(\sigma_1^{[\mu,L]}(0))} \\
&= \frac{2\sigma_1^{[\mu,L]}(0)T_t(\sigma_1^{[\mu,L]}(0)) - T_{t-1}(\sigma_1^{[\mu,L]}(0))}{2\sigma_1^{[\mu,L]}(0)T_t(\sigma_1^{[\mu,L]}(0))} \\
&= 1 - \frac{1}{4\sigma_1^{[\mu,L]}(0)^2}\frac{2\sigma_1^{[\mu,L]}(0)T_{t-1}(\sigma_1^{[\mu,L]}(0))}{T_t(\sigma_1^{[\mu,L]}(0))} \\
&= 1 - \frac{1}{4\sigma_1^{[\mu,L]}(0)^2}\omega_{t-1},
\end{aligned}$$

which can finally be written as

$$\omega_{t+1} = \frac{1}{1 - \frac{1}{4}\left(\frac{1-\kappa}{1+\kappa}\right)^2\omega_t},$$

and we recognize the *Chebyshev semi-iterative method* described in Algorithm 5. ■

This method, unlike the Polyak Heavy-ball (PHB) method, uses a different step-size and momentum at each iteration. However, both are related, as taking the limit of ω_t as $t \to \infty$ in Algorithm 5 we obtain $\omega_\infty = 1 + m$ with $m = \left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^2$. This correspond to the parameters of PHB.

We note that this is only one way to construct a method that has the Chebsyshev polynomial as residual polynomial at every iteration. However, it is possible to construct a different update that have the Chebyshev polynomial at fixed iteration, see for instance (Young, 1953; Agarwal et al., 2021) for one such alterative that does not require momentum.

3.B.2 Polyak Heavy-ball method

Algorithm 6 Polyak Heavy-ball

Input: x_0

Set: $m = \left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^2$ and $h = \frac{2(1+m)}{L+\mu}$.

$x_1 = x_0 - \frac{h}{1+m}\nabla f(x_0)$

for $t = 1, \dots$ **do**

$\quad x_{t+1} = x_t - h\nabla f(x_t) + m(x_t - x_{t-1})$

end

Theorem 3.B.2. *The iterates of the Heavy-ball algorithm verify*

$$x_t - x_* = P_t(H)(x_0 - x_*) \quad \textit{for all} \quad t \in \mathbb{N},$$

with P_t defined as

$$P_t(\lambda) \triangleq (\sqrt{m})^t \left[\frac{2m}{1+m} T_t(\sigma_1^{[\mu,L]}(\lambda)) + \frac{1-m}{1+m} U_t(\sigma_1^{[\mu,L]}(\lambda))\right]. \tag{3.37}$$

Furthermore, this method enjoys a worst-case rate of the form

$$\|x_t - x_*\| = O\left(t\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^t\right). \tag{3.38}$$

Proof. From the update defined in Algorithm 6, we identify

$$\begin{aligned} P_0(\lambda) &= 1 \\ P_1(\lambda) &= 1 - \frac{h}{1+m}\lambda \\ P_{t+1}(\lambda) &= (1+m-h\lambda)P_t(\lambda) - mP_{t-1}(\lambda). \end{aligned}$$

Introducing $\tilde{P}_t \triangleq \frac{P_t}{(\sqrt{m})^t}$, we have

$$\begin{aligned} \tilde{P}_0(\lambda) &= 1 \\ \tilde{P}_1(\lambda) &= \frac{1+m-h\lambda}{(1+m)\sqrt{m}} = \frac{2}{1+m}\sigma_1^{[\mu,L]}(\lambda) \\ \tilde{P}_{t+1}(\lambda) &= \frac{(1+m-h\lambda)}{\sqrt{m}}\tilde{P}_t(\lambda) - \tilde{P}_{t-1}(\lambda) \\ &= 2\sigma_1^{[\mu,L]}(\lambda)\tilde{P}_t(\lambda) - \tilde{P}_{t-1}(\lambda). \end{aligned}$$

This is a second order recurrence, with 2 initializations. It allows us to identify uniquely the family

$$\tilde{P}_t(\lambda) = \frac{2m}{1+m} T_t(\sigma_1^{[\mu,L]}(\lambda)) + \frac{1-m}{1+m} U_t(\sigma_1^{[\mu,L]}(\lambda)). \tag{3.39}$$

where U_t denotes the Chebyshev polynomial of the second kind of degree t. While both T_t and U_t verify the same recursion as $\tilde{P}_t$ and $T_0 = U_0 = \tilde{P}_0 = 1$, the difference between T and U comes when $T_1(X) = X$ and $U_1(X) = 2X$. This is how $\tilde{P}_t$ ends being a linear combination of the T_t and U_t. Finally,

$$P_t(\lambda) = (\sqrt{m})^t \left[\frac{2m}{1+m} T_t(\sigma_1^{[\mu,L]}(\lambda)) + \frac{1-m}{1+m} U_t(\sigma_1^{[\mu,L]}(\lambda))\right]. \tag{3.40}$$

Since by definition $\sigma_1^{[\mu,L]}([\mu,L]) = [-1,1]$, $T_t(\sigma_1^{[\mu,L]}(\lambda)) \leqslant 1$ and $U_t(\sigma_1^{[\mu,L]}(\lambda)) \leqslant t+1, \forall t \in \mathbb{N}$. Hence, $\forall \lambda \in [\mu, L]$,

$$P_t(\lambda) \leqslant (\sqrt{m})^t \left[1 + \frac{1-m}{1+m} t\right] \leqslant (2\sqrt{\kappa}t + 1)\left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^t \tag{3.41}$$

and

$$\|x_t - x_*\| = O\left(t\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^t\right). \tag{3.42}$$

■

3.C Minimax Polynomials and Equioscillation Property

Section 3.B dealt with optimal methods when $\Lambda = [\mu, L]$. Those methods could be derived since the minimax polynomial (Definition 3.A.1) $Z_t^{[\mu,L]}$ is known.

In this section we consider the problem of finding minimax polynomials in a more general setting. We provide a characterization of the minimax polynomial defined in definition 3.A.1. For the sake of simplicity, we actually focus on the polynomial σ_t^Λ solution of (3.16). We can easily adapt the result to Z_t^Λ leveraging Remark 3.4.7. We prove the following theorem.

Theorem 3.C.1. *Let P_K be a degree K polynomial verifying $P_K(\Lambda) \subset [-1,1]$. Then P_K is the unique solution σ_K^Λ of equation (3.16) if and only if there exists a sorted family $(\lambda_i)_{i\in[\![0,K]\!]} \in \left(\overline{\Lambda}\right)^{K+1}$ (where $\overline{\Lambda}$ is the closure of Λ) such that $\forall i \in [\![0,K]\!], P_K(\lambda_i) = (-1)^i$.*

The following proof is technical and requires to introduce several new notations. Hence we first briefly describe the intuition before giving the actual complete proof.

($\Longleftarrow$): Assume P_K "oscillates" $K+1$ times between 1 and -1. Since P_K has a degree K, it is completely determined by its values on those $K+1$ points, using the Lagrange interpolation representation. We prove that P_K is optimal because any other polynomial Q_K, having different values on those $K+1$ points would achieve a smaller value $Q_K(0)$ at 0.

($\Longrightarrow$): We prove this by contradiction. We assume that P_K doesn't oscillate $K+1$ times between 1 and -1, and prove that $P_K(0)$ is not optimal. To do so, we build a small perturbation εQ_K such that $P_K + \varepsilon Q_K$ is a polynomial of degree K, which values on Λ are all in $[-1;1]$, and with an higher value at 0.

(Uniqueness) We reuse the Lagrange interpolation representation to justify that 2 optimal polynomials must "oscillate" on the same points, therefore are equal.

Proof. We prove successively both directions:

($\Longleftarrow$): *Assume $\exists \lambda_0 < \lambda_1 < \cdots < \lambda_K$ such that*

$$\forall i \in [\![0,K]\!], P_K(\lambda_i) = (-1)^i \quad \text{and} \quad P_K(\Lambda) \subset [-1,1]. \tag{3.43}$$

We aim to prove that P_K is the unique solution σ_K^Λ of equation (3.16), that is for any other polynomial Q_K of degree K verifying $Q_K(\Lambda) \subset [-1,1]$, $P_K(0) \geqslant Q_K(0)$.

We introduce such a polynomial Q_K of degree K and bounded in absolute value by 1 on Λ. Let's define, for all $i \in [\![0,\ K]\!]$,

$$v_i \triangleq Q_K(\lambda_i) \in [-1,1]. \tag{3.44}$$

These $K+1$ values characterize Q_K (of degree K), and we can decompose it over Lagrange interpolation polynomials. We have

$$Q_K = \sum_{i=0}^{K} v_i L_{\lambda_i} \quad \text{where} \quad L_{\lambda_i}(X) \triangleq \prod_{j \neq i} \frac{X - \lambda_j}{\lambda_i - \lambda_j} \,. \tag{3.45}$$

The value at 0 can be computed as

$$Q_K(0) = \sum_{i=0}^{K} v_i L_{\lambda_i}(0) = \sum_{i=0}^{K} v_i \prod_{j \neq i} \frac{\lambda_j}{\lambda_j - \lambda_i} \,. \tag{3.46}$$

Maximizing this linear function of $(v_i)_{i \in [\![0,\, K]\!]}$ over the ℓ_∞ ball $B_\infty(1) \triangleq \{(v_i)_{i \in [\![0,\, K]\!]}, \forall i, -1 \leqslant v_i \leqslant 1\}$ leads to, for $v^* \triangleq \arg\min_{v \in B_\infty(1)} \sum_{i=0}^{K} v_i \prod_{j \neq i} \frac{\lambda_j}{\lambda_j - \lambda_i}$,

$$v_i^* = \operatorname{sgn}\left(\prod_{j \neq i} \frac{\lambda_j}{\lambda_j - \lambda_i} \right) = (-1)^i. \tag{3.47}$$

where sgn is the sign function (which maps 0 to 0, $\mathbb{R}_{<0}$ to -1, and $\mathbb{R}_{>0}$ to 1). Finally,

$$P_K(0) \geqslant Q_K(0) \tag{3.48}$$

which concludes the proof.

($\Longrightarrow$): *Assume P_K alternates $s < K+1$ times between -1 and 1 on $\overline{\Lambda}$. We want to show that P_K is not optimal in the sense described above. To do so, we construct a perturbation of P_K that increases its value in 0 while still satisfying the constraint $P_K(\Lambda) \subset [-1, 1]$.*

Let's define

$$\lambda_0^{(1)} < \cdots < \lambda_0^{(\nu_0)} < \lambda_1^{(1)} < \cdots < \lambda_1^{(\nu_1)} < \cdots < \lambda_{s-1}^{(1)} < \cdots < \lambda_{s-1}^{(\nu_{s-1})} \tag{3.49}$$

such that

$$P_K(\lambda_i^{(j)}) = (-1)^i \quad \text{and} \quad \forall \lambda \in \overline{\Lambda}, \left(\exists (i,j) | \lambda = \lambda_i^{(j)} \text{ or } |P_K(\lambda)| < 1 \right). \tag{3.50}$$

In short, $\left(\lambda_i^{(j)}\right)_{(i,j)}$ describes all the extremal points of P_K in Λ. The indices change when the sign changes, while the exponents are used to express the possible consecutive repetitions of the same value (-1 or 1).

Set $(r_i)_{i \in [\![0,s]\!]}$ as any set of positive numbers satisfying:

$$0 < r_0 < \inf(\Lambda) < \lambda_0^{(1)} < \lambda_0^{(\nu_0)} < r_1 < \cdots < r_s < \lambda_{s-1}^{(1)} < \lambda_{s-1}^{(\nu_{s-1})} < \sup(\Lambda) < r_s. \tag{3.51}$$

By definition, each interval $[r_i, r_{i+1}]$, $i \in [\![0,\, s-1]\!]$, contains $\lambda_i^{(j)}$ for all j, but no other extremal points of P_K in $\overline{\Lambda}$. Hence, $P_K([r_i, r_{i+1}] \cap \overline{\Lambda})$ doesn't contain $(-1)^{i+1}$. Since, $\bigcup_{i<s, i \text{ even}} [r_i, r_{i+1}] \cap \overline{\Lambda}$ is compact, and by continuity of P_K, $P_K\left(\bigcup_{i<s, i \text{ even}} [r_i, r_{i+1}] \cap \overline{\Lambda}\right)$ is compact. Therefore,

$$\exists \varepsilon_{-1} > 0 | P_K \left(\bigcup_{i<s, i \text{ even}} [r_i, r_{i+1}] \cap \overline{\Lambda} \right) \subset [-1 + \varepsilon_{-1}, 1]. \tag{3.52}$$

Similarly, we obtain

$$\exists \varepsilon_{1} > 0 | P_K \left(\bigcup_{i<s, i \text{ odd}} [r_i, r_{i+1}] \cap \overline{\Lambda} \right) \subset [-1, 1 - \varepsilon_{1}]. \tag{3.53}$$

We are now equipped to build the aforementioned perturbation. Let

$$Q_K(X) \triangleq \prod_{i\in[\![0,s-1]\!]} (r_i - X). \tag{3.54}$$

Note that Q_K has a degree $s \leqslant K$ and satisfies

$$Q_K\left(\bigcup_{i<s, i \text{ even}} [r_i, r_{i+1}] \cap \overline{\Lambda}\right) \subset \mathbb{R}^- \quad \text{and} \quad Q_K\left(\bigcup_{i<s, i \text{ odd}} [r_i, r_{i+1}] \cap \overline{\Lambda}\right) \subset \mathbb{R}^+. \tag{3.55}$$

Moreover, those sets are compact , by continuity of Q_K, and consequently bounded. We can therefore choose a small enough $\varepsilon > 0$ such that

$$\varepsilon \min Q_K\left(\bigcup_{i<s, i \text{ even}} [r_i, r_{i+1}] \cap \overline{\Lambda}\right) > -\varepsilon_{-1} \quad \text{and} \quad \varepsilon \max Q_K\left(\bigcup_{i<s, i \text{ odd}} [r_i, r_{i+1}] \cap \overline{\Lambda}\right) < \varepsilon_1.$$

This leads to

$$(P_K + \varepsilon Q_K)(\Lambda) \subset [-1, 1]. \tag{3.56}$$

And as by definition, $Q_K(0) > 0$,

$$(P_K + \varepsilon Q_K)(0) > P_K(0). \tag{3.57}$$

Finally $(P_K + \varepsilon Q_K) \in \mathbb{R}_K[X]$. This proves that P_K is not optimal.

(Uniqueness) *Here, we prove that the optimal polynomial is necessarily unique. To do so, we introduce 2 optimal polynomials and show there must actually be identical.*

Let P_K an optimal polynomial and $(\lambda_i)_{i\in[\![0,K]\!]} \in \Lambda^{K+1}$ a family on which P_K interpolates alternatively 1 and -1. Let any other feasible polynomial Q_K and $(v_i)_{i\in[\![0,K]\!]}$ its values on $(\lambda_i)_{i\in[\![0,K]\!]}$:

$$Q_K = \sum_{i=0}^{K} v_i L_{\lambda_i}. \tag{3.58}$$

We have showed in the first point of this proof that the optimal values of v_i are alternatively 1 and -1. Consequently, if Q_K is also optimal,

$$Q_K(\lambda_i) = P_K(\lambda_i) \tag{3.59}$$

for all $i \in [\![0, K]\!]$, which characterizes polynomials of degree K. Then

$$Q_K = P_K \tag{3.60}$$

which shows that the optimal polynomial is unique. ■

We now give the formal statement and the proof of the second result, used in Subsection 3.4.4.

Theorem 3.C.2. *$T_n(\sigma_K)$ is optimal for all n if and only if σ_K verifies the equioscillation property (Definition 3.4.3, hence $\sigma_K = \sigma_K^\Lambda$ by Theorem 3.C.1) and $\overline{\Lambda} = \sigma_K^{-1}([-1, 1])$, i.e. the inverse mapping σ_K^{-1} transforms the interval $[-1, 1]$ into exactly $\overline{\Lambda}$.*

Before providing the proof, we first highlight that the property

$$\forall \lambda \in \Lambda, \sigma_K(\lambda) \in [-1, 1] \tag{3.61}$$

can equivalently be written

$$\overline{\Lambda} \subset \sigma_K^{-1}([-1,1]). \tag{3.62}$$

In other words, we are interested in the case where the reverse inclusion holds as well. This means that

$$\sigma_K(\lambda) \in [-1,1] \Rightarrow \lambda \in \overline{\Lambda}. \tag{3.63}$$

This corresponds to a stronger form of optimality of σ_K: it "fully" uses the available assumption related to Λ, in the sense that no point can be added to $\overline{\Lambda}$ without breaking the condition $\sigma_K(\Lambda) \subset [-1;1]$. For example, on Figure 3.3, σ_3^Λ *does not* satisfy the later property on the center graph, but satisfies it on the right graph. Here, we show that under this condition, $T_n(\sigma_K) = T_n(\sigma_K^\Lambda)$ is optimal (in the sense of (3.16)) for all $n \in \mathbb{N}$.

In Section 3.4.4, we give another view of this condition for $T_n(\sigma_K)$ to be optimal for all n. We can decompose Λ as the union of K intervals Λ_i such that they have disjoint interiors and they are all mapped to $[-1,1]$ by σ_K. Hence, σ_K maps Λ to $[-1,1]$ exactly K times.

Proof. From Theorem 3.C.1, $T_n(\sigma_K)$ is optimal for all n if and only if, for all n, there exist a sorted family of $(\lambda_i)_{i\in[\![0,nK]\!]}$ such that, $T_n(\sigma_K(\lambda_i)) = (-1)^i$.

Let $n \in \mathbb{N}$. We observe that by definition of T_n,

$$T_n(\sigma_K(\lambda)) = \pm 1 \quad \text{if and only if} \quad \exists j \in [\![0,n]\!] | \sigma_K(\lambda) = \cos\frac{j\pi}{n}. \tag{3.64}$$

We successively treat both directions: ($\Longleftarrow$) *we assume σ_K oscillates and $\overline{\Lambda} = \sigma_K^{-1}([-1,1])$. We aim to prove that $T_n(\sigma_K)$ is optimal for all $n \in \mathbb{N}$.*

By equioscillation property, we know that there exists λ_i' such that

$$\sigma_K(\lambda_i') = (-1)^i. \tag{3.65}$$

By the intermediate value theorem, we know that for any $i \in [\![0;K]\!]$, between the pair $\lambda_i', \lambda_{i+1}'$, there exist sorted $(\mu_i^j)_{ni<j<(n+1)i}$ such that for all $j \in [\![ni+1;(n+1)i-1]\!]$,

$$\sigma_K(\mu_i^j) = \cos\frac{j\pi}{n}. \tag{3.66}$$

We identify $\lambda_{ni} = \lambda_i'$ and $\lambda_j = \mu_{\lfloor j/n \rfloor}^j$ for all j not multiple of n. Then, for all $\ell \in [\![0,nK]\!]$:

$$T_n(\sigma_K(\lambda_\ell)) = (-1)^\ell. \tag{3.67}$$

By Theorem 3.C.1, we conclude that $T_n(\sigma_K)$ is optimal for all $n \in \mathbb{N}$.

($\Longrightarrow$) *We assume $T_n(\sigma_K)$ is optimal for all $n \in \mathbb{N}$. Clearly, σ_K is optimal ($n = 1$), and then equioscillates. We prove that moreover*

$$\overline{\Lambda} = \sigma_K^{-1}([-1,1]). \tag{3.68}$$

On the one hand, for any $j \in [\![0,n]\!]$, there exist at most K different λ that verifies $\sigma_K(\lambda) = \cos\frac{j\pi}{n}$ since σ_K has a degree K and is not constant. Therefore, there exist at most $(n+1)K$ different λ such that $\exists j \in [\![0,n]\!] | \sigma_K(\lambda) = \cos\frac{j\pi}{n}$, and by Eq.(3.64), there thus exist at most $(n+1)K$ different λ such that $T_n(\sigma_K(\lambda) = \pm 1$.

On the other hand, the optimality of $T_n(\sigma_K)$ implies the existence of *at least* $nK+1$ such λ in $\overline{\Lambda}$.

Hence all but at most $K-1$ values λ such that $\sigma_K(\lambda) \in \{\cos\frac{j\pi}{n}, j \in [\![0,n]\!]\}$ belong to $\overline{\Lambda}$.

This holds for all n. Therefore for n large enough, all x such that $\sigma(x) \in [-1,1]$ are as close as we want to some $\lambda \in \overline{\Lambda}$. Since $\overline{\Lambda}$ is a closed set, then all x such that $\sigma(x) \in [-1,1]$ are actually in $\overline{\Lambda}$.

We conclude

$$\overline{\Lambda} \supset \sigma_K^{-1}([-1,1]). \tag{3.69}$$

■

3.D Cyclical step-sizes

In this appendix, we provide an analysis of momentum methods with cyclical step-sizes and derive some non-asymptotically optimal variants.

3.D.1 Derivation of optimal algorithm with $K = 2$ alternating step-sizes

In this section, we consider the case where Λ is the union of 2 intervals of same size, as described in Section 3.3.

We start by introducing the following algorithm, and we will prove later that this algorithm is optimal (Theorem 3.D.1)

Algorithm 7 Optimal momentum method with alternating step-sizes ($K = 2$)

Input: Initialization x_0, $\boldsymbol{\mu_1} < \boldsymbol{L_1} < \boldsymbol{\mu_2} < \boldsymbol{L_2}$ (where $\boldsymbol{L_1} - \boldsymbol{\mu_1} = \boldsymbol{L_2} - \boldsymbol{\mu_2}$)

Set: $\rho = \frac{\boldsymbol{L_2}+\boldsymbol{\mu_1}}{\boldsymbol{L_2}-\boldsymbol{\mu_1}}$, $R = \frac{\boldsymbol{\mu_2}-\boldsymbol{L_1}}{\boldsymbol{L_2}-\boldsymbol{\mu_1}}$, $c = \sqrt{\frac{\rho^2-R^2}{1-R^2}}$

$\omega_0 = 2$

$x_1 = x_0 - \frac{1}{\boldsymbol{L_1}}\nabla f(x_0)$

for $t = 1, 2, \ldots$ **do**

$$\begin{aligned} \omega_t &= \left(1 - \frac{1}{4c^2}\omega_{t-1}\right)^{-1} \\ h_t &= \tfrac{\omega_t}{\boldsymbol{L_1}} \quad \text{(if } t \text{ is even)}, \qquad h_t = \tfrac{\omega_t}{\boldsymbol{\mu_2}} \quad \text{(if } t \text{ is odd)} \\ x_{t+1} &= x_t - h_t\nabla f(x_t) + (\omega_t - 1)(x_t - x_{t-1}) \end{aligned}$$

end

Theorem 3.D.1. *Let $f \in \mathcal{C}_\Lambda$ and $x_0 \in \mathbb{R}^d$. Assume Λ defined as in (3.3). The iterates of Algorithm 7 verifies the condition*

$$x_{2n} - x_* = \frac{T_n(\sigma_2^\Lambda(H))}{T_n(\sigma_2^\Lambda(0))}(x_0 - x_*) \tag{3.70}$$

and this is the optimal convergence rate over $\mathcal{C}_\Lambda$.

Proof.

We begin by showing the optimality of the algorithm. Using Proposition 3.D.2, the polynomial in (3.70) equioscillates on Λ, which makes it optimal by Theorem 3.C.1. By optimal, this means this is the optimal convergence rate any first-order algorithm can reach

(See (3.11)). We invite the reader to read Appendix 3.D.3, where we study in details the properties of the alternating steps sizes strategy (i.e., $K = 2$).

As in Appendix 3.B.1, we derive here the constructive approach that leads us to this algorithm.

We now start showing that the iterates of Algorithm 7 follow (3.70). From equation (3.70), projecting onto the eigenspace of eigenvalue λ,

$$x_{2n} - x_* = \frac{T_n(\sigma_2^\Lambda(\lambda))}{T_n(\sigma_2^\Lambda(0))}(x_0 - x_*). \tag{3.71}$$

Then, we find a recursion definition for the subsequence $(x_{2n})_{n\in\mathbb{N}}$. Let $n \geqslant 1$.

$$\begin{aligned} x_{2(n+1)} - x_* &= \frac{T_{n+1}(\sigma_2^\Lambda(\lambda))}{T_{n+1}(\sigma_2^\Lambda(0))}(x_0 - x_*), && (3.72)\\ &= \frac{2\,\sigma_2^\Lambda(\lambda)\,T_n(\sigma_2^\Lambda(\lambda)) - T_{n-1}(\sigma_2^\Lambda(\lambda))}{T_{n+1}(\sigma_2^\Lambda(0))}(x_0 - x_*), && (3.73)\\ &= \frac{2\,\sigma_2^\Lambda(\lambda)\,T_n(\sigma_2^\Lambda(0))}{T_{n+1}(\sigma_2^\Lambda(0))}(x_{2n} - x_*) - \frac{T_{n-1}(\sigma_2^\Lambda(0))}{T_{n+1}(\sigma_2^\Lambda(0))}(x_{2(n-1)} - x_*). && (3.74) \end{aligned}$$

Note that if $\sigma_2^\Lambda(\lambda)$ were a degree 1 polynomial in λ, then we would recognize a momentum update. Here, $\sigma_2^\Lambda(\lambda)$ is actually a degree 2 polynomial in λ. We will then try to identify 2 steps of momentum. From here, let

$$c \triangleq \frac{1}{2}\left(\left(\sigma_K(0) + \sqrt{\sigma_K(0)^2 - 1}\right)^{1/2} + \left(\sigma_K(0) - \sqrt{\sigma_K(0)^2 - 1}\right)^{1/2}\right) = \sqrt{\frac{\sigma_K(0)+1}{2}} \tag{3.75}$$

be the unique positive real number c verifying $T_2(c) = 2c^2 - 1 = \sigma_K(0)$. We end up with

$$x_{2(n+1)} - x_* = \frac{2\,\sigma_2^\Lambda(\lambda)\,T_{2n}(c)}{T_{2(n+1)}(c)}(x_{2n} - x_*) - \frac{T_{2(n-1)}(c)}{T_{2(n+1)}(c)}(x_{2(n-1)} - x_*). \tag{3.76}$$

Note, the above equation suggests to introduce the sequence $z_l \triangleq T_l(c)(x_l - x_*)$. Indeed, the above equality simplifies

$$z_{2(n+1)} = 2\,\sigma_2^\Lambda(\lambda)\,z_{2n} - z_{2(n-1)}. \tag{3.77}$$

Let's look for two steps of the cyclical Heavy-ball method that are together equivalent to (3.76). We look for an algorithm of the form

$$\forall n \geqslant 0, x_{n+1} = x_n - h_n\nabla f(x_n) + \frac{T_{n-1}(c)}{T_{n+1}(c)}\left(x_n - x_{n-1}\right), \tag{3.78}$$

i.e, projecting again onto the eigenspace of eigenvalue λ, we obtain

$$\forall n \geqslant 0, x_{n+1} - x_* = \left(1 + \frac{T_{n-1}(c)}{T_{n+1}(c)} - h_n\lambda\right)(x_n - x_*) - \frac{T_{n-1}(c)}{T_{n+1}(c)}\left(x_{n-1} - x_*\right). \tag{3.79}$$

Here we introduce the notation

$$\omega_l \triangleq \left(1 + \frac{T_{l-1}(c)}{T_{l+1}(c)}\right) = 2c\frac{T_l(c)}{T_{l+1}(c)}, \tag{3.80}$$

and the change of variable

$$\tilde{h_l} \triangleq \frac{h_l}{\omega_l}. \tag{3.81}$$

We rewrite (3.79) in terms of the sequence z and using the sequence $\tilde{h}$,

$$\begin{aligned}\forall n \geqslant 0, z_{n+1} &= T_{n+1}(c)\left(1 + \frac{T_{n-1}(c)}{T_{n+1}(c)} - h_n\lambda\right)(x_n - x_*) - z_{n-1} &(3.82)\\ &= \left(2cT_n(c)(1-\tilde{h}_n\lambda)\right)(x_n - x_*) - z_{n-1} &(3.83)\\ &= \left(2c(1-\tilde{h}_n\lambda)\right)z_n - z_{n-1}. &(3.84)\end{aligned}$$

We now need to find the right sequence $\tilde{h}_n$ such that we recover equation (3.77). Combining the 2 following

$$\begin{aligned}z_{2n+1} &= \left(2c(1-\tilde{h}_{2n}\lambda)\right)z_{2n} - z_{2n-1} &(3.85)\\ z_{2n+2} &= \left(2c(1-\tilde{h}_{2n+1}\lambda)\right)z_{2n+1} - z_{2n} &(3.86)\end{aligned}$$

by isolating the odd index in the second equation and plugging it in the first one, we get

$$z_{2n+2} = \left(4c^2(1-\tilde{h}_{2n}\lambda)(1-\tilde{h}_{2n+1}\lambda) - 1 - \frac{2c(1-\tilde{h}_{2n+1}\lambda)}{2c(1-\tilde{h}_{2n-1}\lambda)}\right)z_{2n} - \frac{2c(1-\tilde{h}_{2n+1}\lambda)}{2c(1-\tilde{h}_{2n-1}\lambda)}z_{2n-2}. \tag{3.87}$$

We need to identify

$$\begin{aligned}2\sigma_2^\Lambda(\lambda) &= 4c^2(1-\tilde{h}_{2n}\lambda)(1-\tilde{h}_{2n+1}\lambda) - 1 - \frac{2c(1-\tilde{h}_{2n+1}\lambda)}{2c(1-\tilde{h}_{2n-1}\lambda)}, &(3.88)\\ 1 &= \frac{2c(1-\tilde{h}_{2n+1}\lambda)}{2c(1-\tilde{h}_{2n-1}\lambda)}. &(3.89)\end{aligned}$$

Hence, we conclude from the second equation that $\tilde{h}_{2n+1} = \tilde{h}_{2n-1} = \tilde{h}_1$ is independent of n. And the first equation then becomes

$$2\sigma_2^\Lambda(\lambda) = 4c^2(1-\tilde{h}_{2n}\lambda)(1-\tilde{h}_1\lambda) - 2 \tag{3.90}$$

leading also to $\tilde{h}_{2n}$ independent of n. We observe an alternating strategy of the "pseudo-step-sizes" $\tilde{h}_0$ and $\tilde{h}_1$. Finally, we must fix them to

$$\sigma_2^\Lambda(\lambda) = 2c^2(1-\tilde{h}_0\lambda)(1-\tilde{h}_1\lambda) - 1. \tag{3.91}$$

Note this is possible because the equation above is valid for $\lambda = 0$ for any choice of $\tilde{h}_0$ and $\tilde{h}_1$ and the polynomial $\sigma_2^\Lambda + 1$ can be defined by its value in 0 and its roots that are exactly $\frac{1}{\tilde{h}_0}$ and $\frac{1}{\tilde{h}_1}$. And from (3.155), those values are $\boldsymbol{\mu}_2$ and $\boldsymbol{L}_1$, which gives the values $\tilde{h}_0 = \frac{1}{\boldsymbol{L}_1}$ and $\tilde{h}_1 = \frac{1}{\boldsymbol{\mu}_2}$.

We now sum up what we have so far. Setting c, $\tilde{h}_0$ and $\tilde{h}_1$ as described above, the iterations

$$\forall n \geqslant 1, x_{n+1} = x_n - \left(1 + \frac{T_{n-1}(c)}{T_{n+1}(c)}\right)\tilde{h}_{\mathrm{mod}(n,2)}\nabla f(x_n) + \frac{T_{n-1}(c)}{T_{n+1}(c)}(x_n - x_{n-1}) \tag{3.92}$$

lead to the recursion (3.77).

Let define $x_1 = x_0 - \tilde{h}_0 \nabla f(x_0)$, and from the above

$$\begin{aligned}
x_2 &= x_1 - \left(1 + \frac{1}{2c^2-1}\right) \tilde{h}_1 \lambda (x_1 - x_*) + \frac{1}{2c^2-1}(x_1 - x_0) && (3.93)\\
x_2 - x_* &= \frac{2c^2}{\sigma_2^\Lambda(0)} \left(1 - \tilde{h}_1\lambda\right)(x_1 - x_*) - \frac{1}{\sigma_2^\Lambda(0)}(x_0 - x_*) && (3.94)\\
&= \frac{2c^2}{\sigma_2^\Lambda(0)} \left(1 - \tilde{h}_1\lambda\right)\left(1 - \tilde{h}_0\lambda\right)(x_0 - x_*) - \frac{1}{\sigma_2^\Lambda(0)}(x_0 - x_*) && (3.95)\\
&= \frac{\sigma_2^\Lambda(\lambda)}{\sigma_2^\Lambda(0)}(x_0 - x_*) && (3.96)\\
z_2 &= \sigma_2^\Lambda(\lambda). && (3.97)
\end{aligned}$$

Finally, the sequence z_{2n} is defined by

$$\begin{aligned}
z_0 &= 1, && (3.98)\\
z_1 &= \sigma_2^\Lambda(\lambda), && (3.99)\\
z_{2(n+1)} &= 2\,\sigma_2^\Lambda(\lambda)\, z_{2n} - z_{2(n-1)}. && (3.100)
\end{aligned}$$

which defines exactly $T_n(\sigma_2^\Lambda(\lambda))$. We conclude $x_{2n} - x_* = \frac{T_n(\sigma_2^\Lambda(\lambda))}{T_n(\sigma_2^\Lambda(0))}(x_0 - x_*)$.

We sum up the algorithm used to reach the above equality:

$$\begin{aligned}
x_1 &= x_0 - \tilde{h}_0 \nabla f(x_0), && (3.101)\\
\forall n \geqslant 0, x_{n+1} &= x_n - \omega_n \tilde{h}_n \nabla f(x_n) + (\omega_n - 1)\,(x_n - x_{n-1})\ . && (3.102)
\end{aligned}$$

with $\omega_n = \left(1 + \frac{T_{n-1}(c)}{T_{n+1}(c)}\right) = \frac{2cT_n(c)}{T_{n+1}(c)}$. Note the recursion

$$\begin{aligned}
\omega_n^{-1} &= \frac{T_{n+1}(c)}{2cT_n(c)} && (3.103)\\
&= \frac{2cT_n(c) - T_{n-1}(c)}{2cT_n(c)} && (3.104)\\
&= 1 - \frac{T_{n-1}(c)}{2cT_n(c)} && (3.105)\\
&= 1 - \frac{1}{4c^2}\frac{2cT_{n-1}(c)}{T_n(c)} && (3.106)\\
&= 1 - \frac{1}{4c^2}\omega_{n-1}. && (3.107)
\end{aligned}$$

Finally, the sequence ω can be computed online using the recursion

$$\omega_n = \frac{1}{1 - \frac{1}{4c^2}\omega_{n-1}} \tag{3.108}$$

with $\omega_0 = 2$. ■

In this appendix, as well as in Appendix 3.B, we end up with some equality of the form

$$\|x_t - x_*\| = \frac{T_n(\sigma_K(H))}{T_n(\sigma_K(0))}\|x_0 - x_*\|\,. \tag{3.109}$$

The next theorem explains how to derive the rate factor from it.

Proposition 3.4.6. *For a given σ_K such that $\sup_{\lambda\in\Lambda}|\sigma_K(\lambda)| = 1$, the asymptotic rate factor τ^{σ_K} of the method associated to the polynomial (3.14) is*

$$1 - \tau^{\sigma_K} = \lim_{t\to\infty} \sqrt[t]{\sup_{\lambda\in\Lambda}|P_t(\lambda;\sigma_K)|} = \left(\sigma_0 - \sqrt{\sigma_0^2 - 1}\right)^{\frac{1}{K}}, \quad \text{with } \sigma_0 \triangleq \sigma_K(0)\,. \tag{3.15}$$

Proof. We observe that the rate factor of the method is upper bounded by

$$\sqrt[t]{\sup_{\lambda\in\Lambda}|Z_t^\Lambda(\lambda)|} = \sqrt[t]{\sup_{\lambda\in\Lambda}\left|\frac{T_{t/K}\left(\sigma_K^\Lambda(\lambda)\right)}{T_{t/K}\left(\sigma_0\right)}\right|} = \sqrt[t]{\frac{1}{|T_{t/K}\left(\sigma_0\right)|}} \quad \text{if } \sup_{\lambda\in\Lambda}|\sigma_K(\lambda)| = 1. \tag{3.110}$$

Since $\sigma_0 > 1$, and by using the explicit formula of Chebyshev polynomials, we have that

$$T_{t/K}\left(\sigma_0\right) = \frac{\left(\sigma_0 + \sqrt{\sigma_0^2-1}\right)^{t/K} + \left(\sigma_0 - \sqrt{\sigma_0^2-1}\right)^{t/K}}{2} \underset{t\to\infty}{\sim} \frac{\left(\sigma_0 + \sqrt{\sigma_0^2-1}\right)^{t/K}}{2}. \tag{3.111}$$

Taking the limit gives

$$\lim_{t\to\infty}\sqrt[t]{\frac{1}{|T_{t/K}\left(\sigma_0\right)|}} = \left(\frac{1}{\sigma_0 + \sqrt{\sigma_0^2-1}}\right)^{\frac{1}{K}} = \left(\sigma_0 - \sqrt{\sigma_0^2-1}\right)^{\frac{1}{K}}. \tag{3.112}$$

■

3.D.2 Derivation of Heavy-ball with K step-sizes cycle

In this section, we consider heavy-ball algorithm with a cycle of K different step-sizes. For convenience, we restate Algorithm 2.

Algorithm 8

Cyclical Heavy-ball $\mathrm{HB}_K(h_0, \ldots, h_{K-1}; m)$

Input: Initialization x_0, momentum $m \in (0,1)$, step-sizes $\{h_0, \ldots, h_{K-1}\}$

$x_1 = x_0 - \frac{h_0}{1+m}\nabla f(x_0)$

for $t = 1, 2, \ldots$ **do**

$$x_{t+1} = x_t - h_{\mathrm{mod}(t,K)}\nabla f(x_t) + m(x_t - x_{t-1})$$

end

We first recall the convergence theorem 3.4.8 stated in Section 3.4.3.

Theorem 3.4.8. *With an arbitrary momentum m and an arbitrary sequence of step-sizes $\{h_i\}$, the worst-case convergence rate $1-\tau$ of Algorithm 2 on $\mathcal{C}_\Lambda$ is*

$$\begin{cases} \sqrt{m} & \text{if } \sigma_* \leqslant 1 \\ \sqrt{m}\left(\sigma_* + \sqrt{\sigma_*^2-1}\right)^{K-1} & \text{if } \sigma_* \in \left(1,\ \frac{1+m^K}{2\left(\sqrt{m}\right)^K}\right) \\ \geqslant 1 \text{ (no convergence)} & \text{if } \sigma_* \geqslant \frac{1+m^K}{2\left(\sqrt{m}\right)^K}, \end{cases} \tag{3.19}$$

where $\sigma_* \triangleq \sup_{\lambda\in\Lambda}|\sigma(\lambda;\{h_i\},m)|$*,* $\sigma(\lambda;\{h_i\},m)$ *is the* K*-degree polynomial*

$$\sigma(\lambda;\{h_i\},m) \triangleq \frac{1}{2}\mathrm{Tr}\left(M_1 M_2 \dots M_K\right), \tag{3.20}$$

and $M_i = \begin{bmatrix} \frac{1+m-h_{K-i}\lambda}{\sqrt{m}} & -1 \\ 1 & 0 \end{bmatrix}$.

Proof. Note a first trick. Let's define $x_{-1} \triangleq x_0 - \frac{h_0}{1+m}\nabla f(x_0)$. This way, $x_{t+1} = x_t - h_{\mathrm{mod}(t,K)}\nabla f(x_t) + m(x_t - x_{t-1})$ holds for any $t \geqslant 0$ (including $t = 0$).

Now, let's introduce the polynomials P_t defined by Proposition 3.4.1 as $x_t - x_* = P_t(H)(x_0 - x_*)$. From now, in order to highlight the K-cyclic behavior, we introduce the indexation $t = nK + r$, with $r \in [\![0, K-1]\!]$.

We verify the following:

$$P_{-1}(\lambda) = 1 - \frac{h_0\lambda}{1+m}, \tag{3.113}$$

$$P_0(\lambda) = 1, \tag{3.114}$$

$$\forall n \geqslant 0, r \in [\![0,K-1]\!], \quad P_{nK+r+1}(\lambda) = (1+m-h_r\lambda)P_{nK+r}(\lambda) - mP_{nK+r-1}(\lambda). \tag{3.115}$$

In order to get rid of the last occurrence of m in equation above, we introduce $\tilde{P}_t(\lambda) \triangleq \frac{1}{(\sqrt{m})^t}P_t(\lambda)$.

This way, the above can be written

$$\tilde{P}_{-1}(\lambda) = \sqrt{m}\left(1 - \frac{h_0\lambda}{1+m}\right) = \frac{2m}{1+m}\sigma_0(\lambda), \tag{3.116}$$

$$\tilde{P}_0(\lambda) = 1, \tag{3.117}$$

$$\forall n \geqslant 0, r \in [\![0,K-1]\!], \quad \tilde{P}_{nK+r+1}(\lambda) = \frac{1+m-h_r\lambda}{\sqrt{m}}\tilde{P}_{nK+r}(\lambda) - \tilde{P}_{nK+r-1}(\lambda). \tag{3.118}$$

In the following, we want to determine a formulation for the polynomials $\tilde{P}_{nK}$. In order to do so, we introduce the following operator:

$$A(\lambda) \triangleq \begin{pmatrix} \frac{1+m-h_{K-1}\lambda}{\sqrt{m}} & -1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} \frac{1+m-h_0\lambda}{\sqrt{m}} & -1 \\ 1 & 0 \end{pmatrix} \triangleq \begin{pmatrix} a(\lambda) & b(\lambda) \\ c(\lambda) & d(\lambda) \end{pmatrix} \tag{3.119}$$

as well as the scalar valued function

$$\sigma(\lambda;\{h_i\},m) \triangleq \frac{1}{2}\mathrm{Tr}(A(\lambda)). \tag{3.120}$$

This operator comes naturally in

$$\begin{pmatrix} \tilde{P}_{(n+1)K}(\lambda) \\ \tilde{P}_{(n+1)K-1}(\lambda) \end{pmatrix} = \begin{pmatrix} \frac{1+m-h_{K-1}\lambda}{\sqrt{m}} & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \tilde{P}_{(n+1)K-1}(\lambda) \\ \tilde{P}_{(n+1)K-2}(\lambda) \end{pmatrix} \tag{3.121}$$

$$= \begin{pmatrix} \frac{1+m-h_{K-1}\lambda}{\sqrt{m}} & -1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} \frac{1+m-h_0\lambda}{\sqrt{m}} & -1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \tilde{P}_{nK}(\lambda) \\ \tilde{P}_{nK-1}(\lambda) \end{pmatrix} \tag{3.122}$$

$$= A(\lambda)\begin{pmatrix} \tilde{P}_{nK}(\lambda) \\ \tilde{P}_{nK-1}(\lambda) \end{pmatrix}. \tag{3.123}$$

Looking K steps at a time makes the analysis much easier as the process applying K steps is then homogeneous (we apply A and A doesn't depend on the index of the iterate).

$$\tilde{P}_{(n+1)K}(\lambda) = a(\lambda)\tilde{P}_{nK}(\lambda) + b(\lambda)\tilde{P}_{nK-1}(\lambda), \tag{3.124}$$

$$\tilde{P}_{(n+1)K-1}(\lambda) = c(\lambda)\tilde{P}_{nK}(\lambda) + d(\lambda)\tilde{P}_{nK-1}(\lambda). \tag{3.125}$$

Combining the two above equations (First one with incremented $n + b(\lambda)$ times the second one - $d(\lambda)$ times the first one) leads to

$$\tilde{P}_{(n+2)K}(\lambda) = (a(\lambda) + d(\lambda))\tilde{P}_{(n+1)K}(\lambda) - (a(\lambda)d(\lambda) - b(\lambda)c(\lambda))\tilde{P}_{nK}(\lambda) \tag{3.126}$$

$$= 2\sigma(\lambda; \{h_i\}, m)\tilde{P}_{(n+1)K}(\lambda) - \tilde{P}_{nK}(\lambda) \tag{3.127}$$

where the second inequality is deduced after we recognize

$$a(\lambda) + d(\lambda) = \mathrm{Tr}(A(\lambda)) = 2\sigma(\lambda; \{h_i\}, m) \tag{3.128}$$

and

$$a(\lambda)d(\lambda) - b(\lambda)c(\lambda) = \mathrm{Det}(A(\lambda)) = 1 \tag{3.129}$$

($A(\lambda)$ is the product of matrices of determinant 1).

In equation (3.127) we recognize the recursion verified by e.g. $(T_n(\sigma(\lambda; \{h_i\}, m)))_{n\in\mathbb{N}}$, or $(U_n(\sigma(\lambda; \{h_i\}, m)))_{n\in\mathbb{N}}$, where T_n (resp. U_n) denotes the first (resp. second) type Chebyshev polynomial of degree n.

Moreover we verify the initialization

$$\tilde{P}_0(\lambda) = 1, \tag{3.130}$$

$$\tilde{P}_K(\lambda) = a(\lambda)\tilde{P}_0(\lambda) + b(\lambda)\tilde{P}_{-1}(\lambda) \tag{3.131}$$

$$= a(\lambda) + b(\lambda)\frac{m}{1+m}\frac{1+m-h_0\lambda}{\sqrt{m}}. \tag{3.132}$$

We also notice that

$$U_n(\sigma(\lambda; \{h_i\}, m)) + \left(b(\lambda)\frac{m}{1+m}\frac{1+m-h_0\lambda}{\sqrt{m}} - d(\lambda)\right) U_{n-1}(\sigma(\lambda; \{h_i\}, m)) \tag{3.133}$$

verifies the same recursion of order 2 than $\tilde{P}_{Kn}$ as well as the same 2 initial terms.

Finally, we conclude

$$\tilde{P}_{nK}(\lambda) = U_n(\sigma(\lambda; \{h_i\}, m)) + \left(b(\lambda)\frac{m}{1+m}\frac{1+m-h_0\lambda}{\sqrt{m}} - d(\lambda)\right) U_{n-1}(\sigma(\lambda; \{h_i\}, m)) \tag{3.134}$$

and

$$P_{nK}(\lambda) = \left(\sqrt{m}\right)^{nK} \tilde{P}_{nK}(\lambda)\,. \tag{3.135}$$

Now we have the full expression of the polynomials associated to algorithm 2. Then we can study it's convergence rate.

Note for any $r \in [\![0, K-1]\!]$, we can have a similar expression of the form

$$P_{nK+r}(\lambda) = \left(\sqrt{m}\right)^{nK} \left(Q_r^1(\lambda)U_n(\sigma(\lambda; \{h_i\}, m)) + Q_r^2(\lambda)U_{n-1}(\sigma(\lambda; \{h_i\}, m))\right) \tag{3.136}$$

with Q_r^1 and Q_r^2 some fixed polynomials. This is the consequence of the fact that all sequences $\tilde{P}_{nK+r}(\lambda)$ verify the same recursion formula. Only initialization are different.

In order to study the factor rate of this algorithm, let's first introduce M an upper bound of all the $|Q_r^i|$. For instance, let M defined as follow.

$$M = \max_{r\in[\![0,K-1]\!],i\in\{1,2\}} \sup_{\lambda\in\Lambda} |Q_r^i(\lambda)|. \tag{3.137}$$

Then,

$$\begin{aligned} \|x_t - x_*\| &\leqslant \sup_{\lambda\in\Lambda} |P_t(\lambda)| \|x_0 - x_*\| && (3.138)\\ &\leqslant M\left(\sqrt{m}\right)^t \left(\sup_{\lambda\in\Lambda} |U_n(\sigma(\lambda;\{h_i\},m))| + \sup_{\lambda\in\Lambda} |U_{n-1}(\sigma(\lambda;\{h_i\},m))| \right) \|x_0 - x_*\|, && (3.139) \end{aligned}$$

with $n = \lfloor \frac{t}{K} \rfloor$.

Set $\sigma_{\sup} \triangleq \sup_{\lambda\in\Lambda} |\sigma(\lambda;\{h_i\},m)|$. The worst-case rate verifies

$$\text{If } \sigma_{\sup} \leqslant 1, \text{ then } r_t \leqslant M\left(\sqrt{m}\right)^t (n+1+n) = O\left(t\left(\sqrt{m}\right)^t\right). \tag{3.140}$$

$$\text{If } \sigma_{\sup} > 1, \text{ then } r_t = O\left(\left(\sqrt{m}\right)^t \left(\sigma_{\sup} + \sqrt{\sigma_{\sup}^2 - 1}\right)^n\right). \tag{3.141}$$

The first case analysis comes from the fact that U_n is bounded by $n+1$ on $[-1,1]$, while the second cases analysis comes from the fact that $U_n(x)$ grows exponentially fast in n at a rate $x + \sqrt{x^2-1}$ when x is outside of $[-1,1]$.

Then the factor rate verifies

$$\text{If } \sigma_{\sup} \leqslant 1, 1 - \tau = \sqrt{m}. \tag{3.142}$$

$$\text{If } \sigma_{\sup} > 1, 1 - \tau = \sqrt{m}\left(\sigma_{\sup} + \sqrt{\sigma_{\sup}^2 - 1}\right)^{1/K}. \tag{3.143}$$

It remains to notice that $\sqrt{m}\left(\sigma_{\sup} + \sqrt{\sigma_{\sup}^2 - 1}\right)^{1/K} < 1$ is equivalent to $\sigma_{\sup} < \frac{1+m^k}{2\left(\sqrt{m}\right)^k}$. ■

From this factor rate analysis, we can state Proposition 3.4.9 of Section 3.4.3.

Proposition 3.4.9. *Let $\sigma(\lambda;\{h_i\},m)$ be the polynomial defined by (3.20), and σ_K^Λ be the optimal link function of degree K defined by (3.16). If the momentum m and the sequence of step-sizes $\{h_i\}$ satisfy*

$$\sigma(\lambda;\{h_i\},m) = \sigma_K^\Lambda(\lambda), \tag{3.21}$$

*then **1)** the parameters are optimal, in the sense that they minimize the asymptotic rate factor from Theorem 3.4.8, **2)** the optimal momentum parameter is*

$$m = \left(\sigma_0 - \sqrt{\sigma_0^2 - 1}\right)^{2/K}, \quad \textit{where } \sigma_0 = \sigma_K^\Lambda(0), \tag{3.22}$$

***3)** the iterates from Algo. 4 with parameters $\{h_i\}$ and m form a polynomial with recurrence (3.18), and **4)** Algorithm 4 achieves the worst-case rate $r_t^{\text{Alg. 3}}$ and the asymptotic rate factor $1 - \tau^{\text{Alg. 3}}$*

$$\begin{aligned} r_t^{\text{Alg. 3}} &= O\left(t\left(\sigma_0 - \sqrt{\sigma_0^2 - 1}\right)^{t/K}\right), && (3.23)\\ 1 - \tau^{\text{Alg. 3}} &= \left(\sigma_0 - \sqrt{\sigma_0^2 - 1}\right)^{1/K}. \end{aligned}$$

Proof. For now we don't assume assumption 3.21 yet. Set $\sigma_0 \triangleq \sigma(0;\{h_i\},m)$. Then, by definition (3.20) of $\sigma(\lambda;\{h_i\},m)$,

$$\sigma_0 = \frac{1}{2}\mathrm{Tr}\left(\begin{bmatrix}\frac{1+m}{\sqrt{m}} & -1\\ 1 & 0\end{bmatrix}^K\right) = T_K\left(\frac{1+m}{2\sqrt{m}}\right) = \frac{1+m^K}{2\left(\sqrt{m}\right)^K}. \tag{3.144}$$

Hence, reversing this equality,

$$\sqrt{m} = \left(\sigma_0 - \sqrt{\sigma_0^2-1}\right)^{\frac{1}{K}}. \tag{3.145}$$

From Theorem 3.4.8, we therefore know

$$\text{If } \sigma_{\sup} \leqslant 1,\ 1-\tau = \left(\sigma_0 - \sqrt{\sigma_0^2-1}\right)^{\frac{1}{K}}. \tag{3.146}$$

$$\text{If } \sigma_{\sup} > 1,\ 1-\tau = \left(\sigma_0 - \sqrt{\sigma_0^2-1}\right)^{\frac{1}{K}}\left(\sigma_{\sup} + \sqrt{\sigma_{\sup}^2-1}\right)^{1/K}. \tag{3.147}$$

But, one can check that

$$\left(\sigma_0 - \sqrt{\sigma_0^2-1}\right)^{\frac{1}{K}}\left(\sigma_{\sup} + \sqrt{\sigma_{\sup}^2-1}\right)^{1/K} \geqslant \left(\frac{\sigma_0}{\sigma_{\sup}} - \sqrt{\left(\frac{\sigma_0}{\sigma_{\sup}}\right)^2-1}\right)^{\frac{1}{K}} \tag{3.148}$$

which shows that a tuning generating the polynomial $\frac{\sigma(\lambda;\{h_i\},m)}{\sigma_{\sup}}$ would lead to a better convergence rate. Hence, we should look for polynomials $\sigma(\lambda;\{h_i\},m)$ verifying $\sigma_{\sup} \leqslant 1$. And then,

$$1-\tau = \sqrt{m} = \left(\sigma_0 - \sqrt{\sigma_0^2-1}\right)^{\frac{1}{K}}. \tag{3.149}$$

which explain we aim at maximizing σ_0 subject to $\sigma_{\sup} \leqslant 1$ ((3.16)).

Finally, we proved **1)**: if $\sigma(\lambda;\{h_i\},m) = \sigma_K^\Lambda(\lambda)$, then the tuning is optimal in the sense that this is the one that minimizes the asymptotic rate factor among all K steps-sizes based tuning.

From now, we assume

$$\sigma(\lambda;\{h_i\},m) = \sigma_K^\Lambda(\lambda). \tag{3.150}$$

Therefore,

$$\sigma_0 = \sigma_K^\Lambda(0) \tag{3.151}$$

and **2)** is already proven above.

3) follows directly from the definition of $\sigma_K^\Lambda(\lambda)$.

Finally, since $\sigma_{\sup} \leqslant 1$, we know

$$1-\tau = \sqrt{m} = \left(\sigma_0 - \sqrt{\sigma_0^2-1}\right)^{1/K} \tag{3.152}$$

which proves part of **4)**.

To prove the expression of the worst-case rate r_t, we need to apply the intermediate result (3.140) instead of Theorem 3.4.8.

■

3.D.3 Example: alternating step-sizes ($K = 2$)

Proposition 3.D.2. *The strategy with 2 step-sizes is optimal on the union of two intervals if and only if they have the same length.*

Proof. This is a direct consequence of Theorem 3.C.2, which implies $\sigma_2^\Lambda(\boldsymbol{\mu}_1) = \sigma_2^\Lambda(\boldsymbol{L}_2) = 1$ and $\sigma_2^\Lambda(\boldsymbol{\mu}_2) = \sigma_2^\Lambda(\boldsymbol{L}_1) = -1$.

This is feasible if and only if $\boldsymbol{L}_2 - \boldsymbol{\mu}_2 = \boldsymbol{L}_1 - \boldsymbol{\mu}_1$ since σ_2^Λ is a degree 2 polynomial.

Indeed, set $\sigma_2^\Lambda(x) = a(x-b)^2 + c$. Then, $\sigma_2^\Lambda(\boldsymbol{\mu}_1) = \sigma_2^\Lambda(\boldsymbol{L}_2)$ implies $a(\boldsymbol{\mu}_1 - b)^2 + c = a(\boldsymbol{L}_2 - b)^2 + c$, then $|\boldsymbol{\mu}_1 - b| = |\boldsymbol{L}_2 - b|$ and finally $b = \frac{\boldsymbol{\mu}_1 + \boldsymbol{L}_2}{2}$. Similarly, $\sigma_2^\Lambda(\boldsymbol{\mu}_2) = \sigma_2^\Lambda(\boldsymbol{L}_1)$ implies $b = \frac{\boldsymbol{\mu}_2 + \boldsymbol{L}_1}{2}$.

We conclude $\frac{\boldsymbol{\mu}_1 + \boldsymbol{L}_2}{2} = \frac{\boldsymbol{\mu}_2 + \boldsymbol{L}_1}{2}$, and $\boldsymbol{L}_2 - \boldsymbol{\mu}_2 = \boldsymbol{L}_1 - \boldsymbol{\mu}_1$. ■

Proposition 3.4.5. *Let $\Lambda = [\boldsymbol{\mu}_1, \boldsymbol{L}_1] \cup [\boldsymbol{\mu}_2, \boldsymbol{L}_2]$ be an union of two intervals of the same size ($\boldsymbol{L}_1 - \boldsymbol{\mu}_1 = \boldsymbol{L}_2 - \boldsymbol{\mu}_2$) and let m, h_0, h_1 be as defined in Algorithm 3. Then the minimax polynomial (solution to (3.12)) is, for all $t = 2n$, $n \in \mathbb{N}_0^+$,*

$$\frac{T_n\left(\sigma_2^\Lambda(\lambda)\right)}{T_n\left(\sigma_2^\Lambda(0)\right)} = \underset{\substack{P \in \mathbb{R}_t[X], \\ P(0)=1}}{\arg\min} \sup_{\lambda \in \Lambda} |P(\lambda)|\,,$$

$$\textit{with}\quad \sigma_2^\Lambda(\lambda) = \frac{1}{2m}\left(1 + m - \lambda h_0\right)\left(1 + m - \lambda h_1\right) - 1\,.$$

Proof. From Theorem 3.C.2,

$$\sigma_2^\Lambda(\boldsymbol{\mu}_1) = 1, \tag{3.153}$$
$$\sigma_2^\Lambda(\boldsymbol{L}_1) = -1, \tag{3.154}$$
$$\sigma_2^\Lambda(\boldsymbol{\mu}_2) = -1, \tag{3.155}$$
$$\sigma_2^\Lambda(\boldsymbol{L}_2) = 1, \tag{3.156}$$

and this implies that $\frac{T_n\left(\sigma_2^\Lambda(\lambda)\right)}{T_n\left(\sigma_2^\Lambda(0)\right)}$ is optimal.

In particular, $\boldsymbol{L}_1$ and $\boldsymbol{\mu}_2$ are roots of $\sigma_2^\Lambda + 1$. Therefore, we know there exists a constant c such that $\sigma_2^\Lambda(\lambda) = c(1 - \frac{\lambda}{\boldsymbol{L}_1})(1 - \frac{\lambda}{\boldsymbol{\mu}_2}) - 1$. Moreover, evaluating this in $\boldsymbol{\mu}_1$ gives $\sigma_2^\Lambda(\boldsymbol{\mu}_1) = c(1 - \frac{\boldsymbol{\mu}_1}{\boldsymbol{L}_1})(1 - \frac{\boldsymbol{\mu}_1}{\boldsymbol{\mu}_2}) - 1 = 1$, so

$$c = \frac{2}{(1 - \frac{\boldsymbol{\mu}_1}{\boldsymbol{L}_1})(1 - \frac{\boldsymbol{\mu}_1}{\boldsymbol{\mu}_2})} \tag{3.157}$$
$$= \frac{2\boldsymbol{L}_1\boldsymbol{\mu}_2}{(\boldsymbol{L}_1 - \boldsymbol{\mu}_1)(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)} \tag{3.158}$$
$$= 2\frac{\left(\frac{\boldsymbol{\mu}_1 + \boldsymbol{L}_2}{2}\right)^2 - R^2\left(\frac{\boldsymbol{L}_2 - \boldsymbol{\mu}_1}{2}\right)^2}{\frac{1-R^2}{4}(\boldsymbol{L}_2 - \boldsymbol{\mu}_1)^2} \tag{3.159}$$
$$= 2\frac{\rho^2 - R^2}{1 - R^2}. \tag{3.160}$$

Then,

$$\sigma_2^\Lambda(\lambda) = 2\frac{\rho^2 - R^2}{1 - R^2}(1 - \frac{\lambda}{\boldsymbol{L}_1})(1 - \frac{\lambda}{\boldsymbol{\mu}_2}) - 1 \tag{3.161}$$

which can be written

$$\sigma_2^\Lambda(\lambda) = 2\left(\frac{1+m}{2\sqrt{m}}\right)^2\left(1-\frac{\lambda}{\boldsymbol{L}_1}\right)\left(1-\frac{\lambda}{\boldsymbol{\mu}_2}\right)-1 \tag{3.162}$$

with $\left(\frac{1+m}{2\sqrt{m}}\right)^2 = \frac{\rho^2-R^2}{1-R^2}$. Finally, $m = \left(\frac{\sqrt{\rho^2-R^2}-\sqrt{\rho^2-1}}{\sqrt{1-R^2}}\right)^2$. ■

Theorem 3.3.1 (Rate factor of $\mathrm{HB}_2(h_0,h_1;m)$)**.** *Let $f\in\mathcal{C}_\Lambda$ and consider the cyclical Heavy-ball method with step-sizes h_0, h_1 and momentum parameter m. The asymptotic rate factor of Algorithm 2 with cycles of length two is*

$$1-\tau = \begin{cases} \sqrt{m} & \text{if } \sigma_* \leqslant 1, \\ \sqrt{m}\left(\sigma_* + \sqrt{\sigma_*^2-1}\right)^{\frac{1}{2}} & \text{if } \sigma_* \in \left(1,\ \frac{1+m^2}{2m}\right), \\ \geqslant 1 \text{ (no convergence)} & \text{if } \sigma_* \geqslant \frac{1+m^2}{2m}, \end{cases}$$

$$\text{with} \quad \sigma_* = \max_{\lambda\in\left\{\mu_1,L_1,\mu_2,L_2,(1+m)\frac{h_0+h_1}{2h_0h_1}\right\}\cap\Lambda} |\sigma_2(\lambda)|$$

$$\text{and} \quad \sigma_2(\lambda) = 2\left(\frac{1+m-\lambda h_0}{2\sqrt{m}}\right)\left(\frac{1+m-\lambda h_1}{2\sqrt{m}}\right)-1\,.$$

Proof. From Theorem 3.4.8 applied to $K=2$, we immediately have the above result with

$$\sigma_{\sup} = \sup_{\lambda\in\Lambda}\left|2\left(\frac{1+m-\lambda h_0}{2\sqrt{m}}\right)\left(\frac{1+m-\lambda h_1}{2\sqrt{m}}\right)-1\right|.$$

To conclude the proof, we need to prove that the optimal value of $|\sigma_2^\Lambda|$ can only be reached on $\left\{\boldsymbol{\mu}_1,\boldsymbol{L}_1,\boldsymbol{\mu}_2,\boldsymbol{L}_2,(1+m)\frac{h_0+h_1}{2h_0h_1}\right\}$. Indeed, σ_2^Λ being convex, its maximal value can only be reached on $\{\boldsymbol{\mu}_1,\boldsymbol{L}_2\}$. Its minimal value is reached on $(1+m)\frac{h_0+h_1}{2h_0h_1}$. Therefore, over Λ, the minimal value of σ_2^Λ is reached on $(1+m)\frac{h_0+h_1}{2h_0h_1}$ if the latest belongs to Λ. Otherwise, its minimal value is reached to the closest point in Λ to $(1+m)\frac{h_0+h_1}{2h_0h_1}$, namely, it can be any point of $\{\boldsymbol{\mu}_1,\boldsymbol{L}_1,\boldsymbol{\mu}_2,\boldsymbol{L}_2\}$. ■

Proposition 3.D.3 (Residual polynomial in the robust region)**.** *Assuming $\sigma_2^\Lambda(\lambda)\geqslant -1, \quad \forall\lambda\in\Lambda$, the residual polynomial associated with the cyclical heavy-ball algorithm is*

$$P_{2n}(\lambda) = m^n\left[\frac{2m}{1+m}\,T_{2n}\left(\sqrt{\left(\frac{1+m-\lambda h_0}{2\sqrt{m}}\right)\left(\frac{1+m-\lambda h_1}{2\sqrt{m}}\right)}\right)\right.$$
$$\left. + \frac{1-m}{1+m}\,U_{2n}\left(\sqrt{\left(\frac{1+m-\lambda h_0}{2\sqrt{m}}\right)\left(\frac{1+m-\lambda h_1}{2\sqrt{m}}\right)}\right)\right]. \tag{3.163}$$

Remark 3.D.4. *The assumption $\sigma_2(\lambda)\geqslant -1, \quad \forall\lambda\in\Lambda$ is verified in the robust region, and is useful here because the term $\left(\frac{1+m-\lambda h_0}{2\sqrt{m}}\right)\left(\frac{1+m-\lambda h_1}{2\sqrt{m}}\right)$ is equal to $\frac{1+\sigma_2(\lambda)}{2}$ and must be positive to make the above expression well-defined. Otherwise the result can hold replacing the square root with some complex number, but it brings no value when we derive the convergence rate from it.*

Proof. This proof reuses elements of the proof of Theorem (3.4.8), especially Equation (3.127). For sake of completeness and simplicity, we prove this result again directly in the special case $K=2$.

We first recall the recursion of Algorithm 2 for $K = 2$. For sake of simplicity, we directly projet it onto the eigenspace associated to the eigenvalue λ of the Hessian of the objective function.

$$\begin{aligned} x_{2n+1} - x_* &= (1 + m - h_0\lambda)(x_{2n} - x_*) - m(x_{2n-1} - x_*). \\ x_{2n+2} - x_* &= (1 + m - h_1\lambda)(x_{2n+1} - x_*) - m(x_{2n} - x_*). \end{aligned} \tag{3.164}$$

Identifying $x_t - x_* = P_t(\lambda)(x_0 - x_*)$ and $P_t(\lambda) = (\sqrt{m})^t \, \tilde{P}_t(\lambda)$,

$$\begin{aligned} \tilde{P}_{2n+1}(\lambda) &= \tfrac{1+m-h_0\lambda}{\sqrt{m}} \tilde{P}_{2n}(\lambda) - \tilde{P}_{2n-1}(\lambda), \\ \tilde{P}_{2n+2}(\lambda) &= \tfrac{1+m-h_1\lambda}{\sqrt{m}} \tilde{P}_{2n+1}(\lambda) - \tilde{P}_{2n}(\lambda). \end{aligned} \tag{3.165}$$

Multiplying the first equation by $\frac{1+m-h_1\lambda}{\sqrt{m}}$ and replacing $\frac{1+m-h_1\lambda}{\sqrt{m}} \tilde{P}_{2n+1}(\lambda)$ and $\frac{1+m-h_1\lambda}{\sqrt{m}} \tilde{P}_{2n-1}(\lambda)$ accordingly to the second equation leads to

$$\tilde{P}_{2n+2}(\lambda) + \tilde{P}_{2n}(\lambda) = \frac{1 + m - h_0\lambda}{\sqrt{m}} \frac{1 + m - h_1\lambda}{\sqrt{m}} \tilde{P}_{2n}(\lambda) - \left(\tilde{P}_{2n}(\lambda) + \tilde{P}_{2n-2}(\lambda) \right) \tag{3.166}$$

which can be written as in equation (3.127)

$$\tilde{P}_{2n+2}(\lambda) = \left(\frac{1 + m - h_0\lambda}{\sqrt{m}} \frac{1 + m - h_1\lambda}{\sqrt{m}} - 2 \right) \tilde{P}_{2n}(\lambda) - \tilde{P}_{2n-2}(\lambda). \tag{3.167}$$

Moreover,

$$\begin{aligned} x_1 - x_* &= (1 - \tfrac{h_0}{1+m}\lambda)(x_0 - x_*), \\ x_2 - x_* &= (1 + m - h_1\lambda)(x_1 - x_*) - m(x_0 - x_*), \end{aligned} \tag{3.168}$$

leading to the initialization

$$\begin{aligned} \tilde{P}_1(\lambda) &= \tfrac{1}{\sqrt{m}}(1 - \tfrac{h_0}{1+m}\lambda)\tilde{P}_0(\lambda), \\ \tilde{P}_2(\lambda) &= \tfrac{1+m-h_1\lambda}{\sqrt{m}} \tilde{P}_1(\lambda) - \tilde{P}_0(\lambda). \end{aligned} \tag{3.169}$$

hence,

$$\tilde{P}_2(\lambda) = \left(\frac{1}{1 + m} \frac{1 + m - h_0\lambda}{\sqrt{m}} \frac{1 + m - h_1\lambda}{\sqrt{m}} - 1 \right) \tag{3.170}$$

and recall

$$\tilde{P}_0(\lambda) = 1. \tag{3.171}$$

It remains to notice that

$$\begin{aligned} &\frac{2m}{1+m} T_{2n}\left(\sqrt{\left(\frac{1 + m - \lambda h_0}{2\sqrt{m}} \right) \left(\frac{1 + m - \lambda h_0}{2\sqrt{m}} \right)} \right) \\ &+ \frac{1-m}{1+m} U_{2n}\left(\sqrt{\left(\frac{1 + m - \lambda h_0}{2\sqrt{m}} \right) \left(\frac{1 + m - \lambda h_0}{2\sqrt{m}} \right)} \right) \end{aligned} \tag{3.172}$$

verifies the same recursion as well as the same initialization for $n = 0$ and $n = 1$. This allows us to identify the 2 sequences of polynomials

$$\begin{aligned} \tilde{P}_{2n}(\lambda) = &\frac{2m}{1+m} T_{2n}\left(\sqrt{\left(\frac{1 + m - \lambda h_0}{2\sqrt{m}} \right) \left(\frac{1 + m - \lambda h_0}{2\sqrt{m}} \right)} \right) \\ &+ \frac{1-m}{1+m} U_{2n}\left(\sqrt{\left(\frac{1 + m - \lambda h_0}{2\sqrt{m}} \right) \left(\frac{1 + m - \lambda h_0}{2\sqrt{m}} \right)} \right) \end{aligned} \tag{3.173}$$

which concludes the proof.

■

Corollary 3.3.2. *The non-asymptotic and asymptotic worst-case rates $r_t^{\text{Alg. 2}}$ and $1-\tau^{\text{Alg. 2}}$ of Algorithm 3 over $\mathcal{C}_\Lambda$ for even iteration number t are*

$$r_t^{\text{Alg. 2}} = \left(\frac{\sqrt{\rho^2-R^2}-\sqrt{\rho^2-1}}{\sqrt{1-R^2}}\right)^t \left(1+t\sqrt{\frac{\rho^2-1}{\rho^2-R^2}}\right),$$
$$1-\tau^{\text{Alg. 2}} = \frac{\sqrt{\rho^2-R^2}-\sqrt{\rho^2-1}}{\sqrt{1-R^2}}.$$

Proof. From Proposition 3.4.5, Algorithm 3's parameter make $\sigma(\lambda;\{h_i\},m) = \sigma_2^\Lambda$. In particular, by definition,

$$-1 \leqslant 2\left(\frac{1+m-\lambda h_0}{2\sqrt{m}}\right)\left(\frac{1+m-\lambda h_1}{2\sqrt{m}}\right) - 1 \leqslant 1. \tag{3.174}$$

and then

$$0 \leqslant \sqrt{\left(\frac{1+m-\lambda h_0}{2\sqrt{m}}\right)\left(\frac{1+m-\lambda h_0}{2\sqrt{m}}\right)} \leqslant 1. \tag{3.175}$$

And we know that $\forall x \leqslant 1$, $T_n(x) \leqslant 1$ and $U_n(x) \leqslant n+1$.

Therefore, using optimal parameters, and from Proposition 3.D.3

$$\tilde{P}_{2n}(\lambda) \leqslant \frac{2m}{1+m} + (2n+1)\frac{1-m}{1+m} = 1+2n\frac{1-m}{1+m}. \tag{3.176}$$

And the worst-case rate is then upper bounded

$$r_t = \left(1+t\frac{1-m}{1+m}\right)(\sqrt{m})^t \tag{3.177}$$

for all t even.

It remains to plug m expression into the above to conclude. ∎

Note that in the proof above, all the expressions are symmetric in (h_0,h_1), which implies that swapping those 2 step-sizes doesn't impact this statement.

Remark 3.D.5. *The previous statement provides the convergence rate of Algorithm 3. It does not state that this is the optimal way to tune Algorithm 2, but comparing the obtained rate to the one of Algorithm 7 does. Another way to derive the optimal parameters, is to start from the result of Theorem 3.3.1 applied on a 2 step-sizes strategy, or from the result of Proposition 3.D.3. This leads to minimizing m under the constraints that $\zeta(\lambda) \triangleq \left(\frac{1+m-\lambda h_0}{2\sqrt{m}}\right)\left(\frac{1+m-\lambda h_1}{2\sqrt{m}}\right)$ has values between 0 and 1 on $\Lambda = [\boldsymbol{\mu}_1, \boldsymbol{L}_1] \cup [\boldsymbol{\mu}_2, \boldsymbol{L}_2]$. By symmetry of Λ and the convex parabola ζ, we know that optimal parameters verify $\zeta(\boldsymbol{L}_1) = \zeta(\boldsymbol{\mu}_2) = 0$. And therefore, $\zeta(\boldsymbol{\mu}_1) = \zeta(\boldsymbol{L}_2) = 1$ maximizes the range of allowed m. This way we recover the tuning of Algorithm 3. Note that ζ is related to $\sigma_2^{(\Lambda)}$ through the relation $\sigma_2^{(\Lambda)} = 2\zeta - 1$, and therefore the 4 mentioned equalities are equivalent to the equioscillation property.*

The next theorem sums up the results of Proposition 3.3.3 and Table 3.1.

Theorem 3.D.6 (Asymptotic speedup of HB with alternating step-sizes)**.**

1. *Let $R \in [0,1)$ be a fixed number, then $\sqrt{m} \underset{\kappa\to 0}{=} 1 - \frac{2\sqrt{\kappa}}{\sqrt{1-R^2}} + o(\sqrt{\kappa})$.*
2. *Let*

$$R(\kappa) \underset{\kappa\to 0}{=} 1 - \frac{\sqrt{\kappa}}{2} + o(\sqrt{\kappa}), \qquad \textit{i.e.,} \quad \Lambda \approx [\mu, \mu + \frac{\sqrt{\mu L}}{4}] \cup [L - \frac{\sqrt{\mu L}}{4}, L],$$

then $\sqrt{m} \underset{\kappa\to 0}{=} 1 - 2\sqrt[4]{\kappa} + o(\sqrt[4]{\kappa})$, therefore leasing to a new square root acceleration.

3. *Let*

$$R(\kappa) \underset{\kappa\to 0}{=} 1 - 2\gamma\kappa + o(\kappa), \qquad \textit{i.e.,} \quad \Lambda \approx [\mu, (1+\gamma)\mu] \cup [L - \gamma\mu, L],$$

then $\sqrt{m} \underset{\kappa\to 0}{=} \sqrt{1 + \frac{1}{\gamma}} - \sqrt{\frac{1}{\gamma}} + o(\kappa)$*, therefore leading to a constant complexity.*

This is summed up in the Table 3.2.

Relative gap R	Set Λ	Rate factor τ	Speedup τ/τ^{PHB}
$R \in [0,1)$	$[\mu, \mu + \frac{1-R}{2}(L-\mu)] \cup [L - \frac{1-R}{2}(L-\mu), L]$	$\frac{2\sqrt{\kappa}}{\sqrt{1-R^2}}$	$(1-R^2)^{-\frac{1}{2}}$
$R = 1 - \sqrt{\kappa}/2$	$[\mu, \mu + \frac{\sqrt{\mu L}}{4}] \cup [L - \frac{\sqrt{\mu L}}{4}, L]$	$2\sqrt[4]{\kappa}$	$\kappa^{-\frac{1}{4}}$
$R = 1 - 2\gamma\kappa$	$[\mu, (1+\gamma)\mu] \cup [L - \gamma\mu, L]$	indep. of κ	$O(\kappa^{-\frac{1}{2}})$

Table 3.2: Case study of the convergence of Algorithm 3 as a function of R, in the regime where $\kappa \to 0$. The **first line** corresponds to a situation where R is independent of κ, and we observe a constant gain w.r.t. heavy-ball. The **second line** study a setting in which R depends on $\sqrt{\kappa}$, meaning the two intervals in Λ are relatively small. The asymptotic rate reads $(1 - 2\sqrt[4]{\kappa})^t$, beating the $(1 - 2\sqrt{\kappa})^t$ lower bound. Finally, in the **third line**, R depends on κ, the two intervals in Λ are so small that the convergence becomes $O(1)$, i.e., is independent of κ.

Proof.

1. Let $R \in [0,1)$. The momentum m satisfies

$$\begin{aligned}\sqrt{m} &\underset{\kappa\to 0}{=} \frac{\sqrt{1 + O(\kappa) - R^2} - \sqrt{4\kappa + O(\kappa^2)}}{\sqrt{1-R^2}} \\ &\underset{\kappa\to 0}{=} \frac{\sqrt{1-R^2} + O(\kappa) - 2\sqrt{\kappa} + O(\kappa)}{\sqrt{1-R^2}} \\ &\underset{\kappa\to 0}{=} 1 - \frac{2\sqrt{\kappa}}{\sqrt{1-R^2}} + O(\kappa).\end{aligned}$$

2. Let $R(\kappa) \underset{\kappa\to 0}{=} 1 - \frac{\sqrt{\kappa}}{2} + o(\sqrt{\kappa})$. The momentum m verifies

$$\begin{aligned}\sqrt{m} &= \sqrt{\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - R^2}{1-R^2}} - \sqrt{\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - 1}{1-R^2}} \\ &= \sqrt{\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - 1}{1-R^2} + 1} - \sqrt{\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - 1}{1-R^2}}.\end{aligned}$$

We first focus on

$$\begin{aligned}\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - 1}{1-R^2} &\underset{\kappa\to 0}{=} \frac{4\kappa + O(\kappa^2)}{\sqrt{\kappa} + o(\sqrt{\kappa})} \\ &\underset{\kappa\to 0}{=} 4\sqrt{\kappa} + o(\sqrt{\kappa}).\end{aligned}$$

Then,

$$\begin{aligned}\sqrt{m} &= \sqrt{\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - 1}{1-R^2} + 1} - \sqrt{\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - 1}{1-R^2}} \\ &\underset{\kappa\to 0}{=} \sqrt{1 + 4\sqrt{\kappa} + o(\sqrt{\kappa})} - \sqrt{4\sqrt{\kappa} + o(\sqrt{\kappa})} \\ &\underset{\kappa\to 0}{=} 1 + 2\sqrt{\kappa} + o(\sqrt{\kappa}) - 2\sqrt[4]{\kappa} + o(\sqrt[4]{\kappa}) \\ &\underset{\kappa\to 0}{=} 1 - 2\sqrt[4]{\kappa} + o(\sqrt[4]{\kappa}).\end{aligned}$$

3. Let $R(\kappa) \underset{\kappa\to 0}{=} 1 - 2\gamma\kappa + o(\kappa)$. The momentum m verifies

$$\begin{aligned}\sqrt{m} &= \sqrt{\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - R^2}{1-R^2}} - \sqrt{\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - 1}{1-R^2}} \\ &= \sqrt{\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - 1}{1-R^2} + 1} - \sqrt{\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - 1}{1-R^2}}.\end{aligned}$$

We first focus on

$$\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - 1}{1-R^2} \underset{\kappa\to 0}{=} \frac{4\kappa + O(\kappa^2)}{4\gamma\kappa + o(\kappa)} \underset{\kappa\to 0}{=} \frac{1}{\gamma} + o(\kappa).$$

Then,

$$\begin{aligned}\sqrt{m} &= \sqrt{\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - 1}{1-R^2} + 1} - \sqrt{\frac{\left(\frac{1+\kappa}{1-\kappa}\right)^2 - 1}{1-R^2}} \\ &\underset{\kappa\to 0}{=} \sqrt{1 + \frac{1}{\gamma} + o(\kappa)} - \sqrt{\frac{1}{\gamma} + o(\kappa)} \\ &\underset{\kappa\to 0}{=} \sqrt{1 + \frac{1}{\gamma}} - \sqrt{\frac{1}{\gamma}} + o(\kappa).\end{aligned}$$

■

3.D.4 Example: 3 cycling step-sizes

Proposition 3.D.7. *The strategy with 3 step-sizes is optimal on the union of two intervals if and only if they are of the form*

$$\left[\mu, \mu + (L-\mu)\left(\frac{1}{2} - \frac{R}{2} + \frac{1-R^2}{4}\right)\right] \cup \left[L - (L-\mu)\left(\frac{1}{2} - \frac{R}{2} - \frac{1-R^2}{4}\right), L\right],$$

for some $R \in [0, 1]$.

Proof. From Theorem 3.C.2, we know that $T_n(\sigma_3)$ is optimal for all n if and only if, Λ is the union of 3 different intervals that are mapped on $[-1, 1]$. Since, we are looking for Λ being the union of 2 intervals, we know 2 of the 3 intervals Λ is composed of share an extremity. Recall $\Lambda = [\mu_1, L_1] \cup [\mu_2, L_2]$. By symmetry, we can assume without loss of generality that

$[\mu_1, L_1]$ is mapped to $[-1, 1]$ twice, and $[\mu_2, L_2]$ once. Let's then introduce $x \in (\mu_1, L_1)$ and say:

$$\begin{aligned} \sigma_3(\mu_1) &= 1, && (3.178)\\ \sigma_3(x) &= -1, && (3.179)\\ \sigma_3(L_1) &= 1, && (3.180)\\ \sigma_3(\mu_2) &= 1, && (3.181)\\ \sigma_3(L_2) &= -1. && (3.182) \end{aligned}$$

Note we also know that x is a local minima of σ_3, leading to $\sigma_3'(x) = 0$. We now know 3 roots of $\sigma_3 + 1$ and 3 roots of $\sigma_3 - 1$, leading to:

$$\begin{aligned} \sigma_3(\lambda) - 1 &= c(\lambda - \mu_1)(\lambda - L_1)(\lambda - \mu_2), && (3.183)\\ \sigma_3(\lambda) + 1 &= c(\lambda - x)^2(\lambda - L_2), && (3.184) \end{aligned}$$

for some non-zero constant c. Here, we want to remove the dependency in x or c. Using the two equalities above,

$$(\lambda - x)^2(\lambda - L_2) - (\lambda - \mu_1)(\lambda - L_1)(\lambda - \mu_2) = \frac{2}{c}. \tag{3.185}$$

Matching the coefficients of the above polynomial leads to

$$\begin{aligned} 2x + L_2 &= \mu_1 + L_1 + \mu_2 && (3.186)\\ &\text{and} && (3.187)\\ 2xL_2 + x^2 &= \mu_1 L_1 + \mu_1\mu_2 + L_1\mu_2. && (3.188) \end{aligned}$$

We plug the expression of x we get from the first equality into the second one,

$$L_2(\mu_1 + L_1 + \mu_2 - L_2) + \left(\frac{\mu_1 + L_1 + \mu_2 - L_2}{2}\right)^2 = \mu_1 L_1 + \mu_1\mu_2 + L_1\mu_2. \tag{3.189}$$

From here, for simplicity, we define

$$r_i \triangleq \frac{L_i - \mu_i}{L_2 - \mu_1}, \qquad \text{for } i \in \{1, 2\}. \tag{3.190}$$

Replacing L_1 and μ_2 by their expression using μ_1, L_2, r_1 and r_2 leads to

$$r_1 = 2\sqrt{r_2} - r_2. \tag{3.191}$$

The reciprocal holds and we can find x using Equation (3.186) or (3.188). Note if Equation (3.191) holds, we can directly express σ_3 as the unique polynomial verifying

$$\begin{aligned} \sigma_3(\mu_1) &= 1, && (3.192)\\ \sigma_3(L_1) &= 1, && (3.193)\\ \sigma_3(\mu_2) &= 1, && (3.194)\\ \sigma_3(L_2) &= -1. && (3.195) \end{aligned}$$

We can therefore conclude

$$\sigma_3(\lambda) = 1 - 2\frac{(\lambda - \mu_1)(\lambda - L_1)(\lambda - \mu_2)}{(L_2 - \mu_1)(L_2 - L_1)(L_2 - \mu_2)}. \tag{3.196}$$

From the new notations $r_1, r_2, \mu = \mu_1, L = L_2$, we know $T_n(\sigma_3^\Lambda)$ is optimal for all n if and only if

$$\Lambda = [\mu, \mu + r_1(L-\mu)] \cup [L - r_2(L-\mu), L]. \tag{3.197}$$

Let R be

$$R = \frac{\mu_2 - L_1}{L_2 - \mu_1} \tag{3.198}$$

as in the 2 step-sizes setting. Here, we have $R = 1 - r_1 - r_2$ and we assume $r_1 = 2\sqrt{r_2} - r_2$. Combining those 2 equalities gives:

$$r_1 = \frac{1}{2} - \frac{R}{2} + \frac{1-R^2}{4}, \tag{3.199}$$

$$r_2 = \frac{1}{2} - \frac{R}{2} - \frac{1-R^2}{4}, \tag{3.200}$$

leading to the desired result, i.e.,

$$\Lambda = [\mu, \mu + (L-\mu)(\frac{1}{2} - \frac{R}{2} + \frac{1-R^2}{4})] \cup [L - (L-\mu)(\frac{1}{2} - \frac{R}{2} - \frac{1-R^2}{4}), L].$$

■

Theorem 3.D.8 (Asymptotic speedup of heavy-ball when cycling over 3 step-sizes)**.** *Let $R \in [0, 1)$ be a fixed number, then*

$$\sqrt{m} \underset{\kappa \to 0}{=} 1 - 2\sqrt{\kappa}\sqrt{\frac{1 - R^2/9}{1 - R^2}} + o(\sqrt{\kappa}). \tag{3.201}$$

Proof. From Equation (3.145),

$$\sqrt{m} = \left(\sigma_3^{(\Lambda)}(0) - \sqrt{\sigma_3^{(\Lambda)}(0)^2 - 1}\right)^{\frac{1}{3}} \quad \text{with} \quad \sigma_3^{(\Lambda)}(0) = 1 + 2\frac{\mu_1 L_1 \mu_2}{(L_2 - \mu_1)(L_2 - L_1)(L_2 - \mu_2)}.$$

Using the previous notations,

$$\mu = \mu_1, \tag{3.202}$$

$$L = L_2, \tag{3.203}$$

$$\kappa = \frac{\mu}{L}, \tag{3.204}$$

$$r_i \triangleq \frac{L_i - \mu_i}{L_2 - \mu_1}, \qquad \text{for } i \in \{1, 2\}, \tag{3.205}$$

we can write $\sigma_3^{(\Lambda)}$ as

$$\begin{aligned}
\sigma_3^{(\Lambda)}(0) &= 1 + 2\frac{\mu_1 L_1 \mu_2}{(L_2-\mu_1)(L_2-L_1)(L_2-\mu_2)}, && (3.206)\\
&= 1 + 2\frac{\kappa(\kappa + r_1(1-\kappa))(1-r_2(1-\kappa))}{(1-\kappa)^3(1-r_1)r_2}, && (3.207)\\
&\underset{\kappa\to 0}{=} 1 + 2\kappa\frac{r_1(1-r_2)}{(1-r_1)r_2,} && (3.208)\\
&= 1 + 2\kappa\frac{\left(\frac{1}{2} - \frac{R}{2} + \frac{1-R^2}{4}\right)\left(\frac{1}{2} + \frac{R}{2} - \frac{1-R^2}{4}\right)}{\left(\frac{1}{2} + \frac{R}{2} + \frac{1-R^2}{4}\right)\left(\frac{1}{2} - \frac{R}{2} - \frac{1-R^2}{4}\right)}, && (3.209)\\
&= 1 + 2\kappa\frac{9 - 10R^2 + R^4}{1 - 2R^2 + R^4}, && (3.210)\\
&= 1 + 2\kappa\frac{(1-R^2)(9-R^2)}{(1-R^2)^2}, && (3.211)\\
&= 1 + 2\kappa\frac{9-R^2}{1-R^2}. && (3.212)
\end{aligned}$$

Then introducing briefly $\varepsilon \triangleq \kappa\frac{9-R^2}{1-R^2} \underset{\kappa\to 0}{\to} 0$,

$$\begin{aligned}
\sqrt{m} &= \left(\sigma_3^{(\Lambda)}(0) - \sqrt{\sigma_3^{(\Lambda)}(0)^2 - 1}\right)^{\frac{1}{3}}, && (3.213)\\
&= \left(1 + 2\varepsilon - \sqrt{1 + 4\varepsilon + 4\varepsilon^2 - 1}\right)^{\frac{1}{3}}, && (3.214)\\
&\underset{\kappa\to 0}{=} 1 - \frac{2}{3}\sqrt{\varepsilon} + o(\sqrt{\varepsilon}). && (3.215)
\end{aligned}$$

Plugging ε expression into the latest gives

$$\sqrt{m} \underset{\kappa\to 0}{=} 1 - 2\sqrt{\kappa}\sqrt{\frac{1 - R^2/9}{1-R^2}} + o(\sqrt{\kappa}). \tag{3.216}$$

■

3.E Beyond quadratic objective: local convergence of cycling methods

In this section, we prove a result of local convergence of the cyclical heavy-ball method out of quadratic setting. We first recall the Theorem 3.5.1 stated in Section 3.5:

Theorem 3.5.1 (Local convergence)**.** *Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ be a twice continuously differentiable function, x_* a local minimizer, and H be the Hessian of f at x_* with $Sp(H) \subseteq \Lambda$. Let x_t denote the result of running Algorithm 2 with parameters $h_1, h_2, \cdots, h_K, m$, and let $1-\tau$ be the linear convergence rate on the quadratic objective* (OPT)*. Then we have*

$$\begin{aligned}
&\forall \varepsilon > 0, \exists \text{ open set } V_\varepsilon : x_0,\, x_* \in V_\varepsilon\\
\implies &\|x_t - x_*\| = O((1 - \tau + \varepsilon)^t)\|x_0 - x_*\|. && (3.24)
\end{aligned}$$

Proof. For any k multiple of K, consider S_k the operator applying k steps of cycling

Heavy-ball on the iterates x_t and x_{t-1} (note since k is a multiple of K, Algorithm 2 consists in repeating the operator S_k). Namely S_k is an operator on $\mathbb{R}^{2d}$ verifying $S_k((x_t, x_{t-1})) = (x_{t+k}, x_{t+k-1})$. This operator is a composition of gradients of f and affine functions, and so it is continuously differentiable.

Applying the mean value theorem along each coordinate of S_k, we have that there exists a matrix-valued function $M(v_1, v_2)$ for all v_1, v_2 in the domain of S_k such that

$$S_k(v_1) - S_k(v_2) = M(v_1, v_2)(v_1 - v_2)\,, \tag{3.217}$$

where the i^{th} rows of $M(v_1, v_2)$ is the gradient of the i^{th} output of S_k evaluated at a vector on the segment between v_1 and v_2.

$$M(v_1, v_2) = \begin{pmatrix} \nabla(S_k)_1(w_1)^T \\ \vdots \\ \nabla(S_k)_i(w_i)^T \\ \vdots \\ \nabla(S_k)_{2d}(w_{2d})^T \end{pmatrix} \text{ where } \forall i \in [\![1, 2d]\!], \begin{cases} (S_k)_i \text{ denotes the } i\text{th coordinate of } S_k. \\ \\ w_i \text{ is a point on the segment } [v_1, v_2]. \end{cases} \tag{3.218}$$

By continuity of those gradients, taking v_1 and v_2 sufficiently close to (x_*, x_*), $M(v_1, v_2)$ can be chosen arbitrarily close to the Jacobian of S_k in (x_*, x_*) denoted by JS_k^*.

Since by assumption the algorithm converges on the quadratic form induced by H at the rate $1 - \tau$, we conclude that the spectral radius of JS_k^* is upper bounded by $1 - \tau$.

From the previous point, we can find a small enough neighborhood of (x_*, x_*) such that $M(v_1, v_2)$ has a spectral radius arbitrarily close to $1 - \tau$, in particular smaller than 1.

Furthermore, it's known for any $\varepsilon > 0$, there exists an operator norm $\|.\|$ such that $\|M(v_1, v_2)\| < 1 - \tau + \varepsilon$. (see e.g. (Bertsekas, 1997, Proposition A.15)).

Hence, for any $\varepsilon > 0$, there exists a neighborhood V of (x_*, x_*) and an operator norm $\|.\|$ as described above such that S_k is a $(1 - \tau + \varepsilon)$-contraction on V for the norm $\|.\|$.

This leads to convergence to the only fixed point (x_*, x_*) with a convergence rate smaller than any $1 - \tau + \varepsilon$.

Moreover, the first step of the Algorithm 2 is continuous with respect to x_0. Hence, for any $V \in \mathbb{R}^{2d}$ neighborhood of (x_*, x_*), there exists $W \in \mathbb{R}^d$ a neighborhood of x_*, such that

$$x_0 \in W \Longrightarrow (x_1, x_0) \in V. \tag{3.219}$$

Finally, for any $\varepsilon > 0$, there exists W a neighborhood of x_* such that the Algorithm 2 converges to x_* with a rate smaller than $1 - \tau + \varepsilon$.

■

3.F Experimental setup

Benchmarks we run using a Google colab public instance with a single CPU. Producing the results of Figure 3.4 took 50 minutes with this setup. The code to reproduce this figure is attached with the supplementary material in the jupyter notebook benchmarks.ipynb .

3.G Comparison with Oymak (2021)

The work of Oymak (2021) also exploits cyclical step-sizes for when the spectral structure of the Hessian contains a gap. This work appeared concurrently to the first version of this

manuscript and takes a somewhat different stand for exploiting this particular spectral structure. We summarize the main differences in the table below.

	Oymak (2021)	This work
Algorithm	Gradient descent	Heavy-ball-type (optimal algorithm)
Cycle length K	K is a function of the spectral assumptions	K is a choice
Structure of the cycle	$\underbrace{\eta_+, \ldots, \eta_+}_{K-1 \text{ times}}, \eta_-$ (Oymak, 2021, Definition 1)	$(h_0, \ldots, h_{K-1})$ (See Algorithm 2)
Optimal among chosen scheme	Not proven / discussed	Yes
Spectral assumption	Bimodal (2 intervals)	Any number of intervals
Spectral assumption in bimodal case	Rate depends on $\frac{L_2}{\mu_2}$ and $\frac{L_1}{\mu_1}$	Rate depends on $L_2 - \mu_2$ and $L_1 - \mu_1$
Typical application case in bimodal	Very strong assumption on $L_1 - \mu_1$ and very weak assumption on $L_2 - \mu_2$.	Weak assumption on both $L_2 - \mu_2$ and $L_1 - \mu_1$, more in line with empirical observations (Papyan, 2018; Ghorbani et al., 2019).
Spectrum is a single interval	Does not recover original rate	Recovers rate and optimal method
Convergence beyond quadratic objectives	Yes (with extra assumptions)	Local convergence

We emphasize the following points.

- While we provide Theorem 3.4.8 which described the convergence rate of any cycling heavy-ball (for any cycle), Oymak (2021) only studied Gradient descent method (without momentum) for a particular cycle (for which cycle length is not a parameter, but fixed by the eigenvalues).
- Moreover, our Theorem 3.4.9 provides the optimal cycle to use for any choice of a cycle length, while optimality is not discussed in Oymak (2021) and the cycle uses only two different step-sizes, which is somewhat arbitrary.

- Furthermore, our work highlights acceleration under assumptions that seem more aligned with empirical observation: Oymak (2021) shows that $L_1 - \mu_1$ needs to be very small for his strategy to be worthwhile, while this is not really the case in our experiments (see Figure 3.1)
- Finally, in the general case in which we cannot assume any gap in the spectrum, we naturally recover the classical optimal method and rate. This is not the case in Oymak (2021) which is suboptimal in this setup.

In this work, we focus on quadratic minimization and give some local convergence guarantee beyond quadratics. On the other hand, Oymak (2021) provides guarantee beyond this setup, at the cost of very restrictive assumptions.

Part II

Tools for optimization over non-parametric classes of functions

4

PEPIT: computer-assisted worst-case analyses of first-order optimization methods in Python

PEPIT is a PYTHON package aiming at simplifying the access to worst-case analyses of a large family of first-order optimization methods possibly involving gradient, projection, proximal, or linear optimization oracles, along with their approximate, or Bregman variants.

In short, PEPIT is a package enabling computer-assisted worst-case analyses of first-order optimization methods. The key underlying idea is to cast the problem of performing a worst-case analysis, often referred to as a performance estimation problem (PEP), as a semidefinite program (SDP) which can be solved numerically. To do that, the package users are only required to write first-order methods nearly as they would have implemented them. The package then takes care of the SDP modeling parts, and the worst-case analysis is performed numerically via a standard solver.

This chapter is based on our work "PEPIT: computer-assisted worst-case analyses of first-order optimization methods in Python" (co-authored with C. Moucer, F. Glineur, J. Hendrickx, A. Taylor, and A. Dieuleveut), currently under review.

Contents

4.1 Introduction

Due to their low cost per iteration, first-order optimization methods became a major tool in the modern numerical optimization toolkit. Those methods are particularly well suited when targeting only low to medium accuracy solutions, and play a central role in many fields of applications that include machine learning and signal processing. Their simplicity further allows both occasional and expert users to use them. On the contrary, when it comes to their analyses (usually based on worst-case scenarios), they are mostly reserved for expert users. The main goal of this work is to allow simpler and reproducible access to worst-case analyses for first-order methods.

PEPIT is a PYTHON package enabling computer-assisted worst-case analysis of a large family of first-order optimization methods. After being provided with a first-order method and a standard problem class, the package reformulates the problem of performing a worst-case analysis as a semidefinite program (SDP). This technique is commonly referred to as *performance estimation problems* (PEPs) and was introduced by Drori and Teboulle (2014); Drori (2014). The package uses PEPs as formalized by Taylor et al. (2017c,a).

In short, performing a worst-case analysis of a first-order algorithm usually relies on four main ingredients: a first-order algorithm (to be analyzed), a class of problems (containing the assumptions on the function to be minimized), a performance measure (measuring the quality of the output of the algorithm under consideration; for convenience here we assume that the algorithm aims at minimizing this performance measure and our analysis aims at finding a worst-case guarantee on it), and an initial condition (measuring the quality of the initial iterate). Performing the worst-case analysis (i.e., computing worst-case scenarios) corresponds to maximizing the performance measure of the algorithm on the class of problems, under a constraint induced by the initial condition. It turns out that such optimization problems can often be solved using SDPs in the context of first-order methods.

PEPs provide a principled approach to worst-case analyses but usually rely on potentially tedious semidefinite programming (SDP) modeling and coding steps. The PEPIT package eases access to the methodology by automatically handling the modeling part, thereby limiting the amount of time spent on this tedious task and the risk of introducing coding mistakes in the process. In short, this work allows users to (i) write their first-order algorithms nearly as they would have implemented them, (ii) let PEPIT (a) perform the modeling and coding steps, and (b) perform the worst-case analysis numerically using tools for semidefinite programming in PYTHON MOSEK (2019); Diamond and Boyd (2016); O'Donoghue et al. (2016).

As a result, the package enables users to easily obtain worst-case analyses for most of the standard first-order methods, classes of problems, performance measures, and initial conditions. This is useful to numerically verify existing convergence guarantees, as well as to ease the development of new analyses and methods. To this end, the toolbox contains tools for analyzing classical scenarios of the first-order literature: standard problem classes (such as convex functions, smooth convex functions, Lipschitz convex functions, etc.) and algorithmic operations (such as gradient, proximal, or linear optimization oracles, etc.). Finally, the package contains more than 75 examples and is designed in an open fashion, allowing users to easily add new ingredients (such as their own problem classes, oracles, or algorithms as examples).

Organization of the paper. This paper is organized as follows. First, Section 4.2 exemplifies the PEP approach on a very simple example, namely computing a worst-case contraction factor for Gradient descent, and shows how to code this example in PEPIT. Section 4.3 provides details on the semidefinite programs that can be formulated through the package along with the relationship between those formulations and the coding steps. Then, Section 4.4 provides a roadmap through the package. Finally, Section 4.5 provides three additional numerical examples (including a composite minimization problem and a stochastic one), and some concluding remarks and perspectives are drawn in Section 4.6.

Related works. The PEPIT package relies on performance estimation problems as formalized in Taylor et al. (2017a). It also contains some improvements and generalizations to other problem and algorithmic classes such as monotone and nonexpansive operators Ryu et al. (2020); Lieder (2021), quadratic optimization Bousselmi et al. (2023), stochastic methods and verification of potential (or Lyapunov/energy) functions Hu et al. (2018); Fazlyab et al. (2018); Taylor and Bach (2019) as inspired by the related control-theoretic IQC framework Lessard et al. (2016). The package also contains numerous examples; e.g., recent analyses and developments from Kim and Fessler (2016); Van Scoy et al. (2017); Gu and Yang (2020); Kim and Fessler (2021); Kim (2021); Lieder (2021); Gannot (2021); Taylor and Drori (2022); Abbaszadehpeivasti et al. (2021); Gorbunov et al. (2022). The package can be seen as an extended open source PYTHON version of the MATLAB package PESTO Taylor et al. (2017b) on various aspects such as its documentation, its coding style and its access through standard open interfaces (such as pip), PEPIT is more professional than PESTO.

Dependencies The package heavily builds on existing software for solving semidefinite programs, including CVXPY Diamond and Boyd (2016), SCS O'Donoghue et al. (2016), and MOSEK MOSEK (2019).

4.2 PEPIT on a simple example

In this section, we illustrate the use of the package for studying the worst-case properties of a standard scenario: Gradient descent for minimizing a smooth strongly convex function. The goal of this elementary example is twofold. First, we want to provide the base mathematical steps enabling the use of semidefinite programming for performing worst-case analyses, together with a corresponding PEPIT code. Second, we want to highlight the main ingredients that can be generalized to other problem setups (e.g., Theorem 4.2.1 below providing "interpolation conditions" for the class of smooth strongly convex functions), allowing us to analyze more algorithms under different assumptions (which are listed in Section 4.4).

For this example, we consider the convex optimization problem

$$\min_{x\in\mathbb{R}^d} f(x), \tag{4.1}$$

where f is L-smooth and μ-strongly convex (notation $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$, or $f \in \mathcal{F}_{\mu,L}$ when d is unspecified). So we assume f to satisfy

- (L-smoothness) $\forall x, y \in \mathbb{R}^d$ we have that

$$f(x) \leqslant f(y) + \langle \nabla f(y); x - y\rangle + \frac{L}{2}\|x - y\|_2^2,$$

- (μ-strong convexity) $\forall x, y \in \mathbb{R}^d$ we have that

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

Our goal for the rest of this section is to show how to compute the smallest possible $\tau(\mu, L, \gamma)$ (often referred to as the "contraction factor") such that

$$\|x_1 - y_1\|_2^2 \leqslant \tau(\mu, L, \gamma)\|x_0 - y_0\|_2^2, \tag{4.2}$$

is valid for all $f \in \mathcal{F}_{\mu,L}$ and all $x_0, y_0 \in \mathbb{R}^d$ when x_1 and y_1 are obtained from gradient steps from respectively x_0 and y_0. That is, $x_1 = x_0 - \gamma\nabla f(x_0)$ and $y_1 = y_0 - \gamma\nabla f(y_0)$. First, we show that the problem of computing $\tau(\mu, L, \gamma)$ can be framed as a semidefinite program (SDP), and then illustrate how to use PEPIT for computing it without going into the SDP modeling details.

4.2.1 A performance estimation problem for the gradient method

It is relatively straightforward to establish that the smallest possible $\tau(\mu, L, \gamma)$ for which (4.2) is valid can be computed as the worst-case value of $\|x_1 - y_1\|_2^2$ when $\|x_0 - y_0\|_2^2 \leqslant 1$. That is, we compute $\tau(\mu, L, \gamma)$ as the optimal value to the following optimization problem:

$$\begin{aligned} \tau(\mu, L, \gamma) = \max_{\substack{f,d \\ x_0,x_1 \in \mathbb{R}^d \\ y_0,y_1 \in \mathbb{R}^d}} \quad & \|x_1 - y_1\|_2^2 \\ \text{s.t. } & d \in \mathbb{N},\ f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d), \\ & \|x_0 - y_0\|_2^2 \leqslant 1, \\ & x_1 = x_0 - \gamma\nabla f(x_0), \\ & y_1 = y_0 - \gamma\nabla f(y_0). \end{aligned} \tag{4.3}$$

As written in (4.3), this problem involves an infinite-dimensional variable f. Our first step towards formulating (4.3) as an SDP consists of reformulating it by sampling f (i.e., evaluating its function value and gradient) at the two points where its gradient is evaluated:

$$\begin{aligned} \tau(\mu, L, \gamma) = \max_{\substack{d, f_{x_0}, f_{y_0} \\ x_0,x_1,g_{x_0} \in \mathbb{R}^d \\ y_0,y_1,g_{y_0} \in \mathbb{R}^d}} \quad & \|x_1 - y_1\|_2^2 \\ \text{s.t. } & d \in \mathbb{N}, \\ & \|x_0 - y_0\|_2^2 \leqslant 1, \\ & \exists f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d) : \begin{cases} f(x_0) = f_{x_0} & \nabla f(x_0) = g_{x_0} \\ f(y_0) = f_{y_0} & \nabla f(y_0) = g_{y_0} \end{cases} \\ & x_1 = x_0 - \gamma g_{x_0}, \\ & y_1 = y_0 - \gamma g_{y_0}, \end{aligned} \tag{4.4}$$

where we replaced the variable f by its discrete version, which we constrain to be "interpolable" (or "extendable") by a smooth strongly convex function over $\mathbb{R}^d$. To arrive at a tractable problem, we use the following interpolation (or extension) result.

Theorem 4.2.1. *(Taylor et al., 2017c, Theorem 4) Let I be an index set and $S = \{(x_i, g_i, f_i)\}_{i\in I}$ be such that $x_i, g_i \in \mathbb{R}^d$ and $f_i \in \mathbb{R}$ for all $i \in I$. There exists a function $F \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ such that $f_i = F(x_i)$ and $g_i = \nabla F(x_i)$ (for all $i \in I$) if and only if for all $i, j \in I$ we have*

$$f_i \geqslant f_j + \langle g_j; x_i - x_j\rangle + \tfrac{1}{2L}\|g_j - g_i\|_2^2 + \tfrac{\mu L}{2(L-\mu)}\|x_i - x_j - \tfrac{1}{L}(g_i - g_j)\|_2^2. \tag{4.5}$$

Using Theorem 4.2.1, we can formulate the problem of computing $\tau(\mu, L, \gamma)$ as a (nonconvex) quadratic problem:

$$\begin{aligned}
\max_{\substack{d, f_{x_0}, f_{y_0}\\ x_0, g_{x_0}\in\mathbb{R}^d\\ y_0, g_{y_0}\in\mathbb{R}^d}} \ & \|(x_0 - \gamma g_{x_0}) - (y_0 - \gamma g_{y_0})\|_2^2 \\
\text{s.t. } & d \in \mathbb{N}, \\
& \|x_0 - y_0\|_2^2 \leqslant 1, \\
& f_{y_0} \geqslant f_{x_0} + \langle g_{x_0}; y_0 - x_0\rangle + \tfrac{1}{2L}\|g_{y_0} - g_{x_0}\|_2^2 \\
& \qquad + \tfrac{\mu L}{2(L-\mu)}\|x_0 - y_0 - \tfrac{1}{L}(g_{x_0} - g_{y_0})\|_2^2, \\
& f_{x_0} \geqslant f_{y_0} + \langle g_{y_0}; x_0 - y_0\rangle + \tfrac{1}{2L}\|g_{y_0} - g_{x_0}\|_2^2 \\
& \qquad + \tfrac{\mu L}{2(L-\mu)}\|x_0 - y_0 - \tfrac{1}{L}(g_{x_0} - g_{y_0})\|_2^2.
\end{aligned} \tag{4.6}$$

Relying on a standard trick from semidefinite programming, one can convexify this problem using a Gram representation of the variable (this is due to maximization over d). That is, we formulate the problem using a positive semidefinite matrix $G \succcurlyeq 0$ defined as

$$G \triangleq \begin{pmatrix} \|x_0 - y_0\|_2^2 & \langle x_0 - y_0; g_{x_0}\rangle & \langle x_0 - y_0; g_{y_0}\rangle \\ \langle x_0 - y_0; g_{x_0}\rangle & \|g_{x_0}\|_2^2 & \langle g_{x_0}; g_{y_0}\rangle \\ \langle x_0 - y_0; g_{y_0}\rangle & \langle g_{x_0}; g_{y_0}\rangle & \|g_{y_0}\|_2^2 \end{pmatrix} \succcurlyeq 0. \tag{4.7}$$

Using this change of variable, we arrive to

$$\begin{aligned}
\max_{f_{x_0}, f_{y_0}, G} \ & G_{1,1} - 2\gamma(G_{1,2} - G_{1,3}) + \gamma^2(G_{2,2} + G_{3,3} - 2G_{2,3}) \\
\text{s.t. } & G \succcurlyeq 0, \\
& G_{1,1} \leqslant 1, \\
& f_{y_0} \geqslant f_{x_0} + \tfrac{1}{L-\mu}\left(\tfrac{\mu L}{2}G_{1,1} - LG_{1,2} + \mu G_{1,3} + \tfrac{1}{2}G_{2,2} - G_{2,3} + \tfrac{1}{2}G_{3,3}\right), \\
& f_{x_0} \geqslant f_{y_0} + \tfrac{1}{L-\mu}\left(\tfrac{\mu L}{2}G_{1,1} - \mu G_{1,2} + LG_{1,3} + \tfrac{1}{2}G_{2,2} - G_{2,3} + \tfrac{1}{2}G_{3,3}\right),
\end{aligned} \tag{4.8}$$

which can be solved numerically using standard tools, see, e.g., Diamond and Boyd (2016); MOSEK (2019); O'Donoghue et al. (2016). Using numerical and/or symbolical computations, one can then easily arrive at $\tau(\mu, L, \gamma) = \max\{(1 - L\gamma)^2, (1 - \mu\gamma)^2\}$ and hence that

$$\|x_1 - y_1\|_2^2 \leqslant \max\{(1 - L\gamma)^2, (1 - \mu\gamma)^2\}\|x_0 - y_0\|_2^2, \tag{4.9}$$

for all $d \in \mathbb{N}$, $f \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ and $x_0, y_0 \in \mathbb{R}^d$ when $x_1, y_1 \in \mathbb{R}^d$ are generated from gradient steps from respectively x_0 and y_0. In the next section, we show how to perform this analysis using PEPIT, which automates the sampling and SDP-modeling procedures. In more complex settings where more functions need to be sampled and/or more iterates have to be taken into account, avoiding those steps allows to largely limits the probability of making a mistake in the process of performing the worst-case analysis (numerically) while allowing to spare a significant amount of time in the process.

Remark 4.2.2 (Important ingredients for the SDP reformulations)**.** *To understand what* PEPIT *can do, it is crucial to understand which elements allowed to cast the worst-case analysis as such a semidefinite program (which is what we refer to as the "modeling" of the problem). In short, the SDP reformulation of the worst-case computation problem was made possible due to* 4 *main ingredients (see, e.g., (Taylor et al., 2017a, Section 2.2)):*

1. *the algorithmic steps can be expressed linearly in terms of the iterates and gradient values (i.e., step-sizes do not depend on the function at hand),*
2. *the class of functions has "interpolation condition"* [1] *that are linear in* G *and the function values,*
3. *the performance measure is linear (or convex piecewise linear) in* G *and the function values,*
4. *the initial condition is linear in* G *and the function values.*

Those ingredients allow the use of PEPs much beyond the simple setup of this section. That is, PEPs apply for performing worst-case analyses involving a variety of first-order oracles, initial conditions, performance measures, and problem classes (see Section 4.3 for the general modeling of the problem, and Section 4.4 for a non-exhaustive list of cases that are covered).

4.2.2 Code

In the previous section, we introduced the PEP and SDP modeling steps for computing a tight worst-case contraction factor for Gradient descent in the form (4.2). Although this particular SDP (4.8) might be solved analytically, many optimization methods lead to larger SDPs with more complicated structures. In general, we can reasonably only hope to solve them numerically. In the following lines, we describe how to use PEPIT for computing a *contraction factor* without explicitly going into the modeling steps. Compared to previous section, we allow ourselves to perform $n \in \mathbb{N}$ iterations and compute the smallest possible value of $\tau(\mu, L, \gamma, n)$ such that

$$\|x_n - y_n\|_2^2 \leqslant \tau(\mu, L, \gamma, n)\|x_0 - y_0\|_2^2, \tag{4.10}$$

where x_n and y_n are computed from n iterations of Gradient descent with step-size γ starting from respectively x_0 and y_0. As illustrated in the previous section for the case $n = 1$, computing the smallest possible such $\tau(\mu, L, \gamma, n)$ is equivalent to computing the worst-case value of $\|x_n - y_n\|_2^2$ under the constraint that $\|x_0 - y_0\|_2^2 \leqslant 1$ (note that we naturally have that $\tau(\mu, L, \gamma, n) \leqslant (\tau(\mu, L, \gamma, 1))^n$). This is what we do in the following lines using PEPIT.

Imports. Before going into the example, we have to include the right PYTHON imports. For this example, it is necessary to perform two imports.

```
1 from PEPit import PEP
2 from PEPit.functions import\
3     SmoothStronglyConvexFunction
```

[1] Interpolation conditions characterize the **existence** of a function (that has particular function values and gradients at given points) in the considered class by a list of constraints on those gradients, points, and function values. Such interpolation theorems (see, e.g., Theorem 4.2.1) have been obtained in the literature for various problem classes, see, e.g., Taylor et al. (2017c,a); Ryu et al. (2020).

Initialization of PEPIT. First, we set the stage by initializing a PEP object. This object allows manipulating the forthcoming ingredients of the PEP, such as functions and iterates.

```
problem = PEP()
```

For the sake of the example, let us pick some simple values for the problem class and algorithmic parameters, for which we perform the worst-case analysis.

```
L = 1.           # Smoothness parameter
mu = .1          # Strong convexity parameter
gamma = 1. / L   # Step-size
n = 1            # Number of iterations
```

Specifying the problem class. Second, we specify our working assumptions on the function to be optimized and instantiate a corresponding object. Here, the minimization problem at hand was of the form (4.1) with a smooth strongly convex function.

```
# Declare an L-smooth mu-strongly convex function
# named "func"
func = problem.declare_function(
    SmoothStronglyConvexFunction,
    mu=mu,   # Strong convexity param.
    L=L)     # Smoothness param.
```

Algorithm initialization. Third, we can instantiate the starting points for the two gradient methods that we will run, and specify an *initial condition* on those points. To this end, two starting points x_0 and y_0 are introduced, one for each trajectory, and a bound on the initial distance between those points is specified as $\|x_0 - y_0\|^2 \leqslant 1$.

```
# Declare two starting points
x_0 = problem.set_initial_point()
y_0 = problem.set_initial_point()

# Initial condition ||x_0 - y_0||^2 <= 1
problem.set_initial_condition((x_0 - y_0) ** 2 <= 1)
```

Algorithm implementation. In this fourth step, we specify the algorithm in a natural format. In this example, we simply use the iterates (which are PEPIT objects) as if we had to implement Gradient descent in practice using a simple loop.

```
# Initialize the algorithm
x = x_0
y = y_0
# Run n steps of the GD method for the two sequences
for _ in range(n):
    # Replace x and y with their next iterate
    x = x - gamma * func.gradient(x)   # call to f'(x)
    y = y - gamma * func.gradient(y)   # call to f'(y)
```

Setting up a performance measure. It is crucial for the worst-case analysis to specify the *metric* for which we wish to compute a worst-case performance. In this example, we wish to compute the worst-case value of $\|x_n - y_n\|^2$, which we specify as follows.

```
29 # Set the performance metric to the distance
30 # ||x_n - y_n||^2
31 problem.set_performance_metric((x-y)**2)
```

Solving the PEP. The last natural stage in the process is to solve the corresponding PEP. This is done via the following line, which will ask PEPIT to perform the modeling steps and to call an appropriate SDP solver (which should be installed beforehand) to perform the worst-case analysis.

```
32 # Solve the PEP
33 pepit_tau = problem.solve()
```

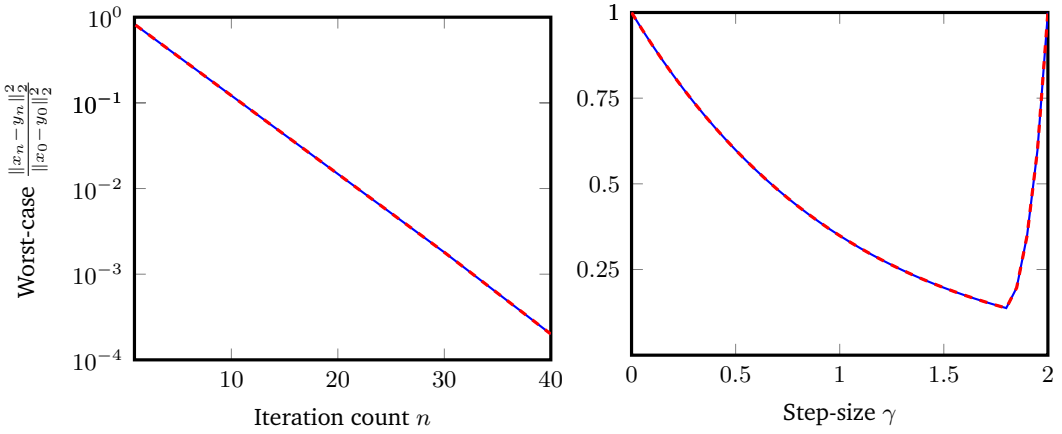
Output. Running these pieces of code (see PEPit/examples/ for the complete example) for some specific values of the parameters $n = 1$, $L = 1$, $\mu = .1$ and $\gamma = 1$, one obtains the following output.

```
1  (PEPit) Setting up the problem: size of the Gram matrix: 4x4
2  (PEPit) Setting up the problem: performance measure is the minimum of 1 element(s)
3  (PEPit) Setting up the problem: Adding initial conditions
4                                  and general constraints ...
5  (PEPit) Setting up the problem: initial conditions and general constraints
6                                  (1 constraint(s) added)
7  (PEPit) Setting up the problem: interpolation conditions for 1 function(s)
8              Function 1 : Adding 2 scalar constraint(s) ...
9              Function 1 : 2 scalar constraint(s) added
10 (PEPit) Setting up the problem: additional constraints for 0 function(s)
11 (PEPit) Compiling SDP
12 (PEPit) Calling SDP solver
13 (PEPit) Solver status: optimal (wrapper:cvxpy, solver: MOSEK);
14                        optimal value: 0.8100000029203449
15 (PEPit) Primal feasibility check:
16         The solver found a Gram matrix that is positive semi-definite
17         up to an error of 1.896548260018477e-09
18         All the primal scalar constraints are verified
19         up to an error of 3.042855638898251e-09
20 (PEPit) Dual feasibility check:
21         The solver found a residual matrix that is positive semi-definite
22         All the dual scalar values associated with inequality constraints
23         are nonnegative
24 (PEPit) The worst-case guarantee proof is perfectly reconstituted
25         up to an error of 4.0078754315331366e-08
26 (PEPit) Final upper bound (dual): 0.8100000036427537
27         and lower bound (primal example): 0.8100000029203449
28 (PEPit) Duality gap: absolute: 7.224087994472939e-10
29                       and relative: 8.918627121515396e-10
```

Note that the size of the SDP is larger than that of Section 4.2 (4×4 instead of 3×3 in (4.7)) because the modeling step is done in a slightly more generic way, which might

not be exploiting all specificities of the problem at hand (see formulation in Section 4.3.1). For more complete examples of worst-case analyses using PEPIT, see Section 4.5.

It is also possible to run the code for different values of the parameters, as exemplified in Figure 4.1. This simple example allows us to observe that numerical values obtained from PEPIT match the worst-case guarantee (4.1a), and to optimize the step-size numerically in Figure 4.1b.



(a) Worst-case guarantee and theoretical tight bound as a function of the iteration count n for $\gamma = \frac{1}{L}$.

(b) Worst-case guarantee and theoretical tight bound as a function of the step-size γ, at iteration $n = 5$.

Figure 4.1: Comparison: worst-case guarantee from PEPIT (**plain blue**) and theoretical tight worst-case bound (4.9) for Gradient descent in terms of $\frac{\|x_n-y_n\|_2^2}{\|x_0-y_0\|_2^2}$ (**dashed red**). Problem parameters fixed to $\mu = 0.1$ and $L = 1$.

4.3 PEPIT code structure and semidefinite formulation

This section provides the general semidefinite program (SDP) that is formulated by the package, as well as its relationship with the code. As already underlined, PEPIT precisely aims at providing simple ways to model performance estimation problems (PEPs) by abstracting the coding of those SDPs. Then, PEPIT passes the SDP either (i) to CVXPY Diamond and Boyd (2016), thereby benefiting from all SDP solvers that are interfaced with it (such as O'Donoghue et al. (2016); MOSEK (2019)), or (ii) directly to MOSEK MOSEK (2019).

Before going into the implementation details of PEPit, note that executing the codes from Section 3.2 to 3.5 requires the following imports.

```
# For Section 3.2 to Section 3.5
import PEPit
import PEPit.functions

# For Section 3.3 and Section 3.5 we also need:
from PEPit.functions import\
    ConvexFunction,\
    ConvexIndicatorFunction,\
    SmoothConvexFunction

```

```
# For the examples of Section 3.3:
from PEPit.primitive_steps import\
    proximal_step,\
    linear_optimization_step,\
    inexact_gradient_step
```

4.3.1 Semidefinite formulation

The package formulates SDPs of the form

$$\begin{aligned} \max_{\tau\in\mathbb{R},G\in\mathbb{S}^{n_p},H\in\mathbb{R}^{n_h}} \quad & \tau \\ \text{s.t. } & G \succcurlyeq 0 \\ & \tau - \operatorname{Tr}(A_i G) - a_i^T H - \alpha_i \leqslant 0 \quad \forall i \in I_1 \\ & \left[\operatorname{Tr}(B_{j,k,i}G) + b_{j,k,i}^T H + \beta_{j,k,i}\right]_{1\leqslant j,k\leqslant n_i} \succcurlyeq 0 \quad \forall i \in I_2, \end{aligned} \tag{4.11}$$

with $n_h, n_p \in \mathbb{N}$, some index sets I_1 and I_2 in $\mathbb{N}$, and sets of problem parameters $\{(A_i, a_i, \alpha_i)\}_{i\in I_1}$, and $\{(B_{j,k,i}, b_{j,k,i}, \beta_{j,k,i})\}_{1\leqslant j,k\leqslant n_i, i\in I_2}$ ($n_i \in \mathbb{N}_{>0}$ for all $i \in I_2$) of appropriate dimensions, which are constructed from the algorithm and the class of problems at hand (and hence all depend on the parameters of the algorithm and of those of the class of problems) for computing appropriate worst-case scenarios.

Formulating such an SDP is usually cumbersome and relatively error-prone, and the role of PEPIT is to generate all those parameters in a user-friendly way. That is, PEPIT parses more natural mathematical object descriptions fed by the user. For doing that, the problem is described not in terms of the SDP variables (G, H) but rather in terms of another couple (P, H) with the following structure:

$$P \triangleq [p_1,\, p_2,\, \ldots,\, p_{n_p}], \qquad H \triangleq [h_1,\, h_2\, \ldots,\, h_{n_h}], \tag{4.12}$$

for some $\{p_i\}_{1\leqslant i\leqslant n_p} \subset \mathbb{R}^d$ for some $d \in \mathbb{N}$, and $\{h_i\}_{1\leqslant i\leqslant n_h} \subset \mathbb{R}$. Hence $P \in \mathbb{R}^{d\times n_p}$ and $H \in \mathbb{R}^{n_h}$. The relationship with (4.11) is that G can be constructed without loss of generality as $G \triangleq P^T P \succcurlyeq 0$.

4.3.2 Base PEPIT objects

Using previous notations, PEPIT allows the user to formulate the problem in terms of (P, H) instead of (G, H), which turns out much more natural for describing many algorithms and problem classes. The base working procedure is as follows, which we detail in the next lines:

- each p_i corresponds to a base PEPIT.POINT object (referred to as a *leaf* element). Such objects can be added and subtracted together, and scaled by real values for forming new objects PEPIT.POINT (which are then combinations of *leaf* elements). PEPIT.POINT objects can be understood as elements of $\mathbb{R}^d$, the space of iterates/gradients.

- Each h_i corresponds to a base PEPIT.EXPRESSION object (referred to as a *leaf* element). Such objects can be added and subtracted together, but also scaled by real values for forming new objects PEPIT.EXPRESSION (which are combinations of *leaf* elements).

PEPIT.EXPRESSION objects can be understood as elements of $\mathbb{R}$, such as, for instance, (scalar) function values.

- It is possible to compute dot products of PEPIT.POINT objects (e.g., $p_i^T p_j$ for some $1 \leqslant i, j \leqslant n_p$), resulting in PEPIT.EXPRESSION objects.
- Comparing two PEPIT.EXPRESSION objects with an operator in $\{=, \leqslant, \geqslant\}$ leads to a PEPIT.CONSTRAINT object. Similarly, PEPIT.EXPRESSION can be gathered in arrays to form PEPIT.PSD_MATRIX objects for formulating semidefinite constraints.

Example 4.3.1. *Following up on the notations from Section 4.2 for describing one iteration of Gradient descent, a possibility is to think of p_1 as corresponding to some x_0, of p_2 as a corresponding gradient g_{x_0}, and of h_1 as the corresponding function value f_{x_0}. For a unit step-size, one can form $x_1 = p_1 - p_2$ as follows.*

```
x_0 = PEPit.Point()  # a leaf PEPit.Point (p_1)
g_x_0 = PEPit.Point()  # a leaf PEPit.Point (p_2)
# a leaf PEPit.Expression (h_1):
f_x_0 = PEPit.Expression()

x_1 = x_0 - g_x_0  # x_1 is a PEPit.Point
g_x_1 = PEPit.Point()  # a leaf PEPit.Point (p_3)
# a leaf PEPit.Expression (h_2):
f_x_1 = PEPit.Expression()
```

It is important to note that each call to PEPIT.POINT() *increments n_p (dimension G in (4.11)). Similarly,* PEPIT.EXPRESSION() *increments n_h. It is cheaper to solve a problem with as few leaf points and leaf expressions as possible.*

For formulating the objective, initial conditions, and interpolation constraints, we use PEPIT.EXPRESSION *objects, which we compare together for forming* PEPIT.CONSTRAINT *objects. For instance, interpolation conditions for convex functions can be formulated via* PEPIT.CONSTRAINT *objects:*

```
# this is a PEPit.Expression object:
expr = f_x_0 + g_x_0 * (x_1 - x_0)
# those are two PEPit.Constraint objects:
cons_1 = (f_x_0 + g_x_0 * (x_1 - x_0) <= f_x_1)
cons_2 = (f_x_1 + g_x_1 * (x_0 - x_1) <= f_x_0)
```

Once all required points, expressions, and constraints are associated with a PEP (as in the example of Section 4.2.2 or in the following lines), PEPIT takes care of formulating the appropriate index sets I_1, and I_2 as well as all problem parameters for (4.11): $\{(A_i, a_i, \alpha_i)\}_{i \in I_1}$, and $\{(B_{j,k,i}, b_{j,k,i}, \beta_{j,k,i})\}_{1 \leqslant j,k \leqslant n_i, i \in I_2}$ for feeding (4.11) to SDP solvers Diamond and Boyd (2016); O'Donoghue et al. (2016); MOSEK (2019). As we can see, specifying constraints in this form is relatively appealing, due to its similarity with natural mathematical statements. However, specifying a large number of such constraints remains relatively cumbersome. Therefore, PEPIT relies on a few additional structures and aliases that allow generating blocks of constraints in an abstract way.

4.3.3 Main PEPIT simplifying abstractions and aliases

Two key abstractions appearing in PEPIT are the *functions* (which can also be manipulated algebraically) and *oracles* (or *primitive steps*, which are simple routines). In both cases, those structures were thought for being able to easily add new types of functions and oracles, as a user.

PEPIT functions. Among the most important building blocks simplifying the formulation of the constraints in (4.11), PEPIT.FUNCTION objects are particularly important. They allow generating large numbers of constraints by remaining close to clean mathematical statements. Their most important characteristics are as follows:

- PEPIT.FUNCTION() creates a new *leaf* function,
- each leaf function contains a list of triplet $\{(x_i, g_i, f_i)\}_i$ which corresponds to the sampled version of the function in the form (points, gradients, function values), as presented in Section 4.2.
- Each PEPIT.FUNCTION object F is featured with a F.ADD_POINT(TRIPLET) method, which adds a triplet to the list of samples associated to F. By relying on appropriate abstractions (exemplified later), the users are expected to almost never use this method explicitly.
- Each PEPIT.FUNCTION object F contains a F.ADD_CLASS_CONSTRAINTS() method that generate the list of interpolation PEPIT.CONSTRAINT associated to the sampled version of F. This method is never explicitly used by the user but allows PEPIT to translate the mathematical statements (description of the function) to actual constraints.
- Each PEPIT.FUNCTION can also be scaled by scalar values and added or subtracted to other PEPIT.FUNCTION for creating new PEPIT.FUNCTION objects.

Functions are also featured with a number of aliases that allow to easily create specific PEPIT.POINT and PEPIT.EXPRESSION objects associated with the function (such as accessing the gradient/value at a specific point, or accessing an optimal point of a function).

Example 4.3.2 (Base operations with functions)**.** *This example shows that we can manipulate functions to create new functions.*

```
f = PEPit.Function()  # a leaf PEPit.Function
h = PEPit.Function()  # a leaf PEPit.Function
F = 2 * f + h  # a PEPit.Function
```

We can also call base function operations on functions that are constructed by combining leaf *functions.*

```
x = PEPit.Point()  # this is p_1
g_x = PEPit.Point()  # this is p_2
f_x = PEPit.Expression()  # this is h_1

F.add_point((x, g_x, f_x))
# internally creates p_3, and h_2
```

```
# and adds one point to the list of f,
# and one to the list of h so that
# the weighted sums of gradients and
# function values at x are correct.
```

However, for simplifying the usage of PEPIT*, one should essentially avoid directly using the method* ADD_POINT*, and rather rely on more readable statements. In that regard, the following piece of code is equivalent to the previous one but relies on more readable operations.*

```
f = PEPit.Function()  # a leaf PEPit.Function
h = PEPit.Function()  # a leaf PEPit.Function
F = 2 * f + h  # a PEPit.Function

x = PEPit.Point()  # this is p_1 (arbitrary point)

g_x = F.gradient(x)  # g_x is the gradient of F at x
f_x = F(x)  # f_x is F(x)
```

Once the functions are features with the appropriate list of points corresponding to their sample versions, those objects will be used to generate the list of interpolation constraints (and the corresponding weight matrices). For creating a class that features a new type of interpolation conditions, it suffices to create a new class that inherits PEPIT.FUNCTION and implements the operation ADD_CLASS_CONSTRAINTS with the appropriate interpolation constraints (see https://github.com/PerformanceEstimation/PEPit/tree/master/PEPit/functions for examples).

Let us now mention an important class of convenient abstraction that belongs to the heart of PEPIT's philosophy.

PEPIT primitive steps (or oracles). PEPIT *primitive steps* (or *oracles*) are essentially simple aliases defining notational shortcuts for some algorithmic operations, rendering them closer to their mathematical abstractions. They generally consist of the appropriate creations (and constraining) of points and expressions, for instance by adding appropriate triplets to sampled versions of the functions under consideration. Let us provide a few examples.

Example 4.3.3 (Proximal operators.)**.** *In this example, we provide two ways to use proximal operators within* PEPIT*. The first one uses the base* PEPIT *methods for doing this, and the second one relies on the more readable* PROXIMAL_STEP *that is provided as a primitive step of* PEPIT*. Those two ways are computationally equivalent, though different in terms of readability. Let us recall that the proximal operation (with unit step-size) associated to f is given by*

$$x_1 = \mathrm{prox}_f(x_0) \triangleq \mathrm{argmin}_x \left\{ f(x) + \tfrac{1}{2}\|x - x_0\|^2 \right\}.$$

For any (closed, proper) convex function f, this operation is well-defined and amounts to an implicit subgradient *operation:*

$$x_1 = x_0 - g_1,$$

with $g_1 \in \partial f(x_1)$.

```
# f is a (closed, proper) convex function
f = PEPit.functions.ConvexFunction()
x0 = PEPit.Point()  # is a point

# how to construct x_1 = prox_f (x_0)?
# (i) initiate some g1 and f1
g1 = PEPit.Point()
f1 = PEPit.Expression()

# (ii) form x1 using x0 and g1
x1 = x0 - g1

# (iii) constrain g1 to be a subgradient of f at x1
f.add_point((x1, g1, f1))
```

Equivalently, using PROXIMAL_STEP*, we have:*

```
# f is a (closed, proper) convex function
f = PEPit.functions.ConvexFunction()
x0 = PEPit.Point()  # is a point

x1, g1, f1 = proximal_step(x0, f, gamma=1)
# note: gamma is a step-size, set to 1 in the example.
```

Example 4.3.4 (Linear optimization oracles.)**.** *This operation is at the core of the Frank-Wolfe (a.k.a. conditional gradient) method, and consists in, given a (closed, convex) domain $\mathcal{K}$ (whose indicator function is denoted by $i_{\mathcal{K}}$) and a search direction d_0, in computing a solution to*

$$x_1 \in \operatorname{argmin}_x d_0^T x + i_{\mathcal{K}}(x),$$

which is mathematically equivalent to writing $-d_0 \in \partial i_{\mathcal{K}}(x_1)$. In PEPIT*, this can also easily be coded as follows.*

```
# ind is a (closed) convex indicator function
ind = PEPit.functions.ConvexIndicatorFunction()
d0 = PEPit.Point()

# how to construct a solution to min_x d0*x + ind(x) ?
# (i) initiate a new point and an expression
x1 = PEPit.Point()
f1 = PEPit.Expression()

# (ii) Constrain -d0 to be a subgradient
# of the indicator at x1
ind.add_point((x1, -d0, f1))
```

Using abstraction again, this code is equivalent to

```
# ind is a (closed) convex indicator function
```

```
ind = PEPit.functions.ConvexIndicatorFunction()
d0 = PEPit.Point()

# how to construct a solution to min_x d0*x + ind(x) ?
x1, _, f1 = linear_optimization_step(d0, ind)
```

Example 4.3.5 (Using approximate gradients.)**.** *Another standard operation in first-order optimization consists in using* approximate *gradient values. For instance, for computing some* x_1 *using a gradient iteration (with unit step-size) with an approximate gradient* $\tilde{d}_0 \approx \nabla f(x_0)$ *for some appropriate function* f*:*

$$x_1 = x_0 - \tilde{d}_0,$$

with $\tilde{d}_0$ *being an* ε *approximation to* $\nabla f(x_0)$ *in the following sense:* $\|\tilde{d}_0 - \nabla f(x_0)\| \leqslant \varepsilon \|\nabla f(x_0)\|$*, say with* $\varepsilon = 0.1$*.*

```
# f is a 1-smooth convex function
f = PEPit.functions.SmoothConvexFunction(L=1)
x0 = PEPit.Point()
epsilon = .1

# how to construct a x1?
# (i) initiate a d0 and the gradient of x0
g0 = f.gradient(x0)
d0 = PEPit.Point()
f.add_constraint((d0 - g0) ** 2 <= epsilon * g0 ** 2)

x1 = x0 - d0
```

This can equivalently be done using INEXACT_GRADIENT_STEP*, as follows.*

```
# f is a 1-smooth convex function
f = PEPit.functions.SmoothConvexFunction(L=1)
x0 = PEPit.Point()

# how to construct a x1?
from PEPit.primitive_steps\
    import inexact_gradient_step

x1, d0, _ = inexact_gradient_step(x0, f, gamma=1,
                                  epsilon=.1,
                                  notion='relative')
```

To conclude this section, PEPit's philosophy is to contain many *abstract* routines that can be associated with simple mathematical statements. Taken separately, those routines are relatively simple and can easily be created or modified by the users.

4.3.4 The objective function of the PEP: performance metrics

A key point that we did not mention so far concerns the objective function of (4.11). It is handled by what is referred to as *performance metrics* in PEPit. In the SDP formula-

tion (4.11), they correspond to the index set I_1 and the set of parameters $\{(A_i, a_i, \alpha_i)\}_{i\in I_1}$, and they are handled by calling the SET_PERFORMANCE_METRIC method (which takes a PEPIT.EXPRESSION as sole argument) associate to a PEP object. By introducing the variable τ in (4.11), the objective of the PEP corresponds to the minimum value of all specified performance metrics.

Example 4.3.6. *This example shows how to specify an objective function (a.k.a.* performance metric*) within* PEPIT.

```
problem = PEPit.PEP()

p1 = PEPit.Point()
p2 = PEPit.Point()
h1 = PEPit.Expression()

# set the PEP objective as the minimum value among
# ||p1||^2, ||p2||^2, and h1.
problem.set_performance_metric(p1**2)
problem.set_performance_metric(p2**2)
problem.set_performance_metric(h1)
```

4.3.5 Formulating and solving the PEP

As a final stage for formulating (4.11), we need to gather all functions and constraints together and reformulate in terms of (4.11). For doing that, PEPIT relies on the PEP object (as provided in Section 4.2.2) as follows:

```
problem = PEPit.PEP()
```

which is used for centralizing all the information about the PEP at hand. In particular, it is a good practice to avoid using PEPIT.POINT() directly, and to rather use PROBLEM.SET_INITIAL_POINT(). Similarly, new functions should be declared through the PEP object using the DECLARE_FUNCTION method:

```
# declares a convex (closed, proper) function h
h = problem.declare_function(ConvexFunction)

# declares a convex (closed) indicator ind
ind = problem.declare_function(ConvexIndicatorFunction)

# declares a 1-smooth convex function f
f = problem.declare_function(SmoothConvexFunction,
                             L=1)
```

For modeling (4.11), we gather all PEPIT.CONSTRAINT objects that are associated with PROBLEM through the different abstractions used in the code. For instance, the PEPIT.PEP object PROBLEM will call the ADD_CLASS_CONSTRAINTS method of all functions involved in the PEP. Once this is done, PEPIT generates all appropriate matrices and index sets for framing the problem as an SDP in the form (4.11), and passes it to either CVXPY Diamond

and Boyd (2016) or directly to MOSEK MOSEK (2019), before performing a few post-processing steps.

4.3.6 Post-processing

Once the PEP is solved numerically, we need to manipulate its output, either for constructing proofs (on the dual PEP side), or counter-examples (on the primal PEP side).

Dual reconstructions. Dual values associated with the different constraints play an important role, as they allow us to construct mathematical proofs.

In the following, we assume that *func* has been defined as a Function used in the PEP, that *constraints_list* has been defined as a list of Constraint objects involved in the PEP and that the PEP has been solved through the PEP.SOLVE() method.

One of the important features of PEPit is to autonomously deal with interpolation constraints for the different functions involved in the PEP. This way, the attribute *list_of_class_constraints* enables to access the list of Constraint objects encoding the interpolation constraints of *func*.

Besides, each dual value can be accessed through the EVAL_DUAL() method of the associated Constraint object:

```
for constraint in func.list_of_class_constraints:
    dual_value = constraint.eval_dual()
    print(dual_value)
```

Moreover, to ease the reconstruction of the proof, and avoid mistakes by associating a constraint to the wrong dual value, a user can name a constraint through the SET_NAME() method and access it later on through the GET_NAME() method. Note it is the responsibility of the user to set a name to the constraints to be able to recover it later.

```
for constraint in constraints_list:
    constraint_name = constraint.get_name()
    dual_value = constraint.eval_dual()
    print(constraint_name, dual_value)
```

Since PEPit deals with interpolation constraints without any intervention from the user side, a short description of each of those constraints is set by default, based on Points'names and *func*'s name which can also be set through a SET_NAME method.

Finally, a user can also obtain all the dual values associated with the interpolation constraints of a function at once, using the GET_CLASS_CONSTRAINTS_DUALS method as

```
# assuming func has been defined
# as a Function object involved in the PEP
# and that the PEP has been solved.

tables = func.get_class_constraints_duals()
```

This method returns a dictionary whose values are pandas.DataFrames, offering great readability. Completing the example of Section 4.2.2 (this time for 2 steps) with this feature, we obtain the following code:

```
import numpy as np
```

```
2
3  from PEPit import PEP
4  from PEPit.functions import\
5      SmoothStronglyConvexFunction
6
7  # We set the parameter of the problem
8  # Here we study the contraction of 1 step
9  # of the GD method with step 1/L on the class of
10 # L=1 smooth and mu=.1 strongly convex functions.
11
12 L = 1.           # Smoothness parameter
13 mu = .1          # Strong convexity parameter
14 gamma = 1. / L   # Step-size
15 n = 2            # Number of iterations
16
17 # Instantiate the PEP object
18 problem = PEP()
19
20 # Declare an L-smooth mu-strongly convex function
21 # named "func"
22 func = problem.declare_function(
23     SmoothStronglyConvexFunction,
24     mu=mu,      # Strong convexity param.
25     L=L,        # Smoothness param.
26     name="f")   # Name
27
28 # Declare two starting points
29 x_0 = problem.set_initial_point(name="x_0")
30 y_0 = problem.set_initial_point(name="y_0")
31
32 # Initial condition ||x_0 - y_0||^2 <= 1
33 problem.set_initial_condition((x_0 - y_0) ** 2 <= 1)
34
35 # Initialize the algorithm
36 x = x_0
37 y = y_0
38 # Run n steps of the GD method for the two sequences
39 for i in range(n):
40
41     # Replace x and y with their next iterates
42     x = x - gamma * func.gradient(x)   # call to f'(x)
43     x.set_name("x_{}".format(i+1))
44
45     y = y - gamma * func.gradient(y)   # call to f'(y)
46     y.set_name("y_{}".format(i+1))
47
48 # Set the performance metric to the distance
```

```
# ||x_n - y_n||^2
problem.set_performance_metric((x-y)**2)

# Solve the PEP
pepit_tau = problem.solve()

# By linearly combining the interpolation constraints
# with the right coefficients, we can prove
# ||x_n - y_n||^2 <= pepit_tau ||x_0 - y_0||^2
# The coefficient we need are the dual values
# of the interpolation constraints of func.
tables = func.get_class_constraints_duals()

# A user can display the dictionary as is,
# or can access one specific table by their name.
# Those names are intuitive, yet to be known,
# for example by displaying the keys of tables.
# Here we use the only key of this dictionary.
table = tables["smoothness_strong_convexity"]

print("\nDual values associated with"
      " interpolation constraints:")
print(table.astype(dtype=np.float16))
```

Running this code outputs the following message:

```
(PEPit) Setting up the problem: size of the Gram matrix: 6x6
(PEPit) Setting up the problem: performance measure is the minimum of 1 element(s)
(PEPit) Setting up the problem: Adding initial conditions
                                and general constraints ...
(PEPit) Setting up the problem: initial conditions and general constraints
                                (1 constraint(s) added)
(PEPit) Setting up the problem: interpolation conditions for 1 function(s)
                         Function 1 : Adding 12 scalar constraint(s) ...
                         Function 1 : 12 scalar constraint(s) added
(PEPit) Setting up the problem: additional constraints for 0 function(s)
(PEPit) Compiling SDP
(PEPit) Calling SDP solver
(PEPit) Solver status: optimal (wrapper:cvxpy, solver: MOSEK);
                       optimal value: 0.6561000087150534
(PEPit) Primal feasibility check:
        The solver found a Gram matrix that is positive semi-definite
        up to an error of 2.584803430062655e-09
        All the primal scalar constraints are verified
        up to an error of 4.590572921792102e-09
(PEPit) Dual feasibility check:
        The solver found a residual matrix that is positive semi-definite
        All the dual scalar values associated with inequality constraints
        are nonnegative up to an error of 3.8932973849815053e-10
(PEPit) The worst-case guarantee proof is perfectly reconstituted
        up to an error of 1.349705540942825e-07
```

```
(PEPit) Final upper bound (dual): 0.6561000067100656
        and lower bound (primal example): 0.6561000087150534
(PEPit) Duality gap: absolute: -2.004987731396568e-09
                     and relative: -3.055917855150253e-09

Dual values associated with interpolation constraints:
IC_f        x_0       y_0       x_1       y_1
x_0    0.000000  1.458008  0.000000  0.000000
y_0    1.458008  0.000000  0.000000  0.000000
x_1   -0.000000  0.000000  0.000000  1.799805
y_1    0.000000 -0.000000  1.799805  0.000000
```

Primal reconstructions. In order to construct an example of a problem on which the algorithm behaves as badly as possible, all PEPIT.POINT, all PEPIT.EXPRESSION and all PEPIT.CONSTRAINT objects can be conveniently evaluated through the EVAL() method. Moreover, their NAME attribute can help to sort them. PEPIT also offers the possibility to search for simpler, potentially low-dimensional problem instances via the trace norm Recht et al. (2010) or the logdet Fazel et al. (2003) heuristics (aiming at finding low-rank feasible matrices G for the problem (4.11) while keeping the same objective value). Those post-processing steps can be accessed via the option of the SOLVE method (see https://pepit.readthedocs.io/en/latest/api/main_modules.html#pep). Examples of such usages can be found in the documentation at https://pepit.readthedocs.io/en/latest/examples/j.html.

4.4 PEPIT: general overview and content

In this section, we go back to the mathematical content of the toolbox and describe the various choices of (i) elementary oracles used in algorithms, (ii) problem or function classes, (iii) performance measures, and (iv) initial conditions, that are naturally handled by PEPIT. PEPIT also allows studying methods for monotone inclusions and fixed point problems, but we do not cover them in this summary. In the optimization setting, the minimization problem under consideration has the form

$$F_\star \triangleq \min_{x\in\mathbb{R}^d} \left\{ F(x) \equiv \sum_{i=1}^{K} f_i(x) \right\}, \tag{4.13}$$

for some $K \in \mathbb{N}$ and where each f_i is assumed to belong to some class of functions denoted by $\mathcal{F}_i$, encoding our working assumptions on f_i. We further assume the algorithms to gather information about the functions $\{f_i\}_i$ only via black-box oracles such as gradient or proximal evaluations.

Black-box oracles. The base black-box optimization oracles/operations available in PEPIT are the following:

- (sub)gradient steps,
- proximal and projection steps,
- linear optimization (or conditional) steps.

PEPIT also allows for their slightly more general approximate/inexact and Bregman (or mirror) versions. Those oracles might be combined with a few other operations, such as exact line-search procedures, as detailed in Table 4.1.

Problem classes. A few base classes of functions are readily available within the package (see Table 4.2 for further details) such as:

Table 4.1: Main base primitive steps (oracles) included in PEPIT. Appropriate references are provided in the corresponding documentation. Some oracles are overlapping and are present for promoting a better readability of the code and for numerical efficiency. Variations around the present oracles can be added at will. For each oracle, x_+ denotes the output of the oracles; the other elements are either input of the oracles or intermediary optimization variables.

Oracle name	Description	Tightness
Subgradient step	$x_+ = x - \gamma g$ with $g \in \partial f(x)$	✔
Epsilon-subgradient step	$x_+ = x - \gamma g$ with $g \in \partial_\varepsilon f(x)$	✔
Inexact gradient step	$x_+ = x - \gamma g$ with $g \approx_\varepsilon \nabla f(x)$ for some notion "$\approx_\varepsilon$" of approximation.	✔
Exact line-search step	$x_+ = \underset{z \in x + \mathrm{span}\{d_i, i \in [\![1,T]\!]\}}{\arg\min} f(z)$	✖
Proximal step	$x_+ = \underset{z}{\arg\min} \left\{ \gamma f(z) + \frac{1}{2}\|z - x\|^2 \right\}$	✔
Inexact proximal step	$x_+ \approx_\varepsilon \underset{z}{\arg\min} \left\{ \gamma f(z) + \frac{1}{2}\|z - x\|^2 \right\}$ for some notion "$\approx_\varepsilon$" of approximation.	✔
Bregman gradient step	$x_+ = \underset{z}{\arg\min} \left[\langle \nabla f(x); z - x \rangle + \frac{1}{\gamma} D_h(z; x) \right]$	✔
Bregman proximal step	$x_+ = \underset{z}{\arg\min} \left[f(z) + \frac{1}{\gamma} D_h(z; x) \right]$	✔
Linear optimization step	$x_+ = \underset{z \mid \mathrm{ind}(z)=0}{\arg\min} \langle \mathrm{dir}; z \rangle$	✔

- convex functions within different classes of assumptions possibly involving bounded gradients (Lipschitz functions), bounded domains, smoothness, and/or strong convexity. Those assumptions might be combined when compatible.
- Convex indicator functions, possibly with a bounded domain.
- Smooth nonconvex functions.
- Convex and quadratically upper bounded functions.
- Quadratic functions.

Beyond the pure optimization setting, PEPIT also allows using operators (see Table 4.3) within different classes of assumptions (namely: nonexpansive, Lipschitz, cocoercive, maximally monotone,

and strongly monotone operators) for studying first-order methods for monotone inclusions and variational inequalities.

Table 4.2: Some base function classes included in PEPIT, detailed in the documentation. Default functional classes within PEPIT. Some classes are overlapping and are present only to promote a better readability of the code.

Function class name	Tightness
Convex (closed, proper) functions	✔
Convex (closed, proper) Lipschitz-continuous functions	✔
Convex (closed, proper) indicator functions	✔
Convex support functions	✔
Smooth strongly convex functions	✔
Smooth convex functions	✔
Smooth (possibly nonconvex) functions	✔
Smooth convex Lipschitz functions	✔
Strongly convex functions	✔
Convex quadratically upper-bounded functions	✔
Restricted secant inequality and error bound	✔
Smooth strongly convex quadratic functions	✔
Smooth convex function by block	✖

Performance measures and initial conditions. An important degree of freedom of the package is to allow a large panel of performance measures and initial conditions. Essentially, everything that can be expressed linearly (or slightly beyond) in function values and quadratically in gradient/iterates (i.e., linear in the Gram representation of Section 4.2) might be considered. Following the notation of Section 4.2.2 (and denoting by $x_\star$ an optimal point of F), typical examples of initial conditions include:

- $\|x_0 - x_\star\|_2^2 \leqslant 1$,
- $\|\nabla F(x_0)\|_2^2 \leqslant 1$ (when F is differentiable, otherwise similar criterion involving some subgradient of F might be used),
- $F(x_0) - F(x_\star) \leqslant 1$,
- any linear combination of the above (see, e.g., examples in the *potential functions* folder of the package).

Table 4.3: Base operator classes within PEPIT, detailed in the documentation. Some classes are overlapping and are present only to promote a better readability of the code. Note that, for some classes, the associated constraints might not be tight, meaning that the methodology might only be able to generate upper-bound on the worst-case behaviors.

Operator class name	Tightness
Monotone (maximally)	✔
Strongly monotone (maximally)	✔
Cocoercive	✔
Lipschitz continuous	✔
Negative comonotone	✔
Cocoercive and strongly monotone	✖
Lipschitz continuous and strongly monotone	✖
Non-expansive	✔
Linear	✔
Symmetric Linear	✔
Skew-symmetric Linear	✔

Similarly, typical examples of performance measures (see Table 4.4 for examples) include: $\|x_n - x_\star\|_2^2$, $\|\nabla F(x_n)\|_2^2$ (when F is differentiable), $F(x_n) - F(x_\star)$, or linear combinations and minimum values of the above,
e.g. $\min_{0 \leqslant i \leqslant n} \|\nabla F(x_i)\|_2^2$ (when F is differentiable).

Examples. PEPIT contains about 75 examples that can readily be used, instantiating the different sets of black-box oracles, problem classes, and initial condition/performance measures. Those examples can be found in the folder PEPIT/EXAMPLES/.

Contributing. PEPIT is designed to allow users to easily contribute to add features to the package. Classes of functions (or operators) as well as black-box oracles can be implemented by following the contributing guidelines from the documentation. We also welcome any new example for analyzing a method/setting that is not already present in the toolbox.

Table 4.4: Examples of common performance measures. This topic is discussed extensively in excellent references that include Nemirovskii (1992, 1994); see also (Taylor et al., 2018b, Tables 1–3) for examples in the context of (proximal) Gradient descent. Considering appropriate performance measures is key for the analyses, and is particularly exploited when looking for appropriate Lyapunov functions, see, e.g., the related Lessard et al. (2016); Taylor and Bach (2019); Taylor et al. (2018a).

Performance measure	Description
Distance	$\|x_n - x_\star\|_2^2$
Gradient norm	$\|\nabla F(x_n)\|_2^2$
Function value	$F(x_n) - F(x_\star)$
Contraction	$\|x_n - y_n\|_2^2$
Lyapunov functions	$a(F(x_n) - F(x_\star)) + \begin{pmatrix} x_n - x_\star \\ \nabla F(x_n) \end{pmatrix}^\top (P \otimes I_d) \begin{pmatrix} x_n - x_\star \\ \nabla F(x_n) \end{pmatrix}$

4.5 A few additional numerical examples

The following section provides a few additional numerical worst-case analyses obtained through PEPit; namely an accelerated gradient method Nesterov (2003), an accelerated Douglas-Rachford splitting Patrinos et al. (2014), and point-SAGA Defazio (2016) (a proximal method for finite-sum minimization).

4.5.1 Analysis of an accelerated gradient method

For this example, we focus again on the problem of minimizing a L-smooth μ-strongly convex function (problem (4.13) with $F \in \mathcal{F}_{\mu,L}$). We consider a classical accelerated gradient method with constant momentum Nesterov (1983, 2003). It can be described as follows for $t \in \{0, ..., n-1\}$ with $y_0 = x_0$:

$$\begin{aligned} x_{t+1} &= y_t - \alpha \nabla F(y_t), \\ y_{t+1} &= x_{t+1} + \beta(x_{t+1} - x_t), \end{aligned} \tag{4.14}$$

with $\kappa = \frac{\mu}{L}$, $\alpha = \frac{1}{L}$ and $\beta = \frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}$. We decide to compute the smallest possible $\tau(n, L, \mu)$ such that the guarantee

$$F(x_n) - F_\star \leqslant \tau(n, L, \mu) \left(F(x_0) - F(x_\star) + \frac{\mu}{2} \|x_0 - x_\star\|_2^2 \right),$$

holds for all $d \in \mathbb{N}$, $F \in \mathcal{F}_{\mu,L}$, $x_0, x_n, x_\star \in \mathbb{R}^d$ where x_n is an output of the accelerated gradient method (4.14) and $x_\star$ is the minimizer of F. In this setting, $\tau(n, L, \mu)$ can be computed as the worst-case value of $F(x_n) - F_\star$ (the performance metric) when $F(x_0) - F_\star + \frac{\mu}{2}\|x_0 - x_\star\|_2^2 \leqslant 1$ (initial condition). As a reference, we compare the output of PEPit to the following worst-case guarantee (d'Aspremont et al., 2021, Corollary 4.15):

$$F(x_n) - F_\star \leqslant \left(1 - \sqrt{\frac{\mu}{L}}\right)^n \left(F(x_0) - F_\star + \frac{\mu}{2} \|x_0 - x_\star\|_2^2 \right). \tag{4.15}$$

A comparison between the output of PEPIT and (4.15) is presented in Figure 4.2a, where we see that (4.15) could be slightly improved to better match the worst-case behavior of the algorithm. The corresponding code can be found in accelerated_gradient_strongly_convex.py from the PEPit/examples/unconstrained_convex_minimization/ directory.

4.5.2 Analysis of an accelerated Douglas-Rachford splitting

In this section, we provide a simple PEPIT example for studying an accelerated Douglas-Rachford splitting method. This method was introduced in Patrinos et al. (2014) where a worst-case analysis is provided for quadratic minimization. We perform a worst-case analysis numerically for a slightly more general setting:

$$F_\star \triangleq \min_x \{F(x) \equiv f_1(x) + f_2(x)\},$$

where f_1 is closed proper and convex, and f_2 is μ-strongly convex and L-smooth. This section focuses on the following accelerated Douglas-Rachford splitting method, described in (Patrinos et al., 2014, Section 4):

$$\begin{aligned} x_t &= \mathrm{prox}_{\alpha f_2}(u_t), \\ y_t &= \mathrm{prox}_{\alpha f_1}(2x_t - u_t), \\ w_{t+1} &= u_t + \theta(y_t - x_t), \\ u_{t+1} &= \begin{cases} w_{t+1} + \frac{t-1}{t+2}(w_{t+1} - w_t) & \text{if } t \geqslant 1, \\ w_{t+1} & \text{otherwise,} \end{cases} \end{aligned}$$

where prox denotes the usual proximal operator, available in PEPIT through the operation PROXIMAL_STEP, as exemplified below. Note that we only show the algorithm description here, the full PEPIT code for this example can be found in the file accelerated_douglas_rachford_splitting.py from the directory PEPit/examples/composite_convex_minimization/.

```
# Compute n steps of
# An accelerated Douglas-Rachford splitting
for t in range(n):
    x[t], _, _ = proximal_step(u[t], func2, alpha)
    y, _, fy = proximal_step(2*x[t] - u[t],
                             func1, alpha)
    w[t+1] = u[t] + theta * (y-x[t])
    if t >= 1:
        u[t+1] = w[t+1] + (t-1)/(t+2) * (w[t+1]-w[t])
    else:
        u[t+1] = w[t+1]
```

When f_2 is a L-smooth μ-strongly convex quadratic function, the following worst-case guarantee is provided by (Patrinos et al., 2014, Theorem 5):

$$F(y_n) - F_\star \leqslant \frac{2\|w_0 - w_\star\|_2^2}{\alpha\theta(n+3)^2}, \tag{4.16}$$

when $\theta = \frac{1-\alpha L}{1+\alpha L}$ and $\alpha < \frac{1}{L}$. A numerical worst-case guarantee for the case $L = 1$, $\mu = 0.01$, $\alpha = 0.9$, $\theta = \frac{1-\alpha L}{1+\alpha L}$ is provided on Figure 4.2b for a few different values of N, where we use (4.16) as a reference for comparison. For each of those values, PEPIT computed a tight (up to numerical precision) worst-case value for which we are not aware of any proven analytical worst-case guarantee beyond the quadratic setting. We see that PEPIT provides an improvement over this guarantee, even when the problem under consideration is not quadratic.

4.5.3 Analysis of point-SAGA

In this section, we use PEPit for studying point-SAGA Defazio (2016), a stochastic algorithm for finite sum minimization:

$$F_\star \triangleq \min_x \left\{ F(x) \equiv \frac{1}{n}\sum_{i=1}^n f_i(x) \right\},$$

where $f_1, \dots, f_n$ are L-smooth and μ-strongly convex functions with a proximal operator available for each of them. At each iteration t, point-SAGA picks $j_t \in \{1, \dots, n\}$ uniformly at random and performs the following updates (a superscript is used for denoting iteration numbers; the subscript is used for referring to the function f_{j_t} chosen uniformly at random):

$$\begin{aligned} z_{j_t}^{(t)} &= x^{(t)} + \gamma \left(g_{j_t}^{(t)} - \frac{1}{n}\sum_i g_i^{(t)} \right), \\ x_{j_t}^{(t+1)} &= \mathrm{prox}_{\gamma f_{j_t}}\left(z_{j_t}^{(t)} \right), \\ g_{j_t}^{(t+1)} &= \frac{1}{\gamma}\left(z_{j_t}^{(t)} - x_{j_t}^{(t+1)} \right), \end{aligned}$$

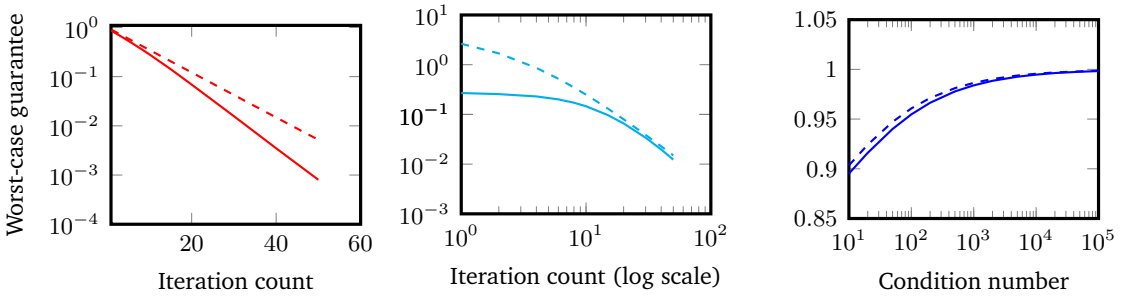
where $\gamma = \frac{\sqrt{(n-1)^2 + 4n\frac{L}{\mu}}}{2Ln} - \frac{\left(1-\frac{1}{n}\right)}{2L}$ is the step-size. In this example, we use a Lyapunov (or potential / energy) function $V(x) = \frac{1}{L\mu}\frac{1}{n}\sum_{i \leqslant n} \|\nabla f_i(x) - \nabla f_i(x_\star)\|_2^2 + \|x - x_\star\|_2^2$, and compute the smallest $\tau(n, L, \mu)$ such that the guarantee

$$\mathbb{E}_{j_t}\left[V\left(x_{j_t}^{(t+1)}\right) \right] \leqslant \tau(n, L, \mu) V(x^{(t)}),$$

holds for all $d \in \mathbb{N}$, $f_i \in \mathcal{F}_{\mu,L}(\mathbb{R}^d)$ (for all $i = 1, \dots, n$), $x^{(t)} \in \mathbb{R}^d$ where $x_{j_t}^{(t+1)}$ is the (random) output generated by point-SAGA, and the expectation is taken over the randomness of j_t. The following simple worst-case guarantee is provided in (Defazio, 2016, Theorem 5) and is used as a reference:

$$\mathbb{E}_{j_t}[V(x_{j_t}^{(t+1)})] \leqslant \frac{1}{1+\mu\gamma} V(x^{(t)}). \tag{4.17}$$

We compare (4.17) to PEPit's tight (up to numerical precision) output in Figure 4.2c. We see that the worst-case guarantee (4.17) can be slightly improved, although pretty accurate, particularly for large values of the condition number. The corresponding PEPit code of this example can be found in the file point_saga.py from the directory PEPit/examples/stochastic_and_randomized_convex_minimization/.



(a) Accelerated gradient method: strong convexity parameter fixed to $\mu = 0.1$. Worst-case guarantee on $F(x_n) - F_\star$ as a function of n.

(b) Accelerated Douglas-Rachford splitting with parameter $\alpha = 0.9$ and $\mu = 0.1$. Worst-case guarantee on $F(y_n) - F_\star$ as a function of n.

(c) Point-SAGA: $n = 5$ functions. Worst-case guarantee on $E_{j_t}[V(x_{j_t}^{(t+1)}]$ as a function of the condition number $\kappa = \frac{L}{\mu}$.

Figure 4.2: Comparisons between (numerical) worst-case bounds from PEPIT (**plain lines**) VS. reference established worst-case guarantees (**dashed lines**) for three different optimization methods. For simplicity, we fixed smoothness constants to $L = 1$.

4.6 Conclusion

The PEPIT package, briefly described in this paper, aims at providing simplified access to worst-case analyses of first-order optimization methods in PYTHON. For doing that, it implements the performance estimation approach while allowing to avoid the possibly heavy semidefinite programming modeling steps. The first version of the package already contains about 75 examples of first-order methods that can be analyzed through this framework. Those examples allow either reproducing or tightening, numerically, known worst-case analyses or provide new ones depending on the particular method and problem class at hand.

Overall, we believe that this package allows quick (in)validations of proofs (a step towards reproducible theory) which should help both the development and the review process in optimization. We also argue that this is a nice pedagogical tool for learning algorithms together with their worst-case properties just by playing with them. Possible extensions under consideration for future versions include an option for searching for Lyapunov (or potential/energy) functions Taylor and Bach (2019); Taylor et al. (2018a); Upadhyaya et al. (2023), for disproving convergence Goujaud et al. (2023a), as well as a numerical proof assistant, and to incorporate recent extensions of PEP/IQCs to distributed and decentralized optimization Sundararajan et al. (2020); Colla and Hendrickx (2021).

5

On Fundamental Proof Structures in First-Order Optimization

First-order optimization methods have attracted a lot of attention due to their practical success in many applications, including in machine learning. Obtaining convergence guarantees and worst-case performance certificates for first-order methods has become crucial for understanding ingredients underlying efficient methods and for developing new ones. However, obtaining, verifying, and proving such guarantees is often a tedious task. Therefore, a few approaches were proposed for rendering this task more systematic, and even partially automated. In addition to helping researchers find convergence proofs, these tools provide insights on the general structures of such proofs. We aim to present those structures, explaining how to build convergence guarantees for first-order optimization methods.

This chapter is based on our tutorial "On Fundamental Proof Structures in First-Order Optimization" (co-authored with A. Dieuleveut, and A. Taylor), presented at CDC 2023.

Contents

5.1 Introduction

In recent years, there has been a significant surge in the interest surrounding first-order optimization methods, primarily driven by their remarkable efficiency on a number of applications, notably within the field of machine learning (see e.g., Bottou and Bousquet (2007)). Theoretical foundations for those methods played a crucial role in this success, e.g., by enabling the development of momentum-type methods (see e.g., Polyak (1963); Nesterov (1983)). Formally, we consider the optimization problem

$$x_\star \triangleq \arg\min_{x\in\mathbb{R}^d} f(x) \tag{OPT}$$

where f belongs to a set $\mathcal{F}$ (often referred to as a "class of functions", e.g., the set of convex functions, the set of strongly convex and smooth functions, or the set of quadratic convex functions, etc.). Classical first-order optimization methods for solving this problem include *Gradient descent* (GD) Cauchy (1847), *Nesterov accelerated gradient method* (NAG) Nesterov (1983), and the *heavy-ball method* (HB) Polyak (1963).

In this context, a key question is to obtain a priori performance guarantees for an iterative algorithm $\mathcal{A}$ (i.e., $\mathcal{A}$ is a rule for generating sequences of approximations $(x_t)_{t\leqslant T}$ to the minimizers of a certain function f) when the function f to be minimized belongs to a set $\mathcal{F}$. The most popular framework for such analyses of optimization algorithms is that of *worst-case analyses*, see, e.g., Nesterov (1983); Dvurechensky et al. (2021); Bubeck (2015); d'Aspremont et al. (2021); Chambolle and Pock (2016). Given an algorithm $\mathcal{A}$, the *worst-case analysis* framework consists in finding guarantees that hold for *every* function of the class.

In other words, we aim at evaluating the worst-case accuracy of $\mathcal{A}$ over the functions of the class $\mathcal{F}$ after a given number of iterations T. For doing so, there are many different possible notions of *accuracy* (or performance) which we denote by $P(f,(x_t)_{t\leqslant T})$ and that we aim at minimizing. Letting x_T be the output of an algorithm, common examples of such metrics include the distance of the last iterate to an optimum $\|x_T-x_\star\|$, the function value accuracy of the last iterate $f(x_T)-f(x_\star)$, or its gradient norm $\|\nabla f(x_T)\|$. Usually, x_T can be arbitrarily bad just by choosing x_0 arbitrarily far away from the optimizer $x_\star$. Therefore, we usually need to assume x_0 to be not too bad, such as $x_0\in\mathcal{N}(x_\star)$ where $\mathcal{N}(x_\star)$ can be any fixed set (that we call a "neighborhood" of the optimizer $x_\star$) and depends on $x_\star$. Common examples of such neighborhood are balls around the optimizer $\{x|\|x-x_\star\|\leqslant R\}$ or the set $\{x|f(x)-f_\star\leqslant R\}$.

The smallest upper bound on $P(f,(x_t)_{t\leqslant T})$ that holds for any dimension $d\geqslant 1$, for any function $f\in\mathcal{F}$, for any starting point $x_0\in\mathcal{N}(x_\star)\subset\mathbb{R}^d$, and for any $(x_t)_{t\leqslant T}$ generated by $\mathcal{A}$ applied on f from x_0, is the optimal value to the problem of computing the worst-case:

$$\left|\begin{array}{ll} \underset{\substack{f\in\mathcal{F},d\geqslant 1\\ (x_t)_{t\leqslant T}\in(\mathbb{R}^d)^{T+1}}}{\text{maximize}} & P(f,(x_t)_{t\leqslant T}) \\ \text{subject to} & \left\{\begin{array}{c} x_0\in\mathcal{N}(x_\star) \\ (x_t)_{t\leqslant T}=\mathcal{A}(f,T,x_0) \end{array}\right. \end{array}\right. \tag{$\mathcal{P}$}$$

In the black-box model, iterative algorithms gather information about f through so-called *oracles*, which we denote by $\mathcal{O}^{(f)}$. Classical oracles used in first-order optimization are gradient evaluations $\mathcal{O}^{(f)}(x)=\nabla f(x)$ and approximate gradients $\mathcal{O}^{(f)}(x)\approx\nabla f(x)$ (e.g., stochastic gradients), but also proximal operators (see, e.g. Combettes and Pesquet (2011)), etc. At step $t\in[\![1,T]\!]$, $\mathcal{A}$ collects oracles on the previous iterates $(\mathcal{O}^{(f)}(x_s))_{s\leqslant t-1}$ and outputs x_t based on those information through the update function A_t as $x_t=A_t((x_s,\mathcal{O}^{(f)}(x_s))_{s<t})$.

Notation. For readability purposes, all notation used throughout this paper are summarized as follows.

Outline. In Section 5.2, we discuss two ways of characterizing classes of functions and detail the main cases for which we can solve ($\mathcal{P}$). In Section 5.3, we discuss an alternative way of describing

Notation	Corresponding object
$\mathcal{F}$	Class of functions (generic form)
f	Objective function
$x_\star$	Optimal point
x_0	Initial iterate
$\mathcal{O}^{(f)}$	Generic oracle applied on f
$\mathcal{A}$	Algorithm (generic form)
$(x_t)_{t\leqslant T}$	Sequence of iterates generated by $\mathcal{A}$, i.e. $(x_t)_{t\leqslant T} = \mathcal{A}(f, T, x_0)$
$(A_t)_{1\leqslant t\leqslant T}$	Update function of the algorithm $\mathcal{A}$, i.e. $\forall t, x_t = A_t((x_s, \mathcal{O}^f(x_s))_{s<t})$
T	Total number of iterations
t	Current iteration index
$\mathcal{F}_{\mu,L}$	Class of L-smooth and μ-strongly convex functions ($0 \leqslant \mu \leqslant L$)
$\mathcal{Q}_{\mu,L}$	Class of L-smooth and μ-strongly convex quadratic functions ($0 \leqslant \mu \leqslant L$)
$(V_t)_t$	Lyapunov sequence
F, G	Linearization variables (after SDP lifting)
$P(f, (x_t)_{t\leqslant T})$	Performance metric

the algorithm $\mathcal{A}$ simplifying the resolution of ($\mathcal{P}$). Section 5.4 outlines a systematic approach for acquiring proofs of worst-case performance certificates and delves into their underlying structures. We further elaborate on how this structure can be exploited for extending the applicability range of the worst-case guarantees. Among others, we show how the properties of these proofs allow building algorithms. Finally, Section 5.6 provides a natural approach for discovering Lyapunov sequences.

5.2 From explicit to implicit classes of functions

This section describes two ways of specifying a class of functions as part of the worst-case analysis of a given algorithm. We describe two different methods to approach and solve ($\mathcal{P}$) depending on the ways $\mathcal{F}$ is specified. More specifically, we focus on two specific classes of functions to illustrate our explanations, namely L-smooth μ-strongly convex quadratic functions (notation $\mathcal{Q}_{\mu,L}$) and L-smooth μ-strongly convex functions (notation $\mathcal{F}_{\mu,L}$).

5.2.1 Convex quadratic optimization

First-order optimization methods were extensively studied in the context of minimizing quadratic convex functions. Such functions can be described **explicitly** as

$$f(x) \triangleq \frac{1}{2}(x - x_\star)^T H(x - x_\star) + f_\star, \tag{5.1}$$

where H is the symmetric positive semi-definite Hessian of f, $x_\star$ its optimizer and $f_\star$ its minimal value. This expression allows to explicitly compute the gradient $\nabla f(x) = H(x - x_\star)$, and first-order optimization methods can be expressed through polynomials due to the following property (e.g., (Goujaud et al., 2022b, Prop.4.1)).

Proposition 5.2.1. *Let $f \in \mathcal{Q}_{0,\infty}$ and $x_0 \in \mathbb{R}^d$. It holds that*

$$x_{t+1} \in x_0 + \operatorname{span}\{\nabla f(x_0), \dots, \nabla f(x_t)\}, \tag{5.2}$$

if and only if there exists a sequence of polynomials $(P_t)_{t\in\mathbb{N}}$, each of degree at most 1 more than the highest degree of all previous polynomials and P_0 of degree 0 (hence the degree of P_t is at most

t), such that

$$\forall\, t \quad x_t - x_\star = P_t(H)(x_0 - x_\star), \quad P_t(0) = 1\,. \tag{5.3}$$

In this context, ($\mathcal{P}$) can be solved by solving a polynomial problem of the form $\max_H \|P_t(H)\|$ where H is a symmetric matrix verifying some conditions (e.g. $\mu I \preccurlyeq H \preccurlyeq LI$ when $f \in \mathcal{Q}_{\mu,L}$). This link between first-order algorithms and polynomials has been used by Golub and Varga (1961) for discovering the Chebyshev method and by Polyak (1963) for the "heavy-ball" method, still used nowadays far beyond quadratic optimization (e.g. in stochastic optimization of neural networks Sutskever et al. (2013)). This property has also been exploited more recently for obtaining new algorithms with provable guarantees on quadratic functions (see e.g., Fischer (2011); Scieur (2018); d'Aspremont et al. (2021); Pedregosa and Scieur (2020); Scieur and Pedregosa (2020); Berthier et al. (2020); Goujaud et al. (2022b,d); Cunha et al. (2022)).

5.2.2 Infinite-dimensional spaces of functions

As opposed to previous sections, many classes of functions are described *implicitly* as regions of infinite-dimensional spaces of functions. In other words, such functions are defined by sets of inequalities. This section deals with the analyses of such classes. This is due to the fact the set of all functions of the class are not described by a finite number of parameters, but rather by constraints (inequalities). Studying ($\mathcal{P}$) for classes that are defined **implicitly** through sets of constraints appears to be much less natural. In this situation, ($\mathcal{P}$) is often referred to as a *performance estimation problem (PEP)* Drori and Teboulle (2014); Taylor et al. (2017c,a). This tool primarily relies on two crucial components: interpolation conditions and SDP lifting.

Interpolation conditions. We remark that the description of the algorithm and the objective of ($\mathcal{P}$) both only depend on the oracle values of f on the iterates $(x_t)_{t\leqslant T}$. We introduce the variables $(\mathcal{O}_t)_{t\leqslant T}$. The constraint $f \in \mathcal{F}$ must be replaced by the constraint that there exists at least one element $f \in \mathcal{F}$ such that $(\mathcal{O}^{(f)}(x_t))_{t\leqslant T} = (\mathcal{O}_t)_{t\leqslant T}$ ($\mathcal{O}_t$ is a reachable value for $\mathcal{O}^{(f)}(x_t)$, when $f \in \mathcal{F}$). As an example, f_t and g_t are potential values of respectively $f(x_t)$ and $\nabla f(x_t)$. Formally, we define the equivalence relation $\sim_{(\mathcal{P})}$ as $f_1 \sim_{(\mathcal{P})} f_2$ if and only if $\forall t \in [\![0, T]\!] \cup \{\star\}, \mathcal{O}^{(f_1)}(x_t) = \mathcal{O}^{(f_2)}(x_t)$. Since the only information $\mathcal{A}$ gathers on f is the oracle outputs at the iterates x_t, two functions coming from the same equivalence class both produce feasible points of ($\mathcal{P}$) with the same objective value. In other words, those two functions are undistinguishable using only the information available to $\mathcal{A}$. We can therefore rewrite ($\mathcal{P}$) in terms of $(\mathcal{O}_t)_{t\leqslant T} \in \mathcal{F}/\sim_{(\mathcal{P})}$ instead of $f \in \mathcal{F}$, so that the set of optimization variables now lives in finite dimension. This constraint is referred to as *interpolation conditions*.

Example 5.2.2 (First-order algorithm on $\mathcal{F}_{\mu,L}$)**.** *Let $L \geqslant \mu > 0$ two positive real numbers. A function f is L-smooth and μ-strongly convex when f is continuously differentiable and verifies the two inequalities:*

$$f(x) \leqslant f(y) + \langle \nabla f(y), x - y\rangle + \frac{L}{2}\|x - y\|^2, \tag{5.4}$$

$$f(x) \geqslant f(y) + \langle \nabla f(y), x - y\rangle + \frac{\mu}{2}\|x - y\|^2, \tag{5.5}$$

for all x, y and where ∇f denotes the gradient of f.

Studying a first-order algorithm (i.e. an algorithm based on the oracle $\mathcal{O}^{(f)} \triangleq (\nabla f, f)$) on the class $\mathcal{F}_{\mu,L}$ appears to be challenging at first sight due to the infinite number of parameters needed for describing $\mathcal{F}_{\mu,L}$. However, (Taylor et al., 2017c, Theorem 4) provides interpolation conditions for the class $\mathcal{F}_{\mu,L}$ of L-smooth μ-strongly convex functions and enables an exact study

of the worst-case of several algorithms on this class of functions:

$$\forall i,j,\ f_i \geqslant f_j + \langle g_j, x_i - x_j\rangle + \tfrac{1}{2L}\|g_i - g_j\|^2 + \tfrac{\mu}{2(1-\mu/L)}\|x_i - \tfrac{1}{L}g_i - x_j + \tfrac{1}{L}g_j\|^2. \tag{IC}$$

Indeed, in this case, ($\mathcal{P}$) can be written in finite dimension as

$$\left|\begin{array}{ll} \underset{\substack{d\geqslant 1,(x_t)_{t\leqslant T}\in(\mathbb{R}^d)^{T+1},x_\star\in\mathbb{R}^d,\\ (g_t,f_t)_{t\leqslant T}\in(\mathbb{R}^d\times\mathbb{R})^{T+1}}}{\text{maximize}} & P((x_t,g_t,f_t)_{t\leqslant T}) \\ \text{s.t.} & \begin{cases} x_0\in\mathcal{N}(x_\star) \\ \forall t\leqslant T,\ x_t = A_t((x_s,\mathcal{O}^f(x_s))_{s<t}) \\ \forall i,j,\ f_i \geqslant f_j + \langle g_j, x_i - x_j\rangle + \frac{1}{2L}\|g_i - g_j\|^2 \\ \qquad + \frac{\mu}{2(1-\mu/L)}\|x_i - \frac{1}{L}g_i - x_j + \frac{1}{L}g_j\|^2. \end{cases} \end{array}\right.$$

SDP lifting. In many cases (see, e.g, Example 5.2.2, and (Taylor et al., 2017a, Theorem 3.5)), interpolation conditions are written in terms of quadratic and bilinear expressions of x_t and g_t and linear expressions of f_t. Because of the quadratic dependency in x_t and g_t, this problem is generally non-convex. *SDP lifting* can convexify this problem if all other parts of this problem also contain only quadratic expressions of x_t and g_t. For example, classical choices for $P(f,(x_t)_{t\leqslant T})$ are $\|x_T - x_\star\|^2$, $f(x_T) - f(x_\star)$, or $\|\nabla f(x_T)\|^2$. Similarly, a classical choice for $x_0\in\mathcal{N}(x_\star)$ is $\|x_0 - x_\star\|^2\leqslant R^2$ for some radius $R>0$. Finally, the updates $(A_t)_t$ of the algorithm $\mathcal{A}$ are often of the form

$$x_t = A_t((x_s,\nabla f(x_s),f(x_s))_{s\leqslant t-1}) = x_0 - \sum_{s=0}^{t-1}\gamma_s^{(t)}\nabla f(x_s) \tag{5.6}$$

for some sequence of scalars $(\gamma_s^{(t)})_{s\in[\![0,t-1]\!]}$. Substituting x_t for $t\geqslant 1$ in the problem by their corresponding expressions given by (5.6) preserves the above observation: the dependency of ($\mathcal{P}$) in $(x_t,g_t)_{t\leqslant T}$ is exclusively quadratic. Actually, in this specific case, all occurrences of $(x_t)_{t\geqslant 1}$ have been replaced by linear combinations of x_0 and $(g_t)_{t\leqslant T}$. *SDP lifting* consists in introducing the Gram matrix G of $(x_0 - x_\star,(g_t)_{t\leqslant T})$. This way, all quadratic expressions of $(x_t,g_t)_{t\leqslant T}$ are linear combinations of the entries of G. We also introduce the vector F storing the values $(f_t - f_\star)_{t\leqslant T}$.

Finally ($\mathcal{P}$) is rewritten with linear objective and constraints only as well as an SDP constraint $G\succcurlyeq 0$.

$$\left|\begin{array}{ll} \underset{F,G\succcurlyeq 0}{\text{maximize}} & \langle F, v_P\rangle + \langle G, M_P\rangle \\ \text{subject to} & \begin{cases} \langle F, v_I\rangle + \langle G, M_I\rangle \leqslant R^2 \\ \forall k,\ \left\langle F, v_{\mathcal{F}}^{(k)}\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}\right\rangle \leqslant 0 \end{cases} \end{array}\right.$$

Vectors $(v_P, v_I, (v_{\mathcal{F}}^{(k)})_k)$ and matrices $(M_P, M_I, (M_{\mathcal{F}}^{(k)})_k)$ are constants depending on the algorithm $\mathcal{A}$, the class $\mathcal{F}$, and the performance metric P under consideration. More specifically, indices P, I and $\mathcal{F}$ respectively correspond to the performance metric, the initialization constraint and the class interpolation conditions. The algorithm is directly encoded in the fact that G does not contain inner product with $(x_t)_{t\geqslant 1}$. As an example, to express $\|x_T - x_\star\|^2$ in terms of G, one needs to actually choose M with $\langle G, M\rangle = \|x_0 - x_\star - \sum_{s=0}^{T-1}\gamma_s^{(T)}\nabla f(x_s)\|^2$.

Key conditions. The above procedure generally works under the following conditions:

- $\mathcal{A}$ is a first-order algorithm whose updates $(A_t)_t$ can be expressed linearly in terms of observed gradients;
- The interpolation constraints of the class of functions $\mathcal{F}$ are known and expressible linearly in F and G;
- The performance metric as well as the initial condition are also expressible linearly in terms of F and G.

Many pairs of function class and algorithm meet the right conditions and have been studied using the PEP framework. Tools in Matlab Taylor et al. (2017b) and Python Goujaud et al. (2022a) have been implemented to automate this task and provide worst-case guarantees. Many examples of usages are listed in the corresponding documentations.

5.3 From explicit to implicit algorithms

So far, we only considered explicit algorithms of the form (5.6). Note that, just as for classes of functions, algorithms can be expressed implicitly via sets of (in)equalities. This is the case for line-search based algorithms. Indeed, the step-size associated with line-search is not uniform over the problem class, therefore algorithms containing line-search update cannot be written as (5.6), and therefore do not meet the key conditions mentioned in the previous section. A relaxation of the Gradient descent with exact line-search has been proposed in De Klerk et al. (2017). Since this algorithm cannot be written as (5.6) with pre-determined $\gamma_s^{(t)}$, we cannot specify $(x_t)_{t\geqslant 1}$ in terms of x_0 and $(g_t)_{t\leqslant T}$. Therefore, all vectors $(x_t, g_t)_{t\leqslant T}$ must be considered as linearly independent. For this problem, G is the Gram matrix of all $(x_t, g_t)_{t\leqslant T}$.

Therefore, the algorithm is not totally encoded in $v_P, M_P, v_I, M_I, v_{\mathcal{F}}$ and $M_{\mathcal{F}}$ anymore and must be specified by new constraints. In particular, the updates of Gradient descent with line-search verify that

$$\langle g_{t+1}, g_t\rangle = 0 \tag{5.7}$$
$$\langle g_{t+1}, x_{t+1} - x_t\rangle = 0 \tag{5.8}$$

As for all the other elements of $(\mathcal{P})$, those constraints only involve quadratic terms of $(x_t, g_t)_{t\leqslant T}$ and can therefore be expressed linearly in terms of G, parametrized by the vectors $(v_{\mathcal{A}}^{(l)})_l$ and matrices $(M_{\mathcal{A}}^{(l)})_l$. This time, $(\mathcal{P})$ writes

$$\left|\begin{array}{ll} \underset{F,G\succcurlyeq 0}{\text{maximize}} & \langle F, v_P\rangle + \langle G, M_P\rangle \\ \text{subject to} & \left\{\begin{array}{rll} & \langle F, v_I\rangle + \langle G, M_I\rangle & \leqslant R^2 \\ \forall k, & \left\langle F, v_{\mathcal{F}}^{(k)}\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}\right\rangle & \leqslant 0 \\ \forall l, & \left\langle F, v_{\mathcal{A}}^{(l)}\right\rangle + \left\langle G, M_{\mathcal{A}}^{(l)}\right\rangle & \leqslant 0 \end{array}\right. \end{array}\right. \tag{PEP-primal}$$

5.4 Proof structures in first-order optimization

There is an extensive literature on first-order optimization, offering a broad range of possibly advanced worst-case guarantees and their associated proofs. In the previous sections, we saw conditions under which the problem of computing worst-case guarantees was tractable. In this section, we detail how to obtain proofs from PEPs and what we can conclude on the general structure of proofs in first-order optimization.

5.4.1 Obtaining proofs with PEPs

Thanks to interpolation conditions and SDP lifting, $(\mathcal{P})$ rewrites as a convex optimization problem. We consider the dual of the problem. Let's then introduce the Lagrangian multipliers τ, $(\lambda_{\mathcal{F}}^{(k)})_k$, $(\lambda_{\mathcal{A}}^{(l)})_l$ associated to the constraints of (PEP-primal).

$$\left|\begin{array}{ll} \underset{F,G\succcurlyeq 0}{\text{maximize}} & \langle F, v_P\rangle + \langle G, M_P\rangle \\ \text{subject to} & \left\{\begin{array}{rll} & \langle F, v_I\rangle + \langle G, M_I\rangle & \leqslant R^2 \quad : \tau \\ \forall k, & \left\langle F, v_{\mathcal{F}}^{(k)}\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}\right\rangle & \leqslant 0 \quad : \lambda_{\mathcal{F}}^{(k)} \\ \forall l, & \left\langle F, v_{\mathcal{A}}^{(l)}\right\rangle + \left\langle G, M_{\mathcal{A}}^{(l)}\right\rangle & \leqslant 0 \quad : \lambda_{\mathcal{A}}^{(l)} \end{array}\right. \end{array}\right.$$

The Lagrangian then writes

$$\begin{aligned}\mathcal{L} \triangleq &\langle F, v_P\rangle + \langle G, M_P\rangle - \tau\left[\langle F, v_I\rangle + \langle G, M_I\rangle - R^2\right]\\ &- \sum_k \lambda_{\mathcal{F}}^{(k)}\left[\left\langle F, v_{\mathcal{F}}^{(k)}\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}\right\rangle\right]\\ &- \sum_l \lambda_{\mathcal{A}}^{(l)}\left[\left\langle F, v_{\mathcal{A}}^{(l)}\right\rangle + \left\langle G, M_{\mathcal{A}}^{(l)}\right\rangle\right]\\ =&\tau R^2 + \left\langle F, v_P - \tau v_I - \sum_k \lambda_{\mathcal{F}}^{(k)} v_{\mathcal{F}}^{(k)} - \sum_l \lambda_{\mathcal{A}}^{(l)} v_{\mathcal{A}}^{(l)}\right\rangle\\ &+ \left\langle G, M_P - \tau M_I - \sum_k \lambda_{\mathcal{F}}^{(k)} M_{\mathcal{F}}^{(k)} - \sum_l \lambda_{\mathcal{A}}^{(l)} M_{\mathcal{A}}^{(l)}\right\rangle\end{aligned}$$

The dual is obtained by maximizing over the primal variables:

$$\left|\begin{array}{l}\underset{\tau,\lambda_{\mathcal{F}}^{(k)},\lambda_{\mathcal{A}}^{(l)}\geqslant 0}{\text{minimize}}\ \tau R^2\\ \text{s.t.}\begin{cases} v_P - \tau v_I - \sum_k \lambda_{\mathcal{F}}^{(k)} v_{\mathcal{F}}^{(k)} - \sum_l \lambda_{\mathcal{A}}^{(l)} v_{\mathcal{A}}^{(l)} = 0\\ M_P - \tau M_I - \sum_k \lambda_{\mathcal{F}}^{(k)} M_{\mathcal{F}}^{(k)} - \sum_l \lambda_{\mathcal{A}}^{(l)} M_{\mathcal{A}}^{(l)} \preccurlyeq 0\end{cases}\end{array}\right. \tag{PEP-dual}$$

For any feasible primal F, G and feasible dual $\tau, (\lambda_{\mathcal{F}}^{(k)})_k, (\lambda_{\mathcal{A}}^{(l)})_l$, we know the objective of the dual is larger than the Lagrangian value, that is:

$$\begin{aligned}&\underbrace{\langle F, v_P\rangle + \langle G, M_P\rangle}_{\text{Performance metric}} - \tau\underbrace{\left[\langle F, v_I\rangle + \langle G, M_I\rangle\right]}_{\text{Initialization}}\\ &\leqslant \sum_k \lambda_{\mathcal{F}}^{(k)}\underbrace{\left[\left\langle F, v_{\mathcal{F}}^{(k)}\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}\right\rangle\right]}_{\text{Class constraint}}\\ &\quad + \sum_l \lambda_{\mathcal{A}}^{(l)}\underbrace{\left[\left\langle F, v_{\mathcal{A}}^{(l)}\right\rangle + \left\langle G, M_{\mathcal{A}}^{(l)}\right\rangle\right]}_{\text{Algorithm constraint}}\\ &\leqslant 0.\end{aligned} \tag{Generic proof}$$

In words, the proof of a worst-case guarantee is obtained by linearly combining all available constraints, with coefficients that are the dual variables of the PEP. Indeed, the difference between the performance metric and τ times the initialization measure of proximity to the optimizer is decomposed as the sum of three terms. The two first ones respectively correspond to the values that are enforced to be negative by the class of functions and the algorithm. The third one is called the residual and is the opposite of a sum of squares of iterates and gradients. An example of full derivation of such a proof is provided in Section 5.5.

Remark 5.4.1 (No duality gap)**.** *There generally exists a feasible point G, F with $G \succ 0$, i.e. verifying the Slater's condition (see Slater (1950)), therefore guaranteeing strong duality of the convex reformulation of ($\mathcal{P}$). To ensure this, one needs to carefully remove iterates x_t from the basis of G when x_t is completely identified from other vectors. For instance, leaving x_1 in the basis of G with the constraint $\|x_1 - (x_0 - \gamma g_0)\|^2 = 0$ instead of replacing x_1 by $x_0 - \gamma g_0$ everywhere, creates an empty interior and can break strong duality. Each time there is no feasible G with $G \succ 0$, we conclude that there is a linear relationship between elements of the basis G is the Gram matrix of. Therefore, maximally reducing the dimension of G ensures strong duality.*

5.4.2 Understanding proofs with PEPs

Obtaining dual feasible points provides valuable insights into essential aspects pertaining to both the class of functions under consideration and the algorithm employed to achieve the associated worst-case guarantee.

Extension to broader sets of algorithms. Drori and Taylor (2020) exploit these insights to *design worst-case optimal algorithms*. The authors' key observation is that (Generic proof) does not rely on all constraints to hold, but rather only on a linear combination of them. Therefore, if instead of assuming that, $\forall l, \left\langle F, v_{\mathcal{A}}^{(l)} \right\rangle + \left\langle G, M_{\mathcal{A}}^{(l)} \right\rangle \leqslant 0$, we can simply assume that $\sum_l \lambda_{\mathcal{A}}^{(l)} \left[\left\langle F, v_{\mathcal{A}}^{(l)} \right\rangle + \left\langle G, M_{\mathcal{A}}^{(l)} \right\rangle \right] \leqslant 0$, therefore relaxing a lot of assumptions about the algorithm and then generalizing the proof to all the algorithms verifying the remaining assumption. This was applied to the impractical algorithm (GFOM) described as follow:

$$\forall t, x_{t+1} = \underset{x \in x_0 + \operatorname{span}\{\nabla f(x_0), \cdots, \nabla f(x_t)\}}{\arg\min} f(x), \tag{GFOM}$$

greedily minimizing the objective value in the affine space of all the observed directions. For some classes of functions, this algorithm is worst-case optimal. This is the case, for instance, for the class of quadratic convex functions on which (GFOM) is equivalent to the so-called *conjugate gradient method*. This is also the case for the class of L-smooth convex functions, allowing to find a broad range of worst-case optimal algorithms on this class, including the so-called *optimized gradient method (OGM)* Drori (2017); Drori and Taylor (2020). Generating such worst-case optimal algorithms works as follow:

1. We note that (GFOM) verifies the following orthogonality constraints:

$$\forall t, \begin{cases} \forall s < t, \langle g_t, g_s \rangle = 0, \\ \forall s \leqslant t, \langle g_t, x_s - x_0 \rangle = 0. \end{cases} \tag{5.9}$$

   Note that following those constraints does not necessarily imply that (PEP-primal)'s primal variables optimal values describe (GFOM). Nevertheless, a sufficient condition on the class $\mathcal{F}$ under consideration for that to happen is that $\mathcal{F}$ is contraction-preserving (see (Drori and Taylor, 2020, Definition 3)), which happens to be the case for $\mathcal{F}_{\mu,L}$ for example.

2. We call the corresponding dual variables $(\beta_{t,s})_{s<t}$ and $(\gamma_{t,s})_{s\leqslant t}$ and collect their optimal values $(\beta^\star_{t,s})_{s<t}$ and $(\gamma^\star_{t,s})_{s\leqslant t}$: it happens that those values can be obtained in closed-form.

3. We group all the constraints as in (5.9), and conclude that the worst-case guarantee of (GFOM), as well as the corresponding proof, would hold if

$$\forall t, \left\langle g_t, \sum_{s=0}^{t-1} \beta_{t,s} g_s + \sum_{s=0}^{t} \gamma_{t,s}(x_s - x_0) \right\rangle \leqslant 0. \tag{5.10}$$

4. When $\gamma_{t,t} \neq 0$, we conclude that, in particular, the algorithm described by the iteration

$$\forall t, x_t = x_0 - \sum_{s=0}^{t-1} \frac{\gamma_{t,s}}{\gamma_{t,t}}(x_s - x_0) + \frac{\beta_{t,s}}{\gamma_{t,t}} g_s \tag{5.11}$$

   annihilates the vector in the right-hand position of the inner product. Therefore, the worst-case guarantee of (GFOM) also applies to $\mathcal{A}$, using the exact same proof.

This method has more recently been used in (Goujaud et al., 2022c, Th.2.4-Cor.2.5) to derive the worst-case optimal algorithm

$$x_t = \frac{t}{t+1} x_{t-1} + \frac{1}{t+1} x_0 - \frac{1}{t+1} \sum_{s=0}^{t-1} \frac{1}{L} g_s \tag{HB}$$

under the class of convex and L-quadratically upper bounded (L-QG$^+$) functions.

Extension to broader classes of functions. Interestingly, (HB) was studied several years ago in Ghadimi et al. (2015) on the class $\mathcal{F}_{0,L}$ of L-smooth convex functions, itself included in the class of L-QG$^+$ convex functions. On the other hand, the obtained guarantee was not better on $\mathcal{F}_{0,L}$ than the one obtained on the class of L-QG$^+$ convex functions. This shows that the guarantee

obtained on $\mathcal{F}_{0,L}$ can be obtained using only the interpolation constraints of the class of L-QG$^+$ convex functions, which is a subset of the set of interpolation constraints of $\mathcal{F}_{0,L}$. In general, for a given class and a given algorithm, when $\lambda_{\mathcal{F}}^{(k)} = 0$ in (Generic proof), we conclude that the corresponding constraint has not been used. This allows to discard all the useless constraints and the result naturally holds on a larger class of functions.

Fewer class constraints allows new algorithms. Most of the time, we study a family of classes of functions, parametrized by some value L. A classical example of this is the class of L-smooth convex functions $\mathcal{F}_{0,L}$. The underlying *interpolation constraints* $\left\langle F, v_{\mathcal{F}}^{(k)}(L)\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}(L)\right\rangle \leqslant 0$ then depend on L. We generally derive and study an algorithm on $\mathcal{F}_{0,L}$, and obtain a guarantee that holds for any L such that $\langle F, v_P(L)\rangle + \langle G, M_P(L)\rangle - \tau(L)\left[\langle F, v_I(L)\rangle + \langle G, M_I(L)\rangle\right] \leqslant 0$. The underlying algorithm can (and usually does) therefore depend on this value that is sometimes hard to access in practice. Using line-search steps is a way to get rid of the dependence on L (there exists for instance line-search version of *OGM* and (HB) that do not involve L), but an exact line-search step is often not available neither. On the other hand, *backtracking line-search* have been proposed Armijo (1966) to replace the class parameter L by any surrogate value $\hat{L}$ that validates all the inequalities that are used. Indeed, we know that for any L,

$$\begin{aligned}
&\underbrace{\langle F, v_P(L)\rangle + \langle G, M_P(L)\rangle}_{\text{Performance metric}} - \tau(L)\underbrace{\left[\langle F, v_I(L)\rangle + \langle G, M_I(L)\rangle\right]}_{\text{Initialization}} \\
&\leqslant \sum_j \lambda^{(j)} \underbrace{\left[\left\langle F, v^{(j)}(L)\right\rangle + \left\langle G, M^{(j)}(L)\right\rangle\right]}_{\text{Constraint}} \\
&\leqslant 0
\end{aligned} \tag{5.12}$$

Therefore, even if we do not have access to L, being able to find some $\hat{L}$ in an online manner such that all the surrogate constraints $\left\langle F, v_{\mathcal{F}}^{(k)}(\hat{L})\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}(\hat{L})\right\rangle \leqslant 0$ hold, allows tuning the algorithm online with this $\hat{L}$ and obtain the guarantee

$$\left\langle F, v_P(\hat{L})\right\rangle + \left\langle G, M_P(\hat{L})\right\rangle - \tau(\hat{L})\left[\left\langle F, v_I(\hat{L})\right\rangle + \left\langle G, M_I(\hat{L})\right\rangle\right] \leqslant 0.$$

We would like to apply bisection search to find such $\hat{L}$, and all we need for that is being able to verify the constraints $\left\langle F, v_{\mathcal{F}}^{(k)}(\hat{L})\right\rangle + \left\langle G, M_{\mathcal{F}}^{(k)}(\hat{L})\right\rangle \leqslant 0$ online. Note however that some constraints may involve the optimizer $x_\star$ or the minimal value $f_\star$ and are then not verifiable. The authors of (d'Aspremont et al., 2021, Remark 4.9) and Park and Ryu (2021) discuss this issue. They note that we only need to verify constraint that actually involve L and that the ones that are problematic are the ones that involve both L and an unknown value. They conclude that, if the dual values associated with these problematic constraints are set to 0, they are not used, and then we can proceed to *backtracking line-search*. They also enforce it by removing those inequalities (or lowering them) and searching for methods that holds on this larger class of functions (verifying less inequalities) in order to be able to apply *backtracking line-search* to get rid of the requirement of knowledge of the parameter class.

5.5 Example: Gradient descent with exact line-search

For sake of better comprehension of the formal reasoning made in Subsection 5.2.2 and Sections 5.3 and 5.4, we detail in this section the development of a proof of convergence guarantee of the form (Generic proof) on an example: the Gradient descent method with exact line-search, defined as

$$\forall t \in [\![1, T]\!],\ x_t = \underset{x \in x_{t-1} + \operatorname{span}\{\nabla f(x_{t-1})\}}{\arg\min} f(x). \tag{GDLS}$$

More precisely, we chose to consider the function value as performance metric, and therefore seek for a guarantee of the form

$$f(x_1) - f_\star \leqslant \tau(f_0 - f_\star), \tag{5.13}$$

with an appropriate τ. Note this problem has been solved in (De Klerk et al., 2017, Theorem 1.2). Here we detail how to find such a guarantee and its proof in a very systematic way, relying on the framework presented in the present tutorial.

The problem can therefore be summarized as follow:

- The objective function belongs to the class $\mathcal{F}_{\mu,L}$ of L-smooth μ-strongly convex functions, i.e. verifies the interpolation constraints (IC),
- We have access to the oracle $\mathcal{O}^f(x)$ verifying:
  - $\mathcal{O}^f(x) \in x + \operatorname{span}\{\nabla f(x)\}$,
  - $\left\langle \nabla f\left(\mathcal{O}^f(x)\right), \nabla f(x)\right\rangle = 0$,
- The algorithm $\mathcal{A}$ iteratively computes the update $x_t = A_t((x_s, \mathcal{O}^f(x_s))_{s<t}) \triangleq \mathcal{O}^f(x_{t-1})$,
- We study exactly one step of this algorithm. That is, we want a guarantee on x_1 given x_0.
- The performance metric that we use is the function value $f(x_1) - f_\star$.
- The neighborhood $\mathcal{N}(x_\star)$ we assume x_0 belongs to is also define by the function value as $\left\{x \mid f(x) - f_\star \leqslant R^2\right\}$ for some positive R.

In summary, the problem ($\mathcal{P}$) writes

$$\left|\begin{array}{ll} \underset{\substack{f\in\mathcal{F}_{\mu,L}, d\geqslant 1\\ (x_\star,x_0,x_1)\in\left(\mathbb{R}^d\right)^3}}{\text{maximize}} & f(x_1) - f_\star \\ \text{subject to} & \left\{\begin{array}{c} f(x_0) - f_\star \leqslant R^2 \\ (x_t)_{t\leqslant 1} = \text{GDLS}(f, T=1, x_0) \end{array}\right. \end{array}\right. \tag{5.14}$$

GDLS's update is defined through an optimization problem. Implementing it into the PEP framework is not straightforward. Instead, we replace the strict definition of the update by first-order optimality conditions of the line search procedure:

$$\begin{aligned} \langle \nabla f(x_1), \nabla f(x_0)\rangle &= 0, \\ \langle \nabla f(x_1), x_1 - x_0\rangle &= 0. \end{aligned}$$

Note the second one is verified because $x_1 - x_0$ is colinear with g_0 and therefore those 2 conditions seem redundant. However, removing the proper definition of (GDLS) makes $x_1 - x_0$ and g_0 non-necessarily colinear anymore, and the two orthogonality conditions are complementary.

Note furthermore that, replacing the actual definition of (GDLS) by some conditions the latter verifies leads to a guarantee that holds over all the algorithms that verify those conditions. This is therefore possibly a relaxation, but the result still holds. Moreover, in this special case, and because we used the two orthogonality conditions and not just one, replacing the definition of (GDLS) by those conditions is tight. This technical assertion is based on the fact the class $\mathcal{F}_{\mu,L}$ is *contraction-preserving*. This reasoning is detailed in Drori and Taylor (2020).

Expressing the constraints of the algorithm and the class, we obtain

$$\left|\begin{array}{ll} \underset{\substack{d\geqslant 1, (x_\star,x_0,x_1))\in(\mathbb{R}^d)^3,\\ (g_0,g_1)\in(\mathbb{R}^d)^2,\ (f_\star,f_0,f_1)\in\mathbb{R}^3}}{\text{maximize}} & f(x_1) - f_\star \\ \text{s.t.} & \left\{\begin{array}{l} f(x_0) - f_\star \leqslant R^2 \\ \langle \nabla f(x_1), \nabla f(x_0)\rangle = 0, \\ \langle \nabla f(x_1), x_1 - x_0\rangle = 0. \\ \forall i,j,\ f_i \geqslant f_j + \langle g_j, x_i - x_j\rangle + \frac{1}{2L}\|g_i - g_j\|^2 \\ \qquad\qquad + \frac{\mu}{2(1-\mu/L)}\|x_i - \frac{1}{L}g_i - x_j + \frac{1}{L}g_j\|^2. \end{array}\right. \end{array}\right.$$

Using SDP lifting, we can formulate this problem as a semi-definite program of the form (PEP-primal) using the variables

$$F = (f_\star, f_0, f_1)^\top$$
$$G = (x_\star, x_0, g_0, x_1, g_1)^\top (x_\star, x_0, g_0, x_1, g_1).$$

We therefore set the parameters of (Generic proof) to the following values:

$$v_P = (-1, 0, 1)^\top, \qquad M_P = 0_5,$$
$$v_I = (-1, 1, 0)^\top, \qquad M_I = 0_5,$$

$$v_{\mathcal{F}}^{(\star,0)} = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}, \quad M_{\mathcal{F}}^{(\star,0)} = \frac{1}{2(1-\kappa)} \begin{pmatrix} \mu & -\mu & 1 & 0 & 0 \\ -\mu & \mu & -1 & 0 & 0 \\ 1 & -1 & \frac{1}{L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$v_{\mathcal{F}}^{(\star,1)} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \quad M_{\mathcal{F}}^{(\star,1)} = \frac{1}{2(1-\kappa)} \begin{pmatrix} \mu & 0 & 0 & -\mu & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\mu & 0 & 0 & \mu & -1 \\ 1 & 0 & 0 & -1 & \frac{1}{L} \end{pmatrix},$$

$$v_{\mathcal{F}}^{(0,\star)} = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}, \quad M_{\mathcal{F}}^{(0,\star)} = \frac{1}{2(1-\kappa)} \begin{pmatrix} \mu & -\mu & \kappa & 0 & 0 \\ -\mu & \mu & -\kappa & 0 & 0 \\ \kappa & -\kappa & \frac{1}{L} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$v_{\mathcal{F}}^{(0,1)} = \begin{pmatrix} 0 \\ -1 \\ 1 \end{pmatrix}, \quad M_{\mathcal{F}}^{(0,1)} = \frac{1}{2(1-\kappa)} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & -\kappa & -\mu & 1 \\ 0 & -\kappa & \frac{1}{L} & \kappa & -\frac{1}{L} \\ 0 & -\mu & \kappa & \mu & -1 \\ 0 & 1 & -\frac{1}{L} & -1 & \frac{1}{L} \end{pmatrix},$$

$$v_{\mathcal{F}}^{(1,\star)} = \begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}, \quad M_{\mathcal{F}}^{(1,\star)} = \frac{1}{2(1-\kappa)} \begin{pmatrix} \mu & 0 & 0 & -\mu & \kappa \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -\mu & 0 & 0 & \mu & -\kappa \\ \kappa & 0 & 0 & -\kappa & \frac{1}{L} \end{pmatrix},$$

$$v_{\mathcal{F}}^{(1,0)} = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix}, \quad M_{\mathcal{F}}^{(1,0)} = \frac{1}{2(1-\kappa)} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & -1 & -\mu & \kappa \\ 0 & -1 & \frac{1}{L} & 1 & -\frac{1}{L} \\ 0 & -\mu & 1 & \mu & -\kappa \\ 0 & \kappa & -\frac{1}{L} & -\kappa & \frac{1}{L} \end{pmatrix},$$

$$v_{\mathcal{A}}^{(1)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad M_{\mathcal{A}}^{(1)} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix},$$

$$v_{\mathcal{A}}^{(2)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad M_{\mathcal{A}}^{(2)} = \frac{1}{2} \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 \end{pmatrix}.$$

Solving this SDP, we find the rate $\tau = \left(\frac{L-\mu}{L+\mu}\right)^2$.
Moreover, the corresponding dual values are

$$\lambda_{\mathcal{F}}^{\star,0} = \frac{2\mu(L-\mu)}{(L+\mu)^2}, \qquad \lambda_{\mathcal{F}}^{\star,1} = \frac{2\mu}{L+\mu},$$
$$\lambda_{\mathcal{F}}^{0,\star} = 0, \qquad \lambda_{\mathcal{F}}^{0,1} = \frac{L-\mu}{L+\mu},$$
$$\lambda_{\mathcal{F}}^{1,\star} = 0, \qquad \lambda_{\mathcal{F}}^{1,0} = 0,$$
$$\lambda_{\mathcal{A}}^{1} = \frac{2}{L+\mu}, \qquad \lambda_{\mathcal{A}}^{2} = 1.$$

Plugging those values in (Generic proof) builds a proof of convergence of (GDLS) with the guarantee $f(x_1) - f_\star \leqslant \left(\frac{L-\mu}{L+\mu}\right)^2 (f_0 - f_\star)$.

$$\begin{aligned}
& f(x_1) - f_\star - \left(\frac{L-\mu}{L+\mu}\right)^2 (f(x_0) - f_\star) \\
\leqslant\ & \tfrac{2\mu(L-\mu)}{(L+\mu)^2}\left(f(x_0) - f_\star + \langle \nabla f(x_0), x_\star - x_0\rangle + \tfrac{1}{2L}\|\nabla f(x_0)\|^2 + \tfrac{\mu}{2(1-\mu/L)}\|x_\star - x_0 + \tfrac{1}{L}\nabla f(x_0)\|^2\right) \\
& + \tfrac{2\mu}{L+\mu}\left(f(x_1) - f_\star + \langle \nabla f(x_1), x_\star - x_1\rangle + \tfrac{1}{2L}\|\nabla f(x_1)\|^2 + \tfrac{\mu}{2(1-\mu/L)}\|x_\star - x_1 + \tfrac{1}{L}\nabla f(x_1)\|^2\right) \\
& + \tfrac{L-\mu}{L+\mu}\left(f(x_1) - f(x_0) + \langle \nabla f(x_1), x_0 - x_1\rangle + \tfrac{1}{2L}\|\nabla f(x_0) - \nabla f(x_1)\|^2 + \tfrac{\mu}{2(1-\mu/L)}\|x_0 - \tfrac{1}{L}\nabla f(x_0) - x_1 + \tfrac{1}{L}\nabla f(x_1)\|^2\right) \\
& + \tfrac{2}{L+\mu}\langle \nabla f(x_1), \nabla f(x_0)\rangle \\
& + \langle \nabla f(x_1), x_1 - x_0\rangle \\
\leqslant\ & 0.
\end{aligned}$$

The first inequality holds independently on the chosen class. It simply results from terms rearrangement. By subtracting the LHS from the RHS, one would find a semi-definite positive quadratic form of the variables $x_0, x_1, \nabla f(x_0)$ and $\nabla f(x_1)$. The second inequality precisely uses the (in)equalities that are specific to the chosen class and algorithm. Note that the two algorithm constraints can be replaced by the sole constraint $\left\langle \nabla f(x_1), x_1 - x_0 + \frac{2}{L+\mu}\nabla f(x_0)\right\rangle = 0$ (as discussed in De Klerk et al. (2017)), showing that this guarantee also holds on the Gradient descent method with fixed steps-size $\frac{2}{L+\mu}$.

5.6 Lyapunov with PEPs

We saw in Subsection 5.4.1 that worst-case proofs essentially writes as (Generic proof):

$$\begin{aligned}
& \underbrace{\langle F, v_P\rangle + \langle G, M_P\rangle}_{\text{Performance metric}} - \tau \underbrace{[\langle F, v_I\rangle + \langle G, M_I\rangle]}_{\text{Initialization}} \\
& \leqslant \sum_j \lambda^{(j)} \underbrace{\left[\left\langle F, v^{(j)}\right\rangle + \left\langle G, M^{(j)}\right\rangle\right]}_{\text{Constraint}} \\
& \leqslant 0. \qquad (5.15)
\end{aligned}$$

Namely, the right linear combination of the available constraints upper bounds the difference between the performance metric and τ times the initial value. Sometimes those proofs can be relatively complicated and a simpler one can be desirable. In particular, this is the case when the algorithm under consideration is run for a few iterations. Lyapunov analyses typically allows reducing the worst-case analyses of T iterations to that of a single iteration, and therefore reducing the complexity of the proof.

For example, for (NAG), described as follow

$$\begin{aligned} \lambda_{t+1} &= \tfrac{1}{2} + \sqrt{\tfrac{1}{4} + \lambda_t^2} \\ y_t &= x_t + \tfrac{\lambda_t - 1}{\lambda_{t+1}}(x_t - x_{t-1}), \\ x_{t+1} &= y_t - \tfrac{1}{L}\nabla f(y_t). \end{aligned} \tag{NAG}$$

on $\mathcal{F}_{0,L}$, we often use the sequence

$$V_t = \lambda_t^2(f_t - f_\star) + \frac{L}{2}\|\lambda_t(x_t - x_\star) + (1-\lambda_t)(x_{t-1} - x_\star)\|^2 \tag{5.16}$$

providing a worst-case convergence guarantee $f(x_T) - f_\star = \mathcal{O}(1/T^2)$. In general, a direct way to find such a sequence is to consider

$$\begin{aligned} V_t = &\underbrace{[\langle F, v_I\rangle + \langle G, M_I\rangle]}_{\text{Initialization}} \\ &+ \sum_{\substack{j \mid \text{only involves} \\ \text{values observed} \\ \text{before step } t}} \lambda^{(j)} \underbrace{\left[\left\langle F, v^{(j)}\right\rangle + \left\langle G, M^{(j)}\right\rangle\right]}_{\text{Constraint}}. \end{aligned} \tag{5.17}$$

Applying this method on (NAG) provides the sequence of complete potential functions

$$\begin{aligned} V_t = &\lambda_t^2(f_t - f_\star) + \frac{L}{2}\|\lambda_t(x_t - x_\star) + (1-\lambda_t)(x_{t-1} - x_\star)\|^2 \\ &+ \frac{1}{2L}\sum_{s=1}^{t-1}[\lambda_{s+1}^2\|\nabla f(x_{s+1})\|^2 + \lambda_{s+1}\|\nabla f(y_s)\| \\ &\qquad + \lambda_s^2\|\nabla f(y_s) - \nabla f(x_s)\|^2] \end{aligned}$$

that allows for free (using the same inequalities as for proving that (5.16) is decreasing) to also conclude that $\min_{t\leqslant T}\|\nabla f(x_t)\|^2 = \mathcal{O}(1/T^3)$, as shown in (Monteiro and Svaiter, 2013, Theorem 5.2.d) and experimentally evidenced using PEPs in (Taylor et al., 2017c, Table 4).

Note that the cumulatively summed up constraints involve both class constraints and algorithm constraints. Therefore, this technique can be applied directly on (GFOM) while looking for an optimal algorithm, its rate, the corresponding proof and a sequence of potential functions at the same time.

5.7 Conclusion

Summary not only is the *performance estimation problem (PEP)* framework a powerful tool to automate the search of guarantees, but also it allows exhibiting general structure of proofs. Understanding this structure enables to generalize results onto larger class of functions or onto a class of methods, but also to find new optimization methods and study their convergence properties. Finally, it also enables to understand how to build a Lyapunov sequence of functions.

Open research directions all this framework relies on two major assumptions: the class constraints are known and homogeneous in $\|x\|^2$ and $\|\nabla f(x)\|^2$ and $f(x)$, and the method's update is a linear combination of previous iterates and observed oracle calls. Therefore, two interesting questions arise: can we automate the search of the interpolation conditions?

And, how can we generalize this framework to non homogeneous class of functions or to non linear methods such as adaptive step-size based methods? A few works already investigate this direction for some specific methods. In particular, Barré et al. (2020) studies a variant of the Heavy-ball method Polyak (1963) using Polyak step-sizes, also discussed in (Barré, 2021, Chapter 4). On the other hand, Gupta et al. (2023) uses PEP techniques to provide worst-case guarantees on several variants of non-linear conjugate gradient methods.

6

Optimal first-order methods for convex functions with a quadratic upper bound

We analyze worst-case convergence guarantees of first-order optimization methods over a function class extending that of smooth and convex functions. This class contains convex functions that admit a simple quadratic upper bound. Its study is motivated by its stability under minor perturbations. We provide a thorough analysis of first-order methods, including worst-case convergence guarantees for several methods, and demonstrate that some of them achieve the optimal worst-case guarantee over the class. We support our analysis by numerical validation of worst-case guarantees using performance estimation problems. A few observations can be drawn from this analysis, particularly regarding the optimality (resp. and adaptivity) of the Heavy-ball method (resp. Heavy-ball with line-search). Finally, we show how our analysis can be leveraged to obtain convergence guarantees over more complex classes of functions. Overall, this study brings insights into the choice of function classes over which standard first-order methods have working worst-case guarantees.

This chapter is based on our work "Optimal first-order methods for convex functions with a quadratic upper bound" (co-authored with A. Taylor, and A. Dieuleveut), currently under review.

Contents

6.1 Introduction

In this paper, we consider the problem of minimizing a convex (closed proper) function

$$f_\star \triangleq \min_{x\in\mathbb{R}^d} f(x), \tag{6.1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is assumed to have a non-empty compact set of global minimizers denoted by $\mathcal{X}_\star$ (which is necessarily convex). Convergence properties of first-order optimization are typically analyzed through worst-case analyses under the *black-box* model (Nemirovskii and Yudin, 1983a). In this formalism, nontrivial guarantees are obtained by assuming the function to be minimized to satisfy certain regularity conditions. In particular, it is common to assume Lipschitz continuity of the gradients of f (also often referred to as smoothness of f) as well as (strong) convexity of f. First-order methods and their analyses for minimizing such functions in the black-box model occupied a great deal of attention, see, e.g., (Nemirovskii and Yudin, 1983a; Polyak, 1987; Nesterov, 2003).

Due to the practical success of first-order methods in large-scale applications, and particularly when aiming for only low to medium accuracy solutions (Bottou and Bousquet (2007) motivate this goal for machine learning), a few trends emerged in the first-order optimization literature. Among them, a particular focus concerned the question of understanding minimal working assumptions under which one could design efficient first-order methods. In other words, many authors looked for weaker/alternate versions to the standard smoothness and strong convexity-type assumptions, still allowing to obtain suitable working guarantees for standard first-order methods.

Relaxations of strong convexity-type assumptions. Convexity alone is not sufficient to a priori guarantee "fast" convergence of usual first-order methods. On the other hand, strong convexity allows to obtain faster (geometric) rates but is a very strong condition. Therefore, many authors studied conditions in between convexity and strong convexity, aiming to obtain faster rates under relatively generic assumptions. In particular, different authors considered the restricted secant inequality (Zhang and Yin, 2013; Guille-Escuret et al., 2022), the error bound (Luo and Tseng, 1993), Łojasiewicz-type inequalities (Polyak, 1963), and many more (Hazan et al., 2015; Kurdyka, 1998; Liu and Wright, 2015; Gong and Ye, 2014; Necoara et al., 2019; Hardt et al., 2018; Abbaszadehpeivasti et al., 2023). Relations between these assumptions were treated at length in (Bolte et al., 2017; Zhang, 2017b). Among those, one of the weakest relaxation is the so-called (lower) quadratic growth, see (Bonnans and Ioffe, 1995; Ioffe, 1994; Anitescu, 2000). Recently, those notions turned out to be useful, e.g. for studying proximal gradient methods, see (Cui et al., 2017; Drusvyatskiy and Lewis, 2018; Drusvyatskiy and Ioffe, 2015; Peng et al., 2020; Zhang, 2017a; Chieu et al., 2021). In the rest of the paper, we focus on (non strongly) convex functions.

Relaxations of smoothness-type assumptions. Generalization of smoothness assumptions were less investigated in the literature. Still, a few such relaxations have emerged, including the relative smoothness, see (Bauschke et al., 2017; Lu et al., 2018; Dragomir et al., 2021; Hanzely et al., 2021), restricted smoothness (Agarwal et al., 2012) and restricted Lipschitz-continuous gradient (Zhang and Yin, 2013). In this work, we consider instead the set of convex functions satisfying the (upper) *quadratic growth* condition, as follows.

Definition 6.1.1. *A function f is L-quadratically upper bounded (denoted L-QG^+) if for all $x \in \mathbb{R}^d$:*

$$f(x) - f_\star \leqslant \frac{L}{2} d(x, \mathcal{X}_\star)^2,$$

where $d(x, \mathcal{X}_\star) = \min_{x_\star \in \mathcal{X}_\star} \|x - x_\star\|_2$. We denote the set of such functions by $\mathrm{QG}^+(L)$, and by QG^+ when L is left unspecified.

This assumption is weaker than smoothness. First, any L-smooth function (i.e. with L-Lipschitz gradient) also belongs to $\mathrm{QG}^+(L)$. On the other hand,

1. Some functions do belong to QG^+ while not being smooth for any value of L. In particular QG^+ contains all Lipschitz non-smooth convex functions which are twice differentiable at all their optimal points. Let us mention a few rules for obtaining (not necessarily smooth) QG^+ functions: (i) any function that can be written as $x \mapsto h(x^\top M x)$, where h is convex Lipschitz continuous and M is a positive semidefinite matrix, or (ii) $x \mapsto h(N(x))$, where h is convex and smooth (or QG^+) and N is a norm (e.g. $N = \|\cdot\|_2$, $N = \|\cdot\|_1$ or $N = \|\cdot\|_\infty$).
2. Some functions belong to $\mathrm{QG}^+(L_Q)$ while being smooth only for some $L_S \gg L_Q$ (see e.g., Eq.(3) and Proposition 4.6 in (Guille-Escuret et al., 2021).) Consequently, even though the worst-case convergence rate over the class of $\mathrm{QG}^+(L)$ functions cannot improve on the rate for L-smooth functions, it is possible for a given smooth function that the guarantee provided by the rate on the QG^+ class is actually *better* than the one resulting from the rate as a smooth function.

More generally, the latter example is related to *condition continuity*, introduced by Guille-Escuret et al. (2021, Definition 4.9). *Condition continuity* is a property of function classes, defined by the fact that a minor modification of the gradient of the function, at any point away from the optimum, cannot strongly affect the class parameter L. This is a desirable property for first-order methods for which the output typically continuously depends on the gradients of the functions minimized: if the function is slightly perturbed away from the optimum, the tuning and convergence guarantees of the algorithm should not be affected. The QG^+ class satisfies condition continuity, while the class of smooth convex functions does not: a minor perturbation of an L-smooth function (thus L-QG^+) can be L_Q-QG^+ and L_S-smooth, with $L_S \gg L_Q$. This also motivates studying QG^+.

Contributions and organization of the paper. The rest of the paper is organized in two main sections. First, in Section 6.2, we analyze a few first-order methods, namely the subgradient method and the Heavy-ball method with and without a line-search. We provide worst-case complexity bounds on the convergence rates as well as corresponding lower complexity bounds. We also provide a lower complexity bound for minimizing convex functions in QG^+ via first-order methods. Finally, we provide interpolation/extensions results for this class of problems, which allows exploring/deriving all previous results in a principled way (using performance estimation problems Drori and Teboulle (2014); Taylor et al. (2017c)). We summarize those results in Table 6.1, together with precise references to the corresponding statements. Secondly, we review the main consequences of our analysis in Section 6.3. More specifically, we underline the facts that (a) the Heavy-ball Algorithm 10 and 11 are optimal on this class of functions, furthermore, (b) Algorithm 11 is adaptive: it achieves the optimal convergence rate for both Lipschitz-continuous functions and QG^+ functions without requiring knowledge of any class parameter. Then, we describe how our theory can be exploited for automatically obtaining convergence rates for different

Table 6.1: Summary of the worst-case guarantees obtained after n iterations of a few different first-order methods on the class of L-QG^+ convex functions, which are obtained in Section 6.2 and Section 6.A. Two main methods are studied, namely the (sub)gradient and the Heavy-ball methods. Some guarantees concern the worst-case function value accuracy at the last iterate. The term "average" in the third column refers to the function value on the Polyak-Rupert averaged iterate. All the provided bounds are proportional to $R^2 = d(x_0, \mathcal{X}_\star)^2$, where x_0 denotes the starting point of the methods.

Method	Step-sizes $(\gamma_t)_{0\leqslant t\leqslant n-1}$		Iterate	Upper bound		Lower bound	
Subgradient (Sec. 6.2.1)	$\frac{1}{L}$	(Alg. 9)	Average	$\frac{L}{2}\frac{R^2}{n+1}$	(Th. 6.2.1)	$\frac{L}{2}\frac{R^2}{n+1}$	(Rem. 6.A.1)
	γ_t	(Alg. 9)	Last	$\times$		$\frac{LR^2}{2}L\gamma_{n-1}$	(Th. 6.2.2)
	$\sim \frac{1}{2L\sqrt{t}}$	(Alg. 14)	Last	$\sim \frac{LR^2}{4\sqrt{n}}$	(Conj. 6.A.3)	$\sim \frac{LR^2}{4\sqrt{n}}$	(Th. 6.2.2)
Heavy-ball (Sec. 6.2.3)	$\frac{1}{L}\frac{1}{t+2}$	(Alg. 10)	Last	$\frac{L}{2}\frac{R^2}{n+1}$	(Th. 6.3.3)	$\frac{L}{2}\frac{R^2}{n+1}$	
	line-search	(Alg. 11)	Last	$\frac{L}{2}\frac{R^2}{n+1}$	(Th. 6.3.3)	$\frac{L}{2}\frac{R^2}{n+1}$	
First-order (Sec. 6.2.2)	Any		Any	-		$\frac{L}{2}\frac{R^2}{n+1}$	(Th. 6.2.3)

classes of functions. Lastly (in Section 6.G), we discuss results that can be obtained when restricting the class to functions satisfying additional assumptions (a relaxation of strong-convexity).

Notation and background results. For problem (6.1), the set $\mathcal{X}_\star$ of minimizers of f is closed and convex (f is proper closed and convex by assumption). Therefore, there exists a unique projection $\pi_{\mathcal{X}_\star}$ onto $\mathcal{X}_\star$, verifying: $\|x - \pi_{\mathcal{X}_\star}(x)\|_2 = d(x, \mathcal{X}_\star)$. We use the classical ∂f for denoting the subdifferential of the function f. Namely, the subdifferential of f at $x \in \mathbb{R}^d$ is the set of all subgradients of f at x: $\partial f(x) = \{g | \forall y \in \mathbb{R}^d, f(y) \geqslant f(x) + \langle g | y - x\rangle\}$. Note that $0 \in \partial f(x) \Leftrightarrow x \in \mathcal{X}_\star$; moreover, if $f \in \mathrm{QG}^+$, then for all $x \in \mathcal{X}_\star$, $\partial f(x) = \{0\}$.

6.2 A few worst-case guarantees for minimizing QG^+ convex functions

In this section, we provide the main technical results of this paper, summarized in Table 6.1. In Subsection 6.2.1, we study the behavior of a *(sub)gradient* method on convex QG^+ functions. A lower complexity bound on the convergence of any first-order method is provided in Subsection 6.2.2. In Subsection 6.2.3, we introduce the Heavy-ball method under consideration and prove its worst-case optimality. Finally, we discuss how interpolation conditions were used for obtaining these results in Subsection 6.2.4.

6.2.1 (Sub)gradient method on QG^+ convex functions

In this subsection, we consider Algorithm 9: the subgradient method, for n iterations and a sequence of step-sizes $(\gamma_t)_{0\leqslant t\leqslant n-1}$. The following result provides a convergence guarantee for the averaged function value accuracy throughout the iterative procedure.

Theorem 6.2.1. (Convergence of Algorithm 9 in average) *Let f be an L-QG^+ convex function. Applying (sub)gradient method on f with step-size $\gamma \triangleq \frac{1}{L}$ leads to the following guarantee:*

$$\frac{1}{n+1}\sum_{k=0}^{n}(f(x_k)-f_\star) \leqslant \frac{L}{2}\frac{1}{n+1}d(x_0,\mathcal{X}_\star)^2. \tag{6.2}$$

Sketch of proof. The proof consists in proving that at each step $f(x_k)-f_\star \leqslant \frac{L}{2}d(x_k,\mathcal{X}_\star)^2 - \frac{L}{2}d(x_{k+1},\mathcal{X}_\star)^2$ and recognizing a telescopic sum on the right hand side. See Appendix 6.A. ■

Algorithm 9 Subgradient method

Input: x_0, $(\gamma_t)_{0\leqslant t\leqslant n}$
for $k = 1 \dots n$ **do**
  Query $g_{k-1} \in \partial f(x_{k-1})$;
  $x_k \leftarrow x_{k-1} - \gamma_{k-1} g_{k-1}$
Output: $(x_t)_{0\leqslant t\leqslant n}$

By convexity of f, this result automatically implies a convergence guarantee for the Polyak-Ruppert (PR) averaging (Polyak and Juditsky, 1992; Ruppert, 1988), $\bar{x}_n = \frac{1}{n+1}\sum_{k=0}^{n} x_k$ with the same convergence rate. For L-smooth convex functions, the same worst-case convergence rate is achieved by both the PR averaging and the last iterate. It is therefore natural to wonder if the subgradient method verifies the same convergence guarantee for the *last iterate*, on QG^+ convex functions. For L-smooth convex functions, Drori and Teboulle (2014, Theorem 3.2) provide the following lower bound on the convergence of Algorithm 9: for any n and any constant sequence of step-sizes $\gamma_k = \gamma \in [0, 2/L]$, there exists an L-smooth convex function f and a starting point x_0 s.t.

$$f(x_n) - f_\star \geqslant \max\left(\frac{L}{2}\frac{1}{1+2nL\gamma}, \frac{L}{2}(1-L\gamma)^{2n}\right) d(x_0,\mathcal{X}_\star)^2. \tag{6.3}$$

Drori and Teboulle (2014, Theorem 3.1) also provide a corresponding worst-case guarantee of the form $f(x_n) - f_\star \leqslant \frac{L}{2}\frac{1}{1+2nL\gamma}d(x_0,\mathcal{X}_\star)^2$ for when $\gamma \in (0, \frac{1}{L})$, which ensures convergence in function value accuracy with a constant step-size rule $\gamma_k = \gamma \in (0, \frac{1}{L})$ at a rate $O(1/n)$.

Here we provide a stricter lower bound for the convergence of the function value for the last iterate: contrary to what happens for smooth convex functions, subgradient methods with constant step-sizes cannot be guaranteed to converge on QG^+ convex functions.

Theorem 6.2.2. (Lower bound for Algorithm 9 - final iterate). *Let $n \in \mathbb{N}$. For any sequence $(\gamma_i)_{0\leqslant i\leqslant n-1} > 0$ and any $\varepsilon > 0$, there exists an L-QG^+ convex function f that verifies, after n iterations of Algorithm 9 with step-sizes $(\gamma_i)_{0\leqslant i\leqslant n-1}$,*

$$f(x_n) - f_\star \geqslant \frac{L}{2}L\gamma_{n-1}d(x_0,\mathcal{X}_\star)^2 - \varepsilon. \tag{6.4}$$

Sketch of proof. The proof consists in finding a function defined on $\mathbb{R}^3$ such that all the iterates except the last one are very close to each other. As QG^+ convex functions might not be differentiable, subgradients might vary very quickly. Therefore, the last iterate might be far away from the others, although all the previous iterates are clustered. A complete proof is provided in Appendix 6.A. ■

As a result, it is necessary to enforce $\gamma_n \to 0$ for ensuring convergence of Algorithm 9 on all problem instances. On the other hand, a similar lower bound to (6.3) (using the same Huber function as that used for the class of smooth convex functions, see (Drori and Teboulle, 2014), or (Taylor et al., 2017c, Section 4)) and modifying x_0 to account for varying step-sizes (we use $x_0 = 1 + 2\sum_{k=0}^{n-1} L\gamma_k$), one can obtain:

$$f(x_n) - f_\star \geqslant \frac{L}{2}\frac{1}{1+2\sum_{k=0}^{n-1}L\gamma_k}d(x_0,\mathcal{X}_\star)^2. \tag{6.5}$$

Consequently the worst-case convergence is slower than $O(1/n)$ as soon as $\gamma_k \to 0$. Overall, the convergence is provably worse for the last iterate over the $\mathrm{QG}^+(L)$-class than over the class of L-smooth convex functions, even though guarantees match for the PR-averaged iterate. Actually, the lower bound is at least the maximum of the RHSs of Equations (6.4) and (6.5). In Section 6.A, we introduce and analyze Algorithm 14, that corresponds to Algorithm 9 with a specific sequence of step-sizes such that Equations (6.4) and (6.5) are equal. This results in a decaying sequence of step-sizes scaling as $O(1/\sqrt{n})$. The next section is devoted to a lower complexity bound on the convergence in function accuracy for any black-box first-order method.

6.2.2 First-order lower bound

The next theorem guarantees that no black-box first-order method can beat a $O(1/n)$ worst-case guarantee in function values uniformly on the set of QG^+ convex functions.

Theorem 6.2.3. (Lower complexity bound) *Let $n \in \mathbb{N}$. There exist some $d \in \mathbb{N}$ and some convex L-QG^+ function f of input space $\mathbb{R}^d$ such that: for any sequence $(x_k)_{0\leqslant k\leqslant n}$ satisfying $x_k - x_0 \in \mathrm{span}\{g_0, g_1, g_2, \dots, g_{k-1}\}$ for all $k \leqslant n$ with $g_i \in \partial f(x_i)$ $(i = 0, \dots, k-1)$, we have:*

$$f(x_n) - f_\star \geqslant \frac{L}{2}\frac{1}{n+1} d(x_0, \mathcal{X}_\star)^2.$$

Sketch of proof. The proof consists in noticing that the function $x \mapsto \frac{L}{2}\|x\|_\infty^2$ explores one new dimension per step and that this new dimension is independent of the way the next point in the sequence is chosen. (Note that a similar function is used in Drori and Teboulle (2016, Appendix A) to obtain lower bounds of gradient methods.) Therefore, choosing $d = n + 1$ explores that there exists one *unseen* dimension after n iterations. This methodology is common to prove lower bounds in first-order optimization, see, e.g., (Nemirovskii, 1994; Nesterov, 2003; Bubeck, 2015). We refer to Appendix 6.B.1 for the complete proof. The above result is also generalized in Appendix 6.B.2 to account for *any* sequence generated by a black-box first-order (possibly without the span assumption as in, e.g., (Nemirovskii, 1994, Chapter 12) for quadratic minimization). ■

One can conclude from Theorem 6.2.3 that no black-box first-order method can enjoy a worst-case guarantee better than $f(x_n) - f_\star \leqslant \frac{L}{2}\frac{1}{n+1} d(x_0, \mathcal{X}_\star)^2$ uniformly on all $d \in \mathbb{N}$, all $f : \mathbb{R}^d \to \mathbb{R}$ that is convex and L-QG^+ and all $x_0 \in \mathbb{R}^d$. This entails that Algorithm 9, with constant step-size $1/L$ and PR averaging, is worst-case optimal for decreasing function values on the class of QG^+ and convex functions. In the next section, we introduce two alternate methods that also achieve this optimal bound, this time without PR averaging. As we see in the sequel, those further developments allow achieving this optimal bound without explicitly using the knowledge of the constant L.

6.2.3 Two methods with optimal last iterate guarantee

In this section, we introduce Algorithm 10 and Algorithm 11 which both achieve the optimal convergence guarantee for the last iterate (see Theorem 6.2.3). The first of

Algorithm 10 Heavy-ball method for QG^+ convex

Input: x_0, L

for $k = 1 \dots n$ **do**

  Pick g_{k-1} from $\partial f(x_{k-1})$

  $x_k \leftarrow \frac{k}{k+1}x_{k-1} + \frac{1}{k+1}x_0 - \frac{1}{k+1}\sum_{i=0}^{k-1}\frac{1}{L}g_i$

Output: x_n

those two methods explicitly relies
on the knowledge of the class parameter L for performing its updates, whereas the second variant allows avoiding using any knowledge on L. Note that the update rule from Algorithm 10 can equivalently be expressed as:

$$x_k \leftarrow x_{k-1} - \frac{1}{L}\frac{1}{k+1}g_{k-1} + \frac{k-1}{k+1}(x_{k-1} - x_{k-2}) \tag{6.6}$$

where $g_{k-1} \in \partial f(x_{k-1})$. This formulation corresponds to the Heavy-ball method, as defined in (Ghadimi et al., 2015, Theorem 2) and for which authors provided a $O(1/n)$ guarantee for L-smooth convex functions.

Algorithm 11 Heavy-ball method with line-search for QG^+ convex

Input: x_0, $v_0 \leftarrow 0$
for $k = 1 \dots n$ **do**
  $y_k \leftarrow \frac{k}{k+1}x_{k-1} + \frac{1}{k+1}x_0$
  Pick $g_{k-1} \in \partial f(x_{k-1})$ such that $\langle g_{k-1}, v_{k-1}\rangle = 0$.
  $v_k \leftarrow v_{k-1} + g_{k-1}$
  $\alpha_k \leftarrow \arg\min_\alpha f(y_k + \alpha v_k)$
  $x_k \leftarrow y_k + \alpha_k v_k$
Output: x_n

Algorithm 11 takes a similar form, but relies on an exact line-search procedure, avoiding to use any knowledge on L. Both methods share the same worst-case guarantee, matching the lower bound result from Theorem 6.2.3. The following theorem provides a necessary condition for an algorithm to share this same worst-case guarantee.

Theorem 6.2.4. (Main result: sufficient condition for being worst-case optimal). *Let $\mathcal{A}$ be an iterative first-order method that verifies, for all convex $\mathrm{QG}^+(L)$ function f, and starting points x_0,*

$$\left\langle g_k, x_k - \left[\frac{k}{k+1}x_{k-1} + \frac{1}{k+1}x_0 - \frac{1}{k+1}\sum_{i=0}^{k-1}\frac{1}{L}g_i\right]\right\rangle \leqslant 0. \tag{6.7}$$

for some sequence $(g_i)_{i\in\mathbb{N}}$ of subgradients $g_i \in \partial f(x_i)$, and where $(x_i)_{i\in\mathbb{N}}$ are the iterates of $\mathcal{A}$. Then, the output x_n of $\mathcal{A}$ achieves the worst-case guarantee:

$$f(x_n) - f_\star \leqslant \frac{L}{2}\frac{1}{n+1}d(x_0, \mathcal{X}_\star)^2.$$

Sketch of proof. The proof is based on a Lyapunov analysis; for $k \geqslant 0$, we define the sequence $k(f(x_{k-1}) - f_\star) + \frac{L}{2}\|x_0 - \pi_{\mathcal{X}_\star}(x_0) - \sum_{i=0}^{k-1}\frac{1}{L}g_i\|^2$ and show it is a decreasing. See Appendix 6.C for a complete and detailed proof. ■

Inequality (6.7) is clearly satisfied for Algorithm 10 by ensuring the right hand side of the inner product being identically 0. For Algorithm 11, the right hand side of the inner product in (6.7) is colinear to the search direction $\sum_{i=0}^{k-1} g_i$. First-order optimality conditions of the exact line-search procedure enforces the inner product in (6.7) to be identically 0. The next corollary follows.

Corollary 6.2.5. *Let $n \in \mathbb{N}$, $d \in \mathbb{N}$, f be a convex QG^+ function, and $x_0 \in \mathbb{R}^d$. Also, let x_n be the output of either Algorithm 10 or Algorithm 11, we have: $f(x_n) - f_\star \leqslant \frac{L}{2}\frac{1}{n+1}d(x_0, \mathcal{X}_\star)^2$.*

In the next section, we discuss how such worst-case analyses were obtained in a principled way, through so-called performance estimation problems (PEPs). An important ingredient to use this methodology is to develop *interpolation* (a.k.a. *extension*) results for the convex QG^+ class.

6.2.4 Extension/interpolation results for QG^+ convex functions

The problem of interpolating/extending within a class of functions can be stated as follows. Given a set of triplet $(x_i, g_i, f_i)_{i\in I} \subset \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}$ for some $d \in \mathbb{N}$ and some index set I, the question of interest is that of recovering a function f in a prescribed class of convex functions $\mathcal{F}$ satisfying

$$f_i = f(x_i) \text{ and } g_i \in \partial f(x_i) \text{ for all } i \in I.$$

A similar problem, often referred to as convex integration, consists in finding such functions by only specifying some subgradients but no function values; see (Rockafellar, 1997); for the case where $\mathcal{F}$ is the class of (closed and proper) convex functions, this problem was also treated at length in (Lambert et al., 2004). Motivated by applications to performance estimation problems (see below), this problem was studied in (Taylor et al., 2017c) for the cases where $\mathcal{F}$ is the class of closed proper (possibly strongly) convex (possibly smooth) functions. In this case, it is possible to obtain simple necessary and sufficient conditions for the set $(x_i, g_i, f_i)_{i\in I}$ to be interpolable; we refer to those conditions as *interpolation conditions*. Such conditions take the form of a set of inequalities on $(x_i, g_i, f_i)_{i\in I}$, and sometimes allow to conveniently deal with discrete versions of functions within a certain class $\mathcal{F}$ (for which we have interpolation conditions at our disposal). There exists a few classes of functions, typical for the analysis of first-order methods, for which such conditions exist, see, e.g., (Taylor et al., 2017a, Theorem 3.3–3.6, Theorem 3.10). The next theorem provides interpolation conditions for the class of convex QG^+ functions.

Theorem 6.2.6. (Interpolation conditions) *Let $(x_i, g_i, f_i)_{i\in I}$ a family of elements in $\mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}$. Set $I_\star$ the (assumed) non-empty subset of I of the indices of elements (x_i, g_i, f_i) verifying $g_i = 0$.*
Then, there exists a $\mathrm{QG}^+(L)$ and convex function f interpolating those points (i.e. such that $\forall i \in I, f(x_i) = f_i$ and $g_i \in \partial f(x_i)$) if and only if

$$\forall i \in I, \forall j \in I, \quad f_i \geqslant f_j + \langle g_j, x_i - x_j\rangle \tag{6.8}$$

$$\forall i \in I_\star, \forall j \in I, \quad f_i \geqslant f_j + \langle g_j, x_i - x_j\rangle + \frac{1}{2L}\|g_j\|^2. \tag{6.9}$$

Sketch of proof. The proof is derived in two steps. First we notice that (6.8) corresponds to the convexity of the function, and we prove (6.9) combining the 2 inequalities respectively corresponding to convexity and L-QG^+ assumptions. Reciprocally, we explicitly build a L-QG^+ convex function from (6.8) and (6.9). See Appendix 6.E for a detailed proof. ■

Application to Performance estimation problems (PEPs). PEPs were introduced by Drori and Teboulle (2014) for developing new analyses of first-order methods; see also (Drori, 2014; Kim and Fessler, 2016) for the first works on this topic. PEPs were later formalized using the concept of *convex interpolation* by (Taylor et al., 2017c,a). PEPs formulate the search for worst-case guarantees as infinite dimensional optimization problems over the considered class of functions, e.g.,

$$\text{Worst-case}\left(\text{Alg. } 9, n=1, \gamma=\tfrac{1}{L}, \text{ convex } \mathrm{QG}^+(L)\right) \triangleq \max_{\substack{f \in \mathrm{QG}^+(L) \text{ convex} \\ x_1 \in x_0 - \frac{1}{L}\partial f(x_0)}} \frac{f(x_1) - f_\star}{d(x_0, \mathcal{X}_\star)^2},$$

for the case of 1-step subgradient descent. In order to numerically solve those problems, it is needed to transform them into a finite dimensional problem. To that end, interpolation conditions play a crucial role, by allowing to reduce the optimization over the (infinite dimensional) class of functions to an optimization over a constrained set of vectors, thus finite dimensional problem.

Consequently, Theorem 6.2.6 allows us to use the PEP framework to study the class of $\mathrm{QG}^+(L)$ convex functions. Therefore, we obtained and verified all the results of this work thanks to the different programming tools that have recently been developed for easing the access to the PEP framework (see the packages (Taylor et al., 2017b; Goujaud et al., 2022a)). For each algorithm studied in this paper, PEPs also provided the associated tight bound. Finally, PEPs also guided us to prove the lower bound of Theorem 6.2.3 thanks to the study of the greedy first-order method (*GFOM*) from (Drori and Taylor, 2020).

6.3 Discussion and concluding remarks

In this section, we discuss a few takeaways of the results. The messages of this section include optimality and adaptivity results for Heavy-ball with a line-search, a discussion on the applicability of this functional class beyond its simple use, as well as a few words on the limitations of considering this simple class of convex functions.

6.3.1 Optimality of HB algorithm

An ever-recurring question in the field of optimization is the convergence of HB methods on smooth and strongly convex functions. On the one hand, HB is optimal (in the sense that it achieves the optimal worst-case guarantee) for convex quadratic objectives and can then be seen as a variant of Chebyshev iterative method (Flanders and Shortley, 1950; Lanczos, 1952; Young, 1953). Even with a simple (constant) tuning of the step-size and momentum parameters, it is optimal (i.e., achieving rates $O(1/n^2)$ for L-smooth convex quadratics, and $(1 - O(\sqrt{\mu/L}))^n$ if the problem is also μ-strongly convex.).

On the other hand, the method does not generalize well to (non quadratic) smooth strongly convex functions: Lessard et al. (2016, Figure 7) built a function for which the Heavy-ball method, tuned with the same dependence to L and μ than for the quadratic case, fails to converge. More generally, while other sets of hyper-parameters allow to obtain convergence for the class of smooth and (strongly) convex (Ghadimi et al., 2015), Heavy-ball was never showed to accelerate w.r.t. the Gradient descent method, and the possibility to obtain such an acceleration remains an open question to the best of our knowledge. Simultaneously, Nesterov's accelerated gradient method (Nesterov, 1983) does achieve such an acceleration.

In summary, while for quadratic convex problems it has the optimal worst-case guarantee, Heavy-ball is believed not to satisfy this property for smooth and (strongly) convex functions. Interestingly, Corollary 6.2.5 shows that Heavy-ball is optimal (with rate $O(1/n)$) over the (larger) class QG^+ convex.

This observation questions the existence of intermediary classes (smaller than QG^+ convex and containing quadratic functions), over which Heavy-ball would achieve a $O(1/n^2)$ convergence guarantee. In the next section, we discuss the adaptivity of the method.

6.3.2 Adaptivity of HB line-search algorithm 3

The search for adaptive and parameter free methods is a major challenge in optimization as the regularity of the function (both in terms of class and class-parameter L) is often unknown. In the rest of the section, we discuss the fact that Algorithm 11 provides a parameter-free and adaptive method for $\mathrm{QG}^+(L)$ and M-Lipschitz functions. Those results are summarized in Table 6.2.

Table 6.2: Optimality of the proposed methods over the set of QG^+ convex functions and M-Lipschitz convex functions. ELS: Exact Line-Search. ✓ indicates optimality among the class and × the contrary. All counter examples are given in App. 6.D. †: constants resulting in optimal convergence rates depend on the class, thus for example heavy-ball with step-size $\frac{\text{constant}}{(t+2)}$ is not adaptive as it does not achieve the optimal rate for both classes with the same constant. ‡: up to a $\log$ factor.

Method			Function class				Parameter free
Algorithm	Step-sizes $(\gamma_t)_{0\leqslant t\leqslant n-1}$	Iterate	$\mathrm{QG}^+(L)$ convex		M-Lipschitz convex		
Subgradient (Alg. 9)	constant†	Average	✓	(Thm. 6.2.1)	×	(Thm. 6.D.1)	×
Subgradient (Alg. 14)	constant$^\dagger/\sqrt{t}$	Average	×	(6.5)	✓‡	(Nesterov, 2003, Sec. 3.2.3)	×
Subgradient (Alg. 15)	ELS	Average	×	(Thm. 6.D.2)	×	(Thm. 6.D.2)	✓
Subgradient (Alg. 15)	ELS	Last	×	(Thm. 6.D.2)	×	(Thm. 6.D.2)	✓
Heavy-ball (Alg. 10)	constant$^\dagger/(t+2)$	Last	✓	(Cor. 6.2.5)	✓	(Drori and Taylor, 2020, , Cor. 3)	×
Heavy-ball (Alg. 11)	ELS	Last	✓	(Cor. 6.2.5)	✓	(Drori and Taylor, 2020, , Cor. 4)	✓

Non-convergence of the line-search version of Algorithm 9. While the line-search version of Algorithm 10 (i.e. Algorithm 11) allows to get rid of the knowledge of L without degrading the convergence guarantee for the latest iterate, this is not the case for subgradient method. In fact subgradient with line-search Algorithm 15 does not converge, even when looking at the PR averaged iterate (see Theorem 6.D.2). It thus destroys the guarantee given in Theorem 6.2.1 for Algorithm 9 with constant step-size $1/L$.

Algorithm 9 is not adaptive to the class of Lipschitz functions. While Algorithm 9 with averaging and constant step-size $1/L$ is optimal for the class $\mathrm{QG}^+(L)$ convex, for any sequence of constant step-sizes, neither the average nor the last iterate of Algorithm 9 converge over the class of M-Lipschitz functions (see Theorem 6.D.1). On the other hand, to obtain a nearly (up to a log factor) optimal convergence rate $O(\log(n)/\sqrt{n})$, one can use $\gamma_t = \text{constant}/\sqrt{t}$ for $t \in \{1, \dots, n\}$, but such a sequence degrades the converge for QG^+ as proved by the lower bound Theorem 6.2.2.

Adaptivity of HB line-search (Algorithm 11). While Algorithm 11 requires to perform exact line-search steps, its first advantage over Algorithm 10 is not requiring the knowledge of the class parameter L. Algorithm 11 is thus adaptive to the class parameter L. This is analogous of *SSEP-based subgradient method* presented in Drori and Taylor (2020, Corollary 3) and its line-search version (See Drori and Taylor, 2020, Corollary 4) that are both optimal on the class of Lipschitz continuous and convex functions. The first one requires the knowledge of the parameter class M, the distance of the starting point to optimum $d(x_0, \mathcal{X}_\star)$ as well as the number of performed steps, while the second one replaces this

knowledge by exact line-search steps. Moreover, we note that *SSEP-based subgradient method* and Algorithm 10 are very similar to each other. Indeed, their respective update steps can be written as $x_k \leftarrow \frac{k}{k+1}x_{k-1} + \frac{1}{k+1}x_0 - \frac{d(x_0,\mathcal{X}_\star)}{M\sqrt{N+1}}\frac{1}{k+1}\sum_{i=0}^{k-1} g_i$ and $x_k \leftarrow \frac{k}{k+1}x_{k-1} + \frac{1}{k+1}x_0 - \frac{1}{L}\frac{1}{k+1}\sum_{i=0}^{k-1} g_i$.

Remarkably, only the two constants in front of $\frac{1}{k+1}\sum_{i=0}^{k-1} g_i$ differ between the two methods. Thus, their two line-search versions are identical. We conclude from Corollary 6.2.5 and Drori and Taylor (2020, Corollary 4) that Algorithm 11 is optimal on the classes of Lipschitz convex functions *and* on the classes of QG^+ convex functions. One can run this algorithm without knowing the type of conditions that are verified by the objective function and still benefit from a guarantee of optimality. This is a significant argument in favor of Algorithm 11.

6.3.3 Leveraging our analysis to obtain convergence bounds on other classes

One of the major limitations of the QG^+ class is that all functions within the class must be twice differentiable at the set of optimal points. This typically excludes some Lipschitz functions, as for example $x \mapsto \|x\|_1$ or problems involving Lasso regularization. In this section, we show that our Section 6.2 can be leveraged to automatically obtain rates on more complete classes of functions, that combine the limitations of the Lipschitz convex and QG^+ convex classes.

To introduce these classes, we denote h a generic function defined on $\mathbb{R}^+$ and verifying: $h(0) = 0$, h is strictly increasing and h is concave. We consider the class of functions with h *relative growth*.

Definition 6.3.1. *A function f is h-relatively upper bounded (denoted h-RG^+) if for all $x \in \mathbb{R}^d$:*

$$f(x) - f_\star \leqslant h\left(d(x,\mathcal{X}_\star)^2\right).$$

We denote the set of such functions by $\mathrm{RG}^+(h)$, and by RG^+ when h is left unspecified.

These classes enable to cover the QG^+ classes, the Lipschitz convex classes, and more.

Remark 6.3.2. *(Examples) These three examples of functions h satisfy the required assumptions:*

1. *When h is the linear function $h : z \mapsto \frac{Lz}{2}$, then simply $\mathrm{RG}^+(h) = \mathrm{QG}^+(L)$.*
2. *When $h : z \mapsto M\sqrt{z}$, then simply $\mathrm{RG}^+(h) \cap \{convex\} = \{M\text{-Lipschitz continuous convex}\}$.*
3. *When $h : z \mapsto M\sqrt{z} + \frac{Lz}{2}$ a broader class containing the limitations of both previous ones.*

In the following, we consider the class of h-RG^+ and convex functions. We then propose Algorithm 12, which consists in applying Algorithm 10 (with the update written as in Equation (6.6)) to the QG^+ convex function $h^{-1} \circ (f - f_\star)$. We obtain the following convergence rate.

Theorem 6.3.3. *Let f an $h - \mathrm{RG}^+$ convex function, and x_0 any starting point. Then Algorithm 12 verifies $f(x_n) - f_\star \leqslant h\left(\frac{d(x_0,\mathcal{X}_\star)^2}{n+1}\right)$. In the examples of Remark 6.3.2, this gives:*

1. *When h is the linear function $h : z \mapsto \frac{Lz}{2}$ (f is L-QG^+ convex), then $f(x_n) - f_\star \leqslant \frac{L}{2}\frac{d(x_0,\mathcal{X}_\star)^2}{n+1}$.*

Algorithm 12 Heavy-ball method for h-RG$^+$ convex functions

Input: x_0, h, $f_\star$

for $k = 1 \dots n$ **do**

Choose g_{k-1} from $\partial f(x_{k-1})$

$\gamma_{k-1} = \frac{1}{2(k+1)} \frac{1}{h' \circ h^{-1}(f(x_{k-1}) - f_\star)}$

$x_k \leftarrow x_{k-1} - \gamma_{k-1} g_{k-1} + \frac{k-1}{k+1}(x_{k-1} - x_{k-2})$

Output: x_n

Algorithm 13 Heavy-ball method for Lipschitz continuous convex functions

Input: x_0, M, $f_\star$

for $k = 1 \dots n$ **do**

Choose g_{k-1} from $\partial f(x_{k-1})$

$\gamma_{k-1} = \frac{1}{k+1} \frac{f(x_{k-1}) - f_\star}{M^2}$

$x_k \leftarrow x_{k-1} - \gamma_{k-1} g_{k-1} + \frac{k-1}{k+1}(x_{k-1} - x_{k-2})$

Output: x_n

2. *When h is the function $h : z \mapsto M\sqrt{z}$ (f is M-Lip. convex), then* $f(x_n) - f_\star \leqslant M \frac{d(x_0, \mathcal{X}_\star)}{\sqrt{n+1}}$.
3. *When h is the function $h : z \mapsto M\sqrt{z} + \frac{Lz}{2}$, then* $f(x_n) - f_\star \leqslant M \frac{d(x_0, \mathcal{X}_\star)}{\sqrt{n+1}} + \frac{L}{2} \frac{d(x_0, \mathcal{X}_\star)^2}{n+1}$.

Sketch of proof. The proof consists in inverting h and applying one of the results on the QG$^+$ convex functions $h^{-1}(f - f_\star)$. For a detailed proof, see Appendix 6.F. ■

We also observe that for $h : z \mapsto \frac{Lz}{2}$, Algorithm 12 is exactly Algorithm 10 and we recover the worst-case guarantee provided in Theorem 6.3.3. Similarly, when considering $h : z \mapsto M\sqrt{z}$, Algorithm 12 is written as Algorithm 13 and we recover the worst-case guarantee provided in (Drori and Taylor, 2020, Cor. 3). However, in this case the algorithm is different: the latest requires the knowledge of $d(x_0, \mathcal{X}_\star)$, while Algorithm 13 requires the knowledge of $f_\star$. In this sense, the two proposed methods are complementary.

Finally, we emphasize the fact that in practice, a lot of machine learning models requires to minimize non-smooth functions that are neither Lipschitz continuous nor QG$^+$ (e.g. least-square regressions with lasso penalization of TV-L$_2$ model widely used in computer vision). These functions can be tackled using the flexible function $h(z) = M\sqrt{z} + \frac{L}{2}z$ Then, applying Theorem 6.3.3 leads to the guarantee given in the third point of Theorem 6.3.3 obtained with Algorithm 12 where $\gamma_{k-1} \leftarrow \frac{1}{k+1} \frac{1}{L} \left[1 - 1/\sqrt{1 + \frac{2L}{M^2}(f(x_{k-1}) - f_\star)}\right]$.

Extension with additional constraint. In Section 6.G, we provide geometric convergence guarantees when the functions are also assumed satisfy a relaxation of strong convexity, QG$^-$.

List of potential applications. The QG$^+$ class, and its extension to $h -$ RG$^+$ offer a flexibility that allows to tackle several non-smooth machine learning problems, including for example RELU activation, L_1 or TV regularization. As an example, the classical TV-L_2 denoising problem, which combines a term with quadratic growth with a non-smooth (even at the optimum) term, is neither Lipschitz nor smooth, but belongs to our class $h -$ RG$^+$, with $h : z \mapsto M\sqrt{z} + \frac{Lz}{2}$ (Remark 6.3.2-3).

Limitations. As specific methods have often been designed for those applications, our approach does not bring a systematic improvement. Yet, we believe it paves the way for adaptive methods that could do so. Remark that our Algorithm 11 is adaptive and optimal for both the class of QG$^+(L)$ functions, and the class of M-Lipschitz continuous ones. Extending our analysis to obtain an adaptive algorithm for $h -$ RG$^+$, which is left as an open direction, would allow to efficiently tackle TV-L^2 type of problems in a parameter-free way.

Conclusion. In this paper, we thoroughly analyze the class of convex QG$^+$ functions. This function class relaxes the smoothness assumption and is motivated by the fact that QG$^+$ satisfies *condition continuity*, a desirable property for analyzing first-order methods.

We analyze several such methods, and provide tight worst-case guarantees for them. Three methods achieve the optimal convergence rate over the class. Our analysis is supported by the derivation of *interpolation conditions* allowing to verify all the results numerically. In particular, we observe that a heavy-ball algorithm results in acceleration (w.r.t. the subgradient method), attaining the lower complexity bound for this class, a surprising result with respect to the smooth case. Moreover, using line-search, we obtain a parameter-free algorithm which is adaptive to the class parameter in QG^+, and to the function class, as it also achieves the optimal rate for Lipschitz functions, a strongly desirable property in practice. Finally, we leverage our results to obtain convergence bounds for more complex classes of functions, combining the difficulties of the QG^+ and the Lipschitz classes. Overall, this work participates to the trend of questioning the relevance of the most classical assumptions used in the analysis of first-order optimization methods. While our results are not intended to provide a definitive answer to this question, it goes one step further by providing an in-depth analysis for a more stable class of functions. Providing a similar analysis while relaxing convexity is a major open challenge.

Organisation of the appendix

This appendix contains the proofs of the theorems stated in the main core of the paper. We also state a conjecture and bring some evidence about its statement. This appendix also contains discussions and extended results.

Appendix 6.A details the results on the subgradient method. Appendix 6.A.1 contains the proof of Theorem 6.2.1, Appendix 6.A.2 contains the proof of Theorem 6.2.2 and Appendix 6.A.3 contains a conjecture that does not appear in the main core of the paper. This appendix also contains some evidence supporting this conjecture.

Appendix 6.B contains the proofs for lower bounds on the class QG^+ convex. Appendix 6.B.1 contains the proofs of Theorem 6.2.3 stating the lower bound under the classical assumption that the difference between the iterates lies into the span of observed gradients. Appendix 6.B.2 extends the latter results without the aforementioned assumption.

Appendix 6.C contains the proof of Theorem 6.2.4, the main result of the paper, stating that all first-order algorithm verifying a given identity, also enjoys an upper bound guarantee.

Appendix 6.D contains the proofs of all the claims that figure in Table 6.2 that are not already made elsewhere in this work or in others.

Appendix 6.E contains the proof of Theorem 6.2.6, essential to use the PEP framework.

Appendix 6.F contains all the proofs and discussions related to the extended class of the RG^+ convex functions

Finally, Appendix 6.G contains linear convergence result under an additional assumption similar to the classical quadratic growth assumption. This result is not presented in the main core in the paper, since it is a bit out of the scope of the main message. However, we thought it was worth mentioning it here.

6.A (Sub)gradient method on QG^+-convex functions

In this appendix, we provide the proof of Theorems 6.2.1 and 6.2.2 stating respectively an upper bound result on the subgradient method with fixed step-size $1/L$ on the Polyak-Rupert averaged iterate, and a lower bound result on the subgradient method on the last iterate. Finally, based on this lower bound, we suggest a specific tuning of the subgradient method for QG^+ convex functions. A conjecture is formulated on the worst-case bound achieved by this method with the prescribed tuning, as well as evidence obtained through the PEP framework.

6.A.1 Convergence of subgradient method with fixed step-size at Polyak-Rupert averaged iterate

In section 6.2.1, we state the following theorem about a worst-case upper bound of Algorithm 9 on the class of QG^+-convex functions. In this section, we provide the proof of this theorem.

Theorem 6.2.1. (Convergence of Algorithm 9 in average) *Let f be an L-QG^+ convex function. Applying (sub)gradient method on f with step-size $\gamma \triangleq \frac{1}{L}$ leads to the following*

guarantee:

$$\frac{1}{n+1}\sum_{k=0}^{n}(f(x_k)-f_\star) \leqslant \frac{L}{2}\frac{1}{n+1}d(x_0,\mathcal{X}_\star)^2. \tag{6.2}$$

Proof. Let $k \in [0,n]$. We have

$$\begin{aligned}
d(x_{k+1},\mathcal{X}_\star)^2 &= \|x_{k+1}-\pi_{\mathcal{X}_\star}(x_{k+1})\|^2 \\
&\leqslant \|x_{k+1}-\pi_{\mathcal{X}_\star}(x_k)\|^2 \\
&= \|x_k-\gamma g_k-\pi_{\mathcal{X}_\star}(x_k)\|^2, \qquad \text{with } g_k\in\partial f(x_k) \\
&= \|x_k-\pi_{\mathcal{X}_\star}(x_k)\|^2 - 2\gamma\langle x_k-\pi_{\mathcal{X}_\star}(x_k), g_k\rangle + \gamma^2\|g_k\|^2 \\
&\overset{Eq.\ (6.9)}{\leqslant} \|x_k-\pi_{\mathcal{X}_\star}(x_k)\|^2 - 2\gamma\left(f(x_k)-f_\star+\frac{1}{2L}\|g_k\|^2\right)+\gamma^2\|g_k\|^2 \\
&= d(x_k,\mathcal{X}_\star)^2 - 2\gamma\left(f(x_k)-f_\star\right) - \gamma\left(\frac{1}{L}-\gamma\right)\|g_k\|^2 \\
&\overset{\gamma=\frac{1}{L}}{=} d(x_k,\mathcal{X}_\star)^2 - \frac{2}{L}\left(f(x_k)-f_\star\right)
\end{aligned}$$

By reordering the terms and summing over k:

$$\sum_{k=0}^{n}(f(x_k)-f_\star) \leqslant \frac{L}{2}d(x_0,\mathcal{X}_\star)^2 \tag{6.10}$$

which leads to the desired results.

■

Remark 6.A.1. *From Theorem 6.2.1, we conclude*

$$\begin{aligned}
\min_{0\leqslant k\leqslant n} f(x_k)-f_\star &\leqslant \frac{L}{2}\frac{1}{n+1}d(x_0,\mathcal{X}_\star)^2 \\
f\left(\frac{1}{n+1}\sum_{k=0}^{n}x_k\right)-f_\star &\overset{\text{(by convexity)}}{\leqslant} \frac{L}{2}\frac{1}{n+1}d(x_0,\mathcal{X}_\star)^2.
\end{aligned}$$

Remark 6.A.2. *Note that this bound is tight not only for QG^+ convex functions, but also for smooth convex functions.*

Indeed, we consider the real Huber function defined as

$$f(x) = \begin{cases} \frac{L}{2}x^2 & \text{if } |x|\leqslant 1 \\ L|x|-\frac{L}{2} & \text{if } |x|\geqslant 1 \end{cases} \tag{6.11}$$

This function is L-smooth convex and often used to find lower bounds (See e.g. Drori and Teboulle, 2014; Taylor et al., 2017c; Kim and Fessler, 2016). Moreover, starting from $x_0 = n+1$, GD with $\gamma=\frac{1}{L}$ leads exactly to $x_k = n+1-k$ for all $k\leqslant n$, hence $f(x_k)-f_\star = L\left(n+\frac{1}{2}-k\right)$ and $\sum_{k=0}^{n}(f(x_k)-f_\star) = L\sum_{k=0}^{n} n+\frac{1}{2}-k = \frac{L}{2}(n+1)^2 = \frac{L}{2}d(x_0,\mathcal{X}_\star)^2$.

6.A.2 Convergence limitation of the subgradient method in last iterate

In this section, we prove Theorem 6.2.2 stating a lower bound guarantee on the convergence of the subgradient method. This proof is by far the most technical of this paper due to the amount of newly introduced notations.

Theorem 6.2.2. (Lower bound for Algorithm 9 - final iterate). *Let $n \in \mathbb{N}$. For any sequence $(\gamma_i)_{0\leqslant i\leqslant n-1} > 0$ and any $\varepsilon > 0$, there exists an L-QG^+ convex function f that verifies, after n iterations of Algorithm 9 with step-sizes $(\gamma_i)_{0\leqslant i\leqslant n-1}$,*

$$f(x_n) - f_\star \geqslant \frac{L}{2} L\gamma_{n-1} d(x_0, \mathcal{X}_\star)^2 - \varepsilon. \tag{6.4}$$

Proof. Let $\eta > 0$.

We introduce the following notations:

- $\delta \triangleq \left(\frac{\eta\sqrt{3}}{1+L\gamma_{n-2}}\right)^{1/2}$
- Huber function
$$h_\delta(x) = \begin{cases} \frac{L}{2}x^2 & \text{if } x \leqslant \delta \\ L\delta x - \frac{L}{2}\delta^2 & \text{if } x > \delta \end{cases} \tag{6.12}$$
- For $i \in [|0, n-1|]$, define $\xi_i \triangleq \delta\left(1 + \sum_{k=i}^{n-2} L\gamma_i\right)$.
- $\lambda = \frac{L\eta}{(1+L\gamma_{n-2})(1+\eta^2+\xi_0^2)}$.

Based on those notations, we define the 3-dimensional function

$$f(x) = \max\left[\frac{L}{2}\left(x^{(1)} - 1 + |x^{(2)}|\sqrt{3}\right), h_\delta\left(x^{(3)}\right), \frac{\lambda}{2}\|x\|_2^2\right]. \tag{6.13}$$

f is convex as maximum of 3 convex functions.

Moreover, we note that $\mathcal{X}_\star = \{0\}$ and $f_\star = 0$.

And each of the three components defining f is smaller that $\frac{L}{2}\|x\|_2^2$. Indeed,

$$\begin{aligned}
\frac{L}{2}\left(x^{(1)} - 1 + |x^{(2)}|\sqrt{3}\right) &= \frac{L}{2}\left(\left(x^{(1)}\right)^2 - \left(x^{(1)} - \frac{1}{2}\right)^2 + \left(x^{(2)}\right)^2 - \left(x^{(2)} - \frac{\sqrt{3}}{2}\right)^2\right) \\
&\leqslant \frac{L}{2}\left(\left(x^{(1)}\right)^2 + \left(x^{(2)}\right)^2\right) \\
&\leqslant \frac{L}{2}\|x\|_2^2 \qquad (6.14) \\
h_\delta\left(x^{(3)}\right) &\leqslant \frac{L}{2}(x^{(3)})^2 \\
&\leqslant \frac{L}{2}\|x\|_2^2 \qquad (6.15) \\
\lambda \leqslant \frac{L\eta}{2\eta} = \frac{L}{2} \leqslant L, \text{ hence } \frac{\lambda}{2}\|x\|_2^2 &\leqslant \frac{L}{2}\|x\|_2^2 \qquad (6.16)
\end{aligned}$$

Therefore, f is also $\mathrm{QG}^+(L)$.

We choose to start the GD algorithm at $x_0 \triangleq \begin{pmatrix}1 & \eta & \xi_0\end{pmatrix}^\top$.

We claim that after i ($0 \leqslant i \leqslant n-1$) steps of GD, $x_i = \begin{pmatrix}1 & \eta & \xi_i\end{pmatrix}^\top$.

This can be proven by induction. Indeed, by definition, this is true for $i = 0$. We now assume this property is true for some $i < n-1$ and want to prove it for $i+1$.

From the 3 remarks

$$h_\delta(\xi_i) \geqslant \frac{L}{2}\left(1 - 1 + |\eta|\sqrt{3}\right) \tag{6.17}$$

$$h_\delta(\xi_i) \geqslant \frac{\lambda}{2}\|x_i\|_2^2 \tag{6.18}$$

$$\xi_i \geqslant \delta, \tag{6.19}$$

we conclude that $\nabla f(x_i) = \begin{pmatrix} 0 & 0 & L\delta \end{pmatrix}^\top$.
Hence $x_{i+1} = x_i - \begin{pmatrix} 0 & 0 & L\gamma_i\delta \end{pmatrix}^\top = \begin{pmatrix} 1 & \eta & \xi_{i+1} \end{pmatrix}^\top$.
Finally, from the 2 remarks

$$\frac{L}{2}(1 - 1 + |\eta|\sqrt{3}) \geqslant h_\delta(\xi_{n-1}) \tag{6.20}$$

$$\frac{L}{2}(1 - 1 + |\eta|\sqrt{3}) \geqslant \frac{\lambda}{2}\|x_{n-1}\|_2^2, \tag{6.21}$$

we conclude that $\nabla f(x_{n-1}) = \frac{L}{2}\begin{pmatrix} 1 & \sqrt{3} & 0 \end{pmatrix}^\top$, leading to

$$x_n = x_{n-1} - \gamma_{n-1}\frac{L}{2}\begin{pmatrix} 1 & \sqrt{3} & 0 \end{pmatrix}^\top = \begin{pmatrix} 1 - \frac{L\gamma_{n-1}}{2} & \eta - \frac{L\gamma_{n-1}\sqrt{3}}{2} & \delta \end{pmatrix}^\top.$$

We compute the two quantities

$$\|x_0\|^2 = 1 + \eta^2 + \xi_0^2 \tag{6.22}$$

$$\begin{aligned} f(x_n) &\geqslant \frac{L}{2}\left(1 - \frac{L\gamma_{n-1}}{2} - 1 + |\eta - \frac{L\gamma_{n-1}\sqrt{3}}{2}|\sqrt{3}\right) \\ &= \frac{L}{2}\left(-\frac{L\gamma_{n-1}}{2} + \left(\frac{L\gamma_{n-1}\sqrt{3}}{2} - \eta\right)\sqrt{3}\right) \\ &= \frac{L}{2}\left(L\gamma_{n-1} - \eta\sqrt{3}\right) \end{aligned} \tag{6.23}$$

Finally,

$$\begin{aligned} \frac{f(x_n) - f_\star}{d(x_0, \mathcal{X}_\star)^2} &\geqslant \frac{L}{2}\frac{L\gamma_{n-1} - \eta\sqrt{3}}{1 + \eta^2 + \xi_0^2} \\ &= \frac{L}{2}\frac{L\gamma_{n-1} - \eta\sqrt{3}}{1 + \eta^2 + \delta^2\left(1 + \sum_{k=i}^{n-2} L\gamma_i\right)^2} \\ &= \frac{L}{2}\frac{L\gamma_{n-1} - \eta\sqrt{3}}{1 + \eta^2 + \frac{\eta\sqrt{3}}{1+L\gamma_{n-2}}\left(1 + \sum_{k=i}^{n-2} L\gamma_i\right)^2} \\ &\xrightarrow[\eta\to 0]{} \frac{L}{2}L\gamma_{n-1}. \end{aligned} \tag{6.24}$$

Hence, for any $\varepsilon > 0$, we can find $\eta > 0$ sufficiently small such that f reaches the claim of the Theorem.

■

6.A.3 A new tuning prescription

Theorem 6.2.2 provides a lower bound on the last iterate value of the subgradient method on the class of QG^+ convex functions. Moreover, a new analysis of the subgradient method on the Huber function (6.12), starting at $x_0 = 1 + 2\sum_{k=0}^{n-1} L\gamma_k$ provides another lower bound.

Combining those 2 results, we know that whatever $(\gamma_i)_{0\leqslant i\leqslant n-1} > 0$ is, there exists f an $L - \mathrm{QG}^+$ convex function as well as a starting point x_0 such that

$$f(x_n) - f_\star \geqslant \max\left(\frac{L}{2}\frac{1}{1 + 2\sum_{k=0}^{n-1} L\gamma_k}, \frac{L}{2}L\gamma_{n-1}\right) d(x_0, \mathcal{X}_\star)^2. \tag{6.25}$$

Naturally, we propose the sequence of $(\gamma_i)_{0\leqslant i\leqslant n-1} > 0$ that verifies for all n, $\frac{L}{2}\frac{1}{1+2\sum_{k=0}^{n-1}L\gamma_k} = \frac{L}{2}L\gamma_{n-1}$ (for each index $n \geqslant 1$). This is summarized in Algorithm 14. We note that $u_n \sim 2\sqrt{n}$ and $\gamma_{n-1} \sim \frac{1}{2L\sqrt{n}}$. The lower bound (6.25) for this method becomes $f(x_n) - f_\star \geqslant \frac{L}{2}L\gamma_{n-1}d(x_0,\mathcal{X}_\star)^2 \sim \frac{L}{4\sqrt{n}}d(x_0,\mathcal{X}_\star)^2$. We conjecture that this bound is actually reached by the proposed method 14.

Algorithm 14 GD with decreasing step-sizes

Input: x_0, L

$u_0 = 1$;

for k=*1* $\dots$ n **do**

  $u_k \leftarrow \frac{u_{k-1}}{2} + \sqrt{\left(\frac{u_{k-1}}{2}\right)^2 + 2}$;

  $\gamma_{k-1} \leftarrow \frac{1}{Lu_k}$;

  Pick $g_{k-1} \in \partial f(x_{k-1})$;

  $x_k \leftarrow x_{k-1} - \gamma_{k-1}g_{k-1}$

Output: x_n

Conjecture 6.A.3. (Convergence of GD with decreasing step-sizes) *The algorithm 14 verifies the following bound on every $L - \mathrm{QG}^+$ convex function f:*

$$f(x_n) - f_\star \leqslant \frac{L}{2u_n}d(x_0,\mathcal{X}_\star)^2. \tag{6.26}$$

where u_n is the sequence used in 14, defined by

$$u_0 = 1 \tag{6.27}$$

$$u_k = \frac{u_{k-1}}{2} + \sqrt{\left(\frac{u_{k-1}}{2}\right)^2 + 2}, \quad \textit{for every } k \in [\![1,n]\!]. \tag{6.28}$$

and verifying

$$u_n \sim 2\sqrt{n}. \tag{6.29}$$

This conjecture is supported by the Figure 6.1 that has been built using the PEP framework. This figure represents the worst-case guarantee of Algorithm 14 as a function of the number of iterations. The conjecture (red curve) follows exactly the numerical worst-case guarantee provided by the PEPs (blue curve) and the equivalent sequence (green curve) is very close to the 2 previous ones.

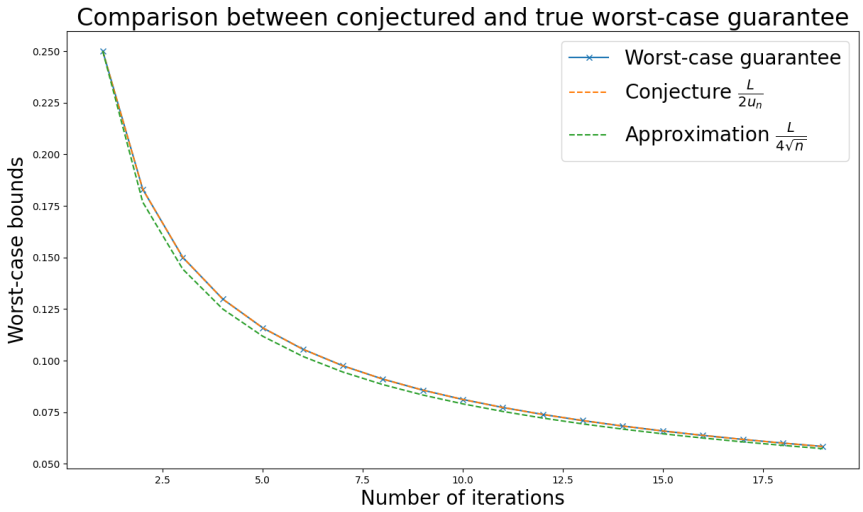


Figure 6.1: Verification of Conjecture 6.A.3 using PEPs

6.B First-order lower bound

In this section, we prove the lower bound of first-order methods on QG^+ convex functions.

In the first subsection, we assume that the iterates of the first-order algorithm must stay in the span of the past observed gradients.

In the second subsection, we release this assumption and still prove the same lower bound. This proof is a bit more technical, hence the reason why we provide the two proofs.

6.B.1 Proof of Theorem 2.3

The following theorem brings a lower bound over all first-order methods verifying that all the iterates lie into the span of the previously observed gradients.

Theorem 6.2.3. (Lower complexity bound) *Let $n \in \mathbb{N}$. There exist some $d \in \mathbb{N}$ and some convex L-QG^+ function f of input space $\mathbb{R}^d$ such that: for any sequence $(x_k)_{0\leqslant k\leqslant n}$ satisfying $x_k - x_0 \in$ span $\{g_0, g_1, g_2, \dots, g_{k-1}\}$ for all $k \leqslant n$ with $g_i \in \partial f(x_i)$ $(i = 0, \dots, k-1)$, we have:*

$$f(x_n) - f_\star \geqslant \frac{L}{2}\frac{1}{n+1}d(x_0, \mathcal{X}_\star)^2.$$

Proof. Consider $f(x) \triangleq \frac{L}{2}\|x\|_\infty^2$ defined on $\mathbb{R}^{n+1}$ and $x_0 = \vec{1}$.

After k steps, the oracle can "choose" to return a vector that lies in the first $k+1$ dimension of the input space, leading to $f(x_n) - f_\star \geqslant f(x_0) - f_\star = \frac{L}{2} = \frac{L}{2}\frac{1}{n+1}d(x_0, \mathcal{X}_\star)^2$. ■

Remark 6.B.1. *Note that by considering instead $x_0 = \begin{pmatrix}1 & 1-\varepsilon/n & 1-2\varepsilon/n & \dots & 1-\varepsilon\end{pmatrix}^\top$, one ends up with $f(x_n) - f_\star \geqslant \frac{L}{2}\frac{1-\varepsilon}{n+1}d(x_0, \mathcal{X}_\star)^2$ **whatever** the oracle "chooses" to return.*

6.B.2 Lower bound proof without span assumption

In this section we release the span assumption and prove that the previously shown lower bound still holds. This proof is a bit more technical than the one of Theorem 6.2.3 proven in Appendix 6.B.1.

Theorem 6.B.2 (Lower bound of first-order algorithm without span assumption)**.** *Let $\mathcal{A}$ a first-order algorithm. Then, for any $n \geqslant 0$, there exists d a positive integer, f a $L-\mathrm{QG}^+$ convex function of input space $\mathbb{R}^d$ and a starting point x_0 such that $f(x_n) - f_\star \geqslant \frac{L}{2}\frac{1}{n+1}d(x_0, \mathcal{X}_\star)^2$.*

Proof. Let $\mathcal{E} = \{v \in \mathbb{R}^{n+1} | \forall i \in [\![1, n+1]\!], |v_i| = 1\}$ the set of the 2^{n+1} vectors of $\mathbb{R}^{n+1}$ which all coordinates are ± 1.

For each $v \in \mathcal{E}$, we introduce $f_v(x) \triangleq \frac{L}{2}\|x - v\|_\infty^2$ defined on $\mathbb{R}^{n+1}$.

First note that all those functions are $L - \mathrm{QG}^+$ convex. We will prove that not only there exists a starting point x_0 and a $L - \mathrm{QG}^+$ convex function such that $f(x_n) - f_\star \geqslant \frac{L}{2}\frac{1}{n+1}d(x_0, \mathcal{X}_\star)^2$, but also that there exists a starting point x_0 and a function among the f_v we introduced above such that the latest holds.

To proceed, we need to show that the algorithm $\mathcal{A}$ cannot know the right v after only n iterations and therefore, cannot guarantee $f(x_n) - f_\star \leqslant \frac{L}{2}$. Taking $x_0 = \vec{0}$, $\|x_0 - x_\star\|^2 = n+1$ whatever $x_\star$ is (among $\mathcal{E}$), hence the result.

In order to prove that the algorithm cannot know the solution after n iterations, we keep track of all the remaining possibilities across time.

We denote by $\mathcal{E}_k$ the remaining possibilities after k steps of the algorithm. In particular, $\mathcal{E}_0 = \mathcal{E}$.

At each step k, $\mathcal{A}$ guesses x_k based on all the previous information, summarized in $\mathcal{E}_k$, and the oracle provides $f(x_k)$ and a subgradient $g_k \in \partial f(x_k)$.

Let $v_k \in \arg\max_{v \in \mathcal{E}_k} \|x_k - v_k\|_\infty$. We consider the case where the oracle behaves like if the objective function to minimize was f_{v_k}. Moreover, in the case where f_{v_k} is not differentiable in x_k, we ask that the oracle returns a subgradient co-linear to a vector belonging to the canonical basis (which is always possible).

Considering i such that g_k is co-linear to e_i, we obtain $\mathcal{E}_{k+1} = \{v \in \mathcal{E}_k | \langle v, e_i \rangle = \langle v_k, e_i \rangle\}$, reducing by half the number of remaining elements at each step (except when the algorithm badly guesses and receive twice the same direction, in which case one of the steps is useless).

After n steps, $\mathcal{E}_n$ contains 2 elements, and the algorithm $\mathcal{A}$ must guess based on nothing. Again, we consider the further one to the last guess as the right solution, and obtain the lower bound provided by the Theorem.

■

6.C Main result: worst-case guarantee of proposed methods

In this section, we prove Theorem 6.2.4, the main result of this paper, stating that all the sequences of iterates verifying a certain property enjoy an upper bound guarantee corresponding to the lower bound presented in Theorem 6.2.3 and proved in Section 6.B. Then, we prove Corollary 6.2.5 which provides 2 algorithms verifying the assumptions of Theorem 6.2.4.

Theorem 6.2.4. (Main result: sufficient condition for being worst-case optimal). *Let $\mathcal{A}$ be an iterative first-order method that verifies, for all convex $\mathrm{QG}^+(L)$ function f, and starting points x_0,*

$$\left\langle g_k, x_k - \left[\frac{k}{k+1}x_{k-1} + \frac{1}{k+1}x_0 - \frac{1}{k+1}\sum_{i=0}^{k-1}\frac{1}{L}g_i\right]\right\rangle \leqslant 0. \tag{6.7}$$

for some sequence $(g_i)_{i\in\mathbb{N}}$ of subgradients $g_i \in \partial f(x_i)$, and where $(x_i)_{i\in\mathbb{N}}$ are the iterates of $\mathcal{A}$. Then, the output x_n of $\mathcal{A}$ achieves the worst-case guarantee:

$$f(x_n) - f_\star \leqslant \frac{L}{2}\frac{1}{n+1}d(x_0, \mathcal{X}_\star)^2.$$

Proof. This proof relies on the Lyapunov function

$$V_n \triangleq n(f(x_{n-1}) - f_\star) + \frac{L}{2}\left\|x_0 - \pi_{\mathcal{X}_\star}(x_0) - \sum_{i=0}^{n-1}\frac{1}{L}g_i\right\|^2. \tag{6.30}$$

For all k, we verify

$$
\begin{aligned}
V_{k+1} - V_k &= \left[(k+1)(f(x_k) - f_\star) + \frac{L}{2}\left\|x_0 - \pi_{\mathcal{X}_\star}(x_0) - \sum_{i=0}^{k}\frac{1}{L}g_i\right\|^2\right] \\
&\quad - \left[k(f(x_{k-1}) - f_\star) + \frac{L}{2}\left\|x_0 - \pi_{\mathcal{X}_\star}(x_0) - \sum_{i=0}^{k-1}\frac{1}{L}g_i\right\|^2\right] \\
&= (f(x_k) - f_\star) + k(f(x_k) - f(x_{k-1})) \\
&\quad + \frac{L}{2}\left[-\frac{2}{L}\left\langle g_k, x_0 - \pi_{\mathcal{X}_\star}(x_0) - \sum_{i=0}^{k-1}\frac{1}{L}g_i\right\rangle + \frac{1}{L^2}\|g_k\|^2\right] \\
&= \left(f(x_k) - f_\star + \frac{1}{2L}\|g_k\|^2\right) + k(f(x_k) - f(x_{k-1})) \\
&\quad - \left\langle g_k, x_0 - \pi_{\mathcal{X}_\star}(x_0) - \sum_{i=0}^{k-1}\frac{1}{L}g_i\right\rangle \\
&\overset{(6.8,\ 6.9)}{\leqslant} \langle g_k, x_k - \pi_{\mathcal{X}_\star}(x_0)\rangle + k\langle g_k, x_k - x_{k-1}\rangle - \left\langle g_k, x_0 - \pi_{\mathcal{X}_\star}(x_0) - \sum_{i=0}^{k-1}\frac{1}{L}g_i\right\rangle \\
&= \left\langle g_k, x_k - \pi_{\mathcal{X}_\star}(x_0) + k(x_k - x_{k-1}) - \left(x_0 - \pi_{\mathcal{X}_\star}(x_0) - \sum_{i=0}^{k-1}\frac{1}{L}g_i\right)\right\rangle \\
&= \left\langle g_k, (k+1)x_k - kx_{k-1} - x_0 + \sum_{i=0}^{k-1}\frac{1}{L}g_i\right\rangle
\end{aligned}
$$

The assumption therefore concludes $\forall k, V_{k+1} \leqslant V_k$. Finally,

$$
(N+1)(f(x_N) - f_\star) + \frac{L}{2}\left\|x_0 - \pi_{\mathcal{X}_\star}(x_0) - \sum_{i=0}^{N}\frac{1}{L}g_i\right\|^2 = V_{n+1} \leqslant V_0 = \frac{L}{2}\|x_0 - x^*\|^2.
$$

In particular, $f(x_N) - f_\star \leqslant \frac{L}{2}\frac{1}{N+1}\|x_0 - x^*\|^2$.

■

Corollary 6.2.5. *Let $n \in \mathbb{N}$, $d \in \mathbb{N}$, f be a convex QG^+ function, and $x_0 \in \mathbb{R}^d$. Also, let x_n be the output of either Algorithm 10 or Algorithm 11, we have: $f(x_n) - f_\star \leqslant \frac{L}{2}\frac{1}{n+1}d(x_0, \mathcal{X}_\star)^2$.*

Proof.

Algorithm 10: First note that the update of Algorithm 10 cancels the RHS of the inner product in Equation (6.7), therefore verifying the assumption of Theorem 6.2.4 achieving the optimal convergence guarantee on the class of convex QG^+ functions.

Algorithm 11: Let's first prove that this algorithm is well-defined. By the assumption that $\mathcal{X}_\star = \left\{x \in \mathbb{R}^d \mid f(x) = f_\star = \min_{y\in\mathbb{R}^d} f(y)\right\}$ is compact, and not empty, we can consider $x_\star$ any of its element, and $R > 0$ sufficiently large so that the open ball $B(x_\star, R)$ centered in $x_\star$ and of radius R contains $\mathcal{X}_\star$. With those notations, any point x on the sphere $S(x_\star, R)$ does not belong to $\mathcal{X}_\star$, therefore verifies $f(x) > f_\star$. By continuity of convex functions and compactness of the sphere, f reaches a minimum $m > f_\star$ on the sphere $S(x_\star, R)$. Given this, for any x such that $\|x - x_\star\| > R$, we have $f(x) - f_\star \geqslant \frac{\|x-x_\star\|}{R}\left(f\left(x_\star + \frac{R}{\|x-x_\star\|}(x - x_\star)\right) - f_\star\right) \geqslant \frac{\|x-x_\star\|}{R}m$. This shows in particular that f is coercive, i.e. $\lim_{\|x\|\to\infty} f(x) = \infty$.
Therefore, each line-search step is well-defined as f necessarily admits a minimum on

every affine line. From the existence of such a minimum, it is well known that their exists a subgradient to this optimum that is orthogonal to the search direction and this is what Algorithm 11 uses.
Recall that the line-search returns the updates $x_k = \frac{k}{k+1}x_{k-1} + \frac{1}{k+1}x_0 - \frac{\alpha_\star}{k+1}\sum_{i=0}^{k-1} g_i$. where $\alpha_\star$ is the optimal parameter of the line-search. Therefore, the RHS of the inner product in Equation (6.7) can be written $(\alpha_\star - \frac{1}{L})\frac{1}{k+1}\sum_{i=0}^{k-1} g_i$ and g_k is chosen by the algorithm to be orthogonal to $\frac{1}{k+1}\sum_{i=0}^{k-1} g_i$, therefore canceling the inner product and verifying the assumption and conclusion of Theorem 6.2.4. ■

6.D Summary of convergence results on QG^+ convex and Lipschitz convex

In this section, we state and prove the 2 results of Table 6.2 that are not already proven elsewhere.

Table 6.2: Optimality of the proposed methods over the set of QG^+ convex functions and M-Lipschitz convex functions. ELS: Exact Line-Search. ✓ indicates optimality among the class and × the contrary. All counter examples are given in App. 6.D. †: constants resulting in optimal convergence rates depend on the class, thus for example Heavy-ball with step-size $\frac{\text{constant}}{(t+2)}$ is not adaptive as it does not achieve the optimal rate for both classes with the same constant. ‡: up to a $\log$ factor.

Method			Function class				Parameter free
Algorithm	Step-sizes $(\gamma_t)_{0\leqslant t\leqslant n-1}$	Iterate	$\mathrm{QG}^+(L)$ convex		M-Lipschitz convex		
Subgradient (Alg. 9)	constant†	Average	✓	(Thm. 6.2.1)	×	(Thm. 6.D.1)	×
Subgradient (Alg. 14)	constant$^\dagger/\sqrt{t}$	Average	×	(6.5)	✓‡	(Nesterov, 2003, Sec. 3.2.3)	×
Subgradient (Alg. 15)	ELS	Average	×	(Thm. 6.D.2)	×	(Thm. 6.D.2)	✓
Subgradient (Alg. 15)	ELS	Last	×	(Thm. 6.D.2)	×	(Thm. 6.D.2)	✓
Heavy-ball (Alg. 10)	constant$^\dagger/(t+2)$	Last	✓	(Cor. 6.2.5)	✓	(Drori and Taylor, 2020, , Cor. 3)	×
Heavy-ball (Alg. 11)	ELS	Last	✓	(Cor. 6.2.5)	✓	(Drori and Taylor, 2020, , Cor. 4)	✓

We first state Theorem 6.D.1.

Theorem 6.D.1. *For any $M > 0$ and any $\gamma > 0$, the subgradient method 9 with constant step-size γ cannot be guaranteed to converge to optimum on all the M-Lipschitz continuous convex functions, both in last iterate and in Polyak-Rupert averaged iterate.*

Proof. First we note that $f \triangleq z \mapsto M|z|$ is M-Lipschitz continuous and convex. Let $\gamma > 0$. We consider $x_0 = \frac{3}{4}M\gamma$ the starting point of the subgradient method with constant step-size γ. We verify that $x_1 = -\frac{1}{4}M\gamma$ and that the sequence $(x_t)_t$ cycles back to $\frac{3}{4}M\gamma$. Therefore, the sequence itself does not converge and the sequence of the PR averaged iterates converges to $\frac{1}{4}M\gamma$, while the optimum value would be 0. ■

Then, one could wonder whether performing exact line search steps on the subgradient method (Algorithm 15.) leads to convergence on Lipschitz continuous convex or QG^+ convex function. We now prove Theorem 6.D.2 stating that the subgradient method

with exact line search does not converge for all functions neither of the class of Lipschitz continuous convex functions nor of the class of QG^+ convex functions, neither in last iterate nor in Polyak-Rupert averaged iterate.

Theorem 6.D.2. *There exists a QG^+ convex function f_{QG} and a Lipschitz continuous convex function f_{Lip} such that, the iterates $((x_n)_n^{f_{QG}})_n$ and $((x_n)_n^{f_{Lip}})_n$ obtained by Algorithm 15, verify the 2 guarantees*

$$f(x_n) - f_\star \geqslant \frac{L}{6}\|x_0 - x_\star\|^2. \tag{6.31}$$

$$f(\bar{x_n}) - f_\star \geqslant \frac{L}{6}\|x_0 - x_\star\|^2. \tag{6.32}$$

where $\bar{x_n}$ denotes the Polyak-Rupert averaged iterate obtained from the sequence $(x_n)_n$.

Proof. Considering $f_{Lip}(z) = M\|x\|_\infty$ and $f_{QG}(z) = \frac{L}{2}\|x\|_\infty^2$ defined on $\mathbb{R}^3$, the iterates of Algorithm 15 can be cycling between the four points $(1,1,1)$, $(1,-1,1)$, $(-1,1,1)$ and $(-1,-1,1)$. Therefore, the aforementioned statement. ■

Algorithm 15 Subgradient method with line-search

Input: x_0, $v_0 \leftarrow 0$

for $k = 1 \dots n$ **do**

  Pick $g_{k-1} \in \partial f(x_{k-1})$.

  $\alpha_k \leftarrow \arg\min_\alpha f\left(x_{k-1} - \alpha g_{k-1}\right)$

  $x_k \leftarrow x_{k-1} - \alpha_k g_{k-1}$

Output: x_n

6.E Interpolation results for QG^+ convex functions

The interpolation conditions of a given class represent the key ingredient to use the PEP framework on this class. Theorem 6.2.6 provides the interpolation conditions for the classes of QG^+ convex functions. In this section, we recall this result and prove it.

Theorem 6.2.6. (Interpolation conditions) *Let $(x_i, g_i, f_i)_{i\in I}$ a family of elements in $\mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}$. Set $I_\star$ the (assumed) non-empty subset of I of the indices of elements (x_i, g_i, f_i) verifying $g_i = 0$.*

Then, there exists a $\mathrm{QG}^+(L)$ and convex function f interpolating those points (i.e. such that $\forall i \in I, f(x_i) = f_i$ and $g_i \in \partial f(x_i)$) if and only if

$$\forall i \in I, \forall j \in I, \quad f_i \geqslant f_j + \langle g_j, x_i - x_j\rangle \tag{6.8}$$

$$\forall i \in I_\star, \forall j \in I, \quad f_i \geqslant f_j + \langle g_j, x_i - x_j\rangle + \frac{1}{2L}\|g_j\|^2. \tag{6.9}$$

Proof.

We prove the two implications one by one.

$\Rightarrow$: Assume there exists such a convex-QG^+ function f that interpolates $(x_i, g_i, f_i)_{i\in I}$. Equation (6.8) follows immediately from convexity. Let's prove equation (6.9). Let $i \in I_\star, \forall j \in I$ and $x \in \mathbb{R}^d$. We have:

$$f_j + \langle g_j, x - x_j\rangle \overset{\text{CVX}}{\leqslant} f(x) \overset{\mathrm{QG}^+}{\leqslant} \min_{z\in\mathbb{R}^d} f(z) + \frac{L}{2}d(x, \mathcal{X}_\star)^2 \leqslant f_i + \frac{L}{2}\|x - x_i\|^2.$$

Rewriting the previous equation for $x = x_i + \frac{1}{L}g_j$ leads to equation (6.9).

$\Leftarrow$: Let's consider equations (6.8) and (6.9) are verified. Applying (6.8) with $j \in I_\star$

$$\forall i \in I, \forall j \in I_\star, f_i \geqslant f_j \tag{6.33}$$

In particular, $\forall i \in I_\star, \forall j \in I_\star, f_i = f_j$. Hence, let's introduce $f_\star$ the common value of all the f_i for $i \in I_\star$. Let's denote here $\mathcal{X}^\star$ the convex hull of $\{x_i\}_{i\in I_\star}$. Finally let's introduce $\mu \triangleq 2\min_{i\in I\setminus I_\star}\left(\frac{f_i - f_\star}{d(x_i,\mathcal{X}^\star)^2}\right) \overset{\mathrm{QG}^+}{\leqslant} L$.

Let's prove that the following function f is a solution:

$$f(x) = \max\left(\max_{j\in I}\left(f_j + \langle g_j, x - x_j\rangle\right), f_\star + \frac{\mu}{2}d\left(x,\mathcal{X}^\star\right)^2\right). \tag{6.34}$$

- $\forall i \in I, f(x_i) = f_i$: For all $i \in I$, equation (6.8) shows $\max_{j\in I}\left(f_j + \langle g_j, x_i - x_j\rangle\right) \leqslant f_i$ and the definition of μ leads to $f_\star + \frac{\mu}{2}d\left(x_i,\mathcal{X}^\star\right)^2 \leqslant f_i$. Hence, for all $i \in I$, $f(x_i) \leqslant f_i$. Moreover, from equation (6.34), $f(x) \geqslant (f_i + \langle g_i, x - x_i\rangle)$, hence $f(x_i) \geqslant f_i$. Finally, we conclude $\forall i \in I, f(x_i) = f_i$.
- $\forall i \in I, g_i \in \partial f(x_i)$: $\forall i \in I, \forall x, f(x) \geqslant (f_i + \langle g_i, x - x_i\rangle)$, and from the previous point, we conclude $\forall i \in I, \forall x, f(x) \geqslant (f(x_i) + \langle g_i, x - x_i\rangle)$. Finally, $\forall i \in I, g_i \in \partial f(x_i)$.
- f is convex: f is defined as the maximum of convex functions, hence is convex.
- f is QG^+: We aim at proving that $\forall x \in \mathbb{R}^d, f(x) \leqslant f_\star + \frac{L}{2}d\left(x,\mathcal{X}^\star\right)^2$. Since it is clear that $\forall x \in \mathbb{R}^d, f_\star + \frac{\mu}{2}d\left(x,\mathcal{X}^\star\right)^2 \leqslant f_\star + \frac{L}{2}d\left(x,\mathcal{X}^\star\right)^2$, it remains to prove that $\forall x \in \mathbb{R}^d, \forall j \in I, f_j + \langle g_j, x - x_j\rangle \leqslant f_\star + \frac{L}{2}d\left(x,\mathcal{X}^\star\right)^2$. The latest is also equivalent to $\forall x \in \mathbb{R}^d, \forall j \in I, \forall x_\star \in \mathcal{X}^\star, f_j + \langle g_j, x - x_j\rangle \leqslant f_\star + \frac{L}{2}\|x - x_\star\|^2$. For j and $x_\star$ fixed, this expression is a quadratic form in x, optimized for $x = x_\star + \frac{1}{L}g_j$. Hence, we need to show $\forall j \in I, \forall x_\star \in \mathcal{X}^\star, f_j + \left\langle g_j, x_\star + \frac{1}{L}g_j - x_j\right\rangle \leqslant f_\star + \frac{L}{2}\left\|x_\star + \frac{1}{L}g_j - x_\star\right\|^2$, also rewritten $\forall x_\star \in \mathcal{X}^\star, \forall j \in I, f_\star \geqslant f_j + \langle g_j, x_\star - x_j\rangle + \frac{1}{2L}\|g_j\|^2$. Since $\mathcal{X}^\star$ is the convex hull of $\{x_i\}_{i\in I_\star}$, the latter is obtained by a linear combination (with non-negative weights) of equation 6.9 for different values of i.

■

6.F Convergence bound on other classes

In this section, we naturally extend the previous results to the class of $h - \mathrm{RG}^+$ (See. Definition 6.3.1) convex functions.

Theorem 6.3.3. *Let f an $h - \mathrm{RG}^+$ convex function, and x_0 any starting point. Then Algorithm 12 verifies $f(x_n) - f_\star \leqslant h\left(\frac{d(x_0,\mathcal{X}_\star)^2}{n+1}\right)$. In the examples of Remark 6.3.2, this gives:*

1. *When h is the linear function $h : z \mapsto \frac{Lz}{2}$ (f is L-QG^+ convex), then $f(x_n) - f_\star \leqslant \frac{L}{2}\frac{d(x_0,\mathcal{X}_\star)^2}{n+1}$.*
2. *When h is the function $h : z \mapsto M\sqrt{z}$ (f is M-Lip. convex), then $f(x_n) - f_\star \leqslant M\frac{d(x_0,\mathcal{X}_\star)}{\sqrt{n+1}}$.*
3. *When h is the function $h : z \mapsto M\sqrt{z} + \frac{Lz}{2}$, then $f(x_n) - f_\star \leqslant M\frac{d(x_0,\mathcal{X}_\star)}{\sqrt{n+1}} + \frac{L}{2}\frac{d(x_0,\mathcal{X}_\star)^2}{n+1}$.*

Proof.

First note that since h is strictly increasing, h^{-1} is well-defined. Furthermore, Definition 6.3.1 can be expressed as $h^{-1}\left(f(x) - f_\star\right) \leqslant d(x,\mathcal{X}_\star)^2$. Therefore, $h^{-1}\left(f - f_\star\right)$ is

$2-\mathrm{QG}^+$. And we know by definition that h is increasing and concave, then h^{-1} is increasing and convex. Since f is also convex, so is $h^{-1}(f-f_\star)$.

We conclude that $h^{-1}(f-f_\star)$ is $2-\mathrm{QG}^+$ and convex, and then we know that Algorithm 10 applied on $h^{-1}(f-f_\star)$ leads to

$$h^{-1}(f(x_n)-f_\star) \leqslant \frac{d(x_0,\mathcal{X}_\star)^2}{n+1}.$$

It remains to compose the above by h and to notice that Equation 6.6 applied on $h^{-1}(f-f_\star)$ is exactly Algorithm 12.

■

6.G Linear convergence guarantees under lower bound assumption

In all this section, we assume that f is convex and $h-\mathrm{RG}^+$ for a certain h. Moreover, we consider that f verifies the following additional assumption (referred to as "Łojasiewicz error bound inequality" in (Bolte et al., 2017)) for a given $\kappa \geqslant 1$:

Assumption 6.G.1. *For all $x \in \mathbb{R}^d$, $f(x)-f_\star \geqslant h\left(\frac{d(x,\mathcal{X}_\star)^2}{\kappa}\right)$.*

Remark 6.G.2. *When h is the linear function $h: R \mapsto \frac{LR}{2}$, then f is simply $L-\mathrm{QG}^+$ convex as well as $\mu-\mathrm{QG}^-$ $\left(\text{where } \mu \triangleq \frac{L}{\kappa}\right)$(See (Guille-Escuret et al., 2021) for the definition of QG^-).*

We introduce the Algorithm 16 based on the restart idea studied in (Nemirovskii and Nesterov, 1985; Nesterov, 2013; Iouditski and Nesterov, 2014).

Algorithm 16 Heavy-ball with restart

Input: x_0, h, $f_\star$

for $k = 1 \dots n$ **do**

  Choose g_{k-1} from $\partial f(x_{k-1})$

  $l \leftarrow k \bmod \lfloor \kappa e \rfloor - 1$ (between 1 and $\lfloor \kappa e \rfloor - 1$).

  $x_k \leftarrow x_{k-1} - \frac{1}{2(l+1)} \frac{1}{h' \circ h^{-1}(f(x_{k-1})-f_\star)} g_{k-1} + \frac{l-1}{l+1}(x_{k-1}-x_{k-2})$

Output: x_n

This algorithm comes with the linear convergence rate guarantee

Theorem 6.G.3. *Algorithm 16 verifies for every n multiple of $\lfloor \kappa e \rfloor - 1$:*

$$d(x_n,\mathcal{X}_\star)^2 \leqslant \left(1-\frac{1}{\kappa e}\right)^n d(x_0,\mathcal{X}_\star)^2. \tag{6.35}$$

Proof.

From Theorem 6.3.3, running Algorithm 12 leads to the guarantee

$$f(x_n) - f_\star \leqslant h\left(\frac{d\left(x_0, \mathcal{X}_\star\right)^2}{n+1}\right). \tag{6.36}$$

From the additional assumption (6.G.1), we can upper bound the left hand size of the above and write

$$h\left(\frac{d(x_n, \mathcal{X}_\star)^2}{\kappa}\right) \leqslant h\left(\frac{d\left(x_0, \mathcal{X}_\star\right)^2}{n+1}\right).$$

Hence,

$$d(x_n, \mathcal{X}_\star)^2 \leqslant \frac{\kappa}{n+1} d(x_0, \mathcal{X}_\star)^2.$$

The latest is a contraction guarantee. Indeed, for n sufficiently large, $d(x_n, \mathcal{X}_\star)^2 < d(x_0, \mathcal{X}_\star)^2$. The average contraction factor is $\left(\frac{\kappa}{n+1}\right)^{1/n}$ and is minimized for $n \approx \lfloor \kappa e \rfloor - 1$. Choosing such a n leads to a contraction factor upper bounded by $1 - \frac{1}{\kappa e}$.

This corresponds to the convergence rate obtained by applying $\lfloor \kappa e \rfloor - 1$ steps of Algorithm 12 and then restarting it. This is described as Algorithm 16.

$\lfloor \kappa e \rfloor - 1$ steps of Algorithm 12 therefore leads to a contraction factor of $\frac{\kappa}{\lfloor \kappa e \rfloor} \leqslant \left(1 - \frac{1}{\kappa e}\right)^{\lfloor \kappa e \rfloor - 1}$. Thus applying the same algorithm restarted every $\lfloor \kappa e \rfloor - 1$ steps leads to a contraction factor of $\left(1 - \frac{1}{\kappa e}\right)^{q(\lfloor \kappa e \rfloor - 1)}$ after $q\left(\lfloor \kappa e \rfloor - 1\right)$ steps.

■

Corollary 6.G.4. *Algorithm 16 verifies for every n,*

$$f(x_n) - f_\star \leqslant h\left(e\left(1 - \frac{1}{\kappa e}\right)^n d(x_0, \mathcal{X}_\star)^2\right). \tag{6.37}$$

Proof.

Consider $n = q\left(\lfloor \kappa e \rfloor - 1\right) + r$, with $0 \leqslant r < \lfloor \kappa e \rfloor - 1$.

Combining Theorem 6.G.3 applied on $q\left(\lfloor \kappa e \rfloor - 1\right)$ steps and Theorem 6.3.3 applied for the latest r steps from starting point $x_{q(\lfloor \kappa e \rfloor - 1)}$, we get

$$\begin{aligned}
f(x_n) - f_\star &\overset{\text{Theorem 6.G.3}}{\leqslant} h\left(\frac{d(x_{q(\lfloor \kappa e \rfloor - 1)}, \mathcal{X}_\star)^2}{r+1}\right) \\
&\overset{\text{Theorem 6.3.3}}{\leqslant} h\left(\frac{\left(1 - \frac{1}{\kappa e}\right)^{q(\lfloor \kappa e \rfloor - 1)} d(x_0, \mathcal{X}_\star)^2}{r+1}\right) \\
&\leqslant h\left(\frac{\left(1 - \frac{1}{\kappa e}\right)^{-r}\left(1 - \frac{1}{\kappa e}\right)^n d(x_0, \mathcal{X}_\star)^2}{r+1}\right) \\
&\leqslant h\left(\frac{e\left(1 - \frac{1}{\kappa e}\right)^n d(x_0, \mathcal{X}_\star)^2}{r+1}\right) \\
&\leqslant h\left(e\left(1 - \frac{1}{\kappa e}\right)^n d(x_0, \mathcal{X}_\star)^2\right)
\end{aligned}$$

■

Example 6.G.5. (Logistic regression) *The logistic objective function is strictly convex and smooth. But it is not strongly convex. However, its square is still convex and is* QG^+ *(not necessarily smooth anymore), and is still not necessarily strongly convex, but is* QG^-*. Therefore, Theorem 6.G.3 provides a linear convergence guarantee for Algorithm 16.*

Note that with smoothness and convexity only, the classical theory does not guarantee linear convergence of Logistic regression. Indeed, a tighter analysis is required, using for example self-concordance (see e.g. Bach, 2010; Marteau-Ferey et al., 2019).

7

Counter-examples in first-order optimization: a constructive approach

While many approaches were developed for obtaining worst-case complexity bounds for first-order optimization methods in the last years, there remain theoretical gaps in cases where no such bound can be found. In such cases, it is often unclear whether no such bound exists (e.g. because the algorithm might fail to systematically converge) or simply if the current techniques do not allow finding them. In this work, we propose an approach to automate the search for cyclic trajectories generated by first-order methods. This provides a constructive approach to show that no appropriate complexity bound exists, thereby complementing approaches providing sufficient conditions for convergence. Using this tool, we provide ranges of parameters for which the famous Polyak heavy-ball, Nesterov accelerated gradient, inexact Gradient descent, and three-operator splitting algorithms fail to systematically converge, and show that it nicely complements existing tools searching for Lyapunov functions.

This chapter is based on our work "Counter-examples in first-order optimization: a constructive approach" (co-authored with A. Dieuleveut, and A. Taylor), published in L-CSS and CDC 2023.

Contents

7.1 Introduction

In the last years, first-order optimization methods (or algorithms) have attracted a lot of attention due to their practical success in many applications, including in machine learning (see, e.g., Bottou and Bousquet (2007)). Theoretical foundations for those methods played a crucial role in this success, e.g., by enabling the development of momentum-type methods (see, e.g., Polyak (1963); Nesterov (1983)). Formally, we consider the optimization problem

$$x_\star \triangleq \arg\min_{x\in\mathbb{R}^d} f(x) \tag{OPT}$$

for a function f belonging to a class of functions $\mathcal{F}$ (e.g., the set of convex functions, or the set of strongly convex and smooth functions, etc.). Classical first-order optimization methods for solving this problem include *Gradient descent* (GD), *Nesterov accelerated gradient method* (NAG) Nesterov (1983), and the *heavy-ball method* (HB) Polyak (1963). These families of algorithms are parametrized: for example, GD is parametrized by a step-size γ and HB is parametrized by both a step-size γ and a momentum parameter β. We generically denote by $\mathcal{A}$ any such method, for a specific choice of its parameters. For a given class of function $\mathcal{F}$ and an algorithm $\mathcal{A}$, we typically aim at answering the question

> Does $\mathcal{A}$ converge on every function of $\mathcal{F}$ to their respective minimum?

Common examples of function classes $\mathcal{F}$ include the set $\mathcal{F}_{\mu,L}$ of μ-strongly convex and L-smooth functions, and the set $\mathcal{Q}_{\mu,L}$ of μ-strongly convex and L-smooth *quadratic* functions, for $\mu, L \geqslant 0$.

This type of analysis, requiring results to hold on every function of a given class $\mathcal{F}$ is commonly referred to as *worst-case analysis* and is the most popular paradigm for the analysis of optimization algorithms, see, e.g., Nesterov (1983); Dvurechensky et al. (2021); Bubeck (2015); d'Aspremont et al. (2021); Chambolle and Pock (2016). In this context, a very successful technique for proving worst-case convergence consists in looking for a decreasing sequence (called *Lyapunov sequence* Lyapunov and Fuller (1992); Kalman and Bertram (1960a,b)) of expressions V_t of the iterates x_t, i.e. such that

$$\forall f \in \mathcal{F},\ \forall t,\ \forall x_t, \quad V_{t+1}((x_s)_{s\leqslant t+1}) \leqslant V_t((x_s)_{s\leqslant t}), \tag{7.1}$$

where some quantity of interest is upper-bounded by $V_T((x_s)_{s\leqslant T})$ as T goes to infinity. For instance, when studying GD with step-size $1/L$ on the class $\mathcal{F}_{0,L}$ of L-smooth convex functions, we prove that $\forall f \in \mathcal{F}_{0,L}$, $(t+1)(f(x_{t+1}) - f(x_\star)) + \frac{1}{2}\|x_{t+1} - x_\star\|^2 \leqslant t(f(x_t) - f(x_\star)) + \frac{1}{2}\|x_t - x_\star\|^2$. Therefore, $V_t((x_s)_{s\leqslant t}) = t(f(x_t) - f(x_\star)) + \frac{1}{2}\|x_t - x_\star\|^2$ defines a decreasing sequence, and $f(x_t) - f(x_\star) \leqslant V_t((x_s)_{s\leqslant t})/t \leqslant V_0(x_0)/t$, proving convergence of this method on this class of functions.

Due to the simplicity of the underlying proofs, the Lyapunov approach is particularly popular, e.g., for NAG Nesterov (1983); Beck and Teboulle (2009), and HB Ghadimi et al. (2015). See Bansal and Gupta (2019); d'Aspremont et al. (2021) for surveys on this topic.

Necessary condition for worst-case convergence. While finding a decreasing Lyapunov sequence guarantees convergence, not finding one does not guarantee anything: there may still exist a Lyapunov sequence, that the current analysis was not able to capture, or the method could converge without the existence of such Lyapunov sequence.

Establishing that a method *provably does not admit a worst-case convergence analysis* is therefore critical for avoiding spending an indefinite amount of time and effort searching for a non-existent convergence guarantee. The existence of a *cycle* for the algorithm on a particular function means that it diverges on that function: in other words, the absence of cycle on all functions is a necessary condition for worst-case convergence. Moreover, a cycle can be observed after only a *finite* number of steps of the algorithm, while observing the divergence of a non-periodic sequence is difficult or impossible. Overall, this makes the search for cycles a computationally practical way of proving divergence.

In order to discover cycles, we rely on computer-assisted worst-case analysis. *Performance estimation problems* (PEP, (Drori and Teboulle, 2014; Taylor et al., 2017c)) provide a systematic approach to obtain convergence guarantees, including the search for appropriate Lyapunov arguments. Some packages (especially `Pesto`, (Taylor et al., 2017b) and `Pepit` (Goujaud et al., 2022a)) automate these tasks. We formulate cycle discovery as a minimization problem that can be cast in a PEP, and rely on the `Pepit` package to solve it.

Examples: We demonstrate the applicability of our method on several examples. In particular, the case of HB illustrates the potential of our methodology. In fact, the search for the step-size γ and momentum β parameters leading to the fastest worst-case convergence over $\mathcal{F}_{\mu,L}$ is still an open problem, and the existence of parameters resulting in an accelerated rate remains a lingering question. Indeed, Lessard et al. (2016) exhibits a smooth and strongly convex function on which HB cycles, for parameters γ and β optimizing the worst-case guarantee on $\mathcal{Q}_{\mu,L}$. On the other hand, Ghadimi et al. (2015) obtains a worst-case convergence on $\mathcal{F}_{\mu,L}$ for other parameters, but without acceleration. Recently, Upadhyaya et al. (2023) proposes a very general procedure to find Lyapunov sequences and extended the region of parameters γ and β HB provably converges on, leveraging PEPs. However, outside this region of the parameter space, the question of the convergence of the HB method remains open in the absence of a proof of divergence. For this example, our approach demonstrates that a cycle exists for almost all parameters for which no Lyapunov is known.

Summary of contributions: This paper proposes a systematic approach to prove that no worst-case certificate of convergence can be obtained for a given algorithm $\mathcal{A}$ on a class $\mathcal{F}$. To do so, we establish the existence of a function in $\mathcal{F}$ over which $\mathcal{A}$ cycles. We illustrate our approach by applying it to three famous first-order optimization algorithms, namely HB, NAG, inexact gradient descent with relatively bounded error. We further showcase the applicability of the approach to more general types of problems by studying the three-operator splitting method for monotone inclusions. For each method, we describe the set of parameters for which it is known to converge and the ones where we establish the existence of a cycle. In the first three examples, our approach enables to fill the gap: we show the existence of cycles for all parametrizations not known to result in convergence.

Organization: The rest of the paper is organized as follows. In Section 7.2, we introduce the concept of a stationary algorithm and formally define a cycle. In Section 7.3, we present our methodology to discover cycles, relying on PEP. Finally, in Section 7.4, we provide the numerical results.

7.2 Definitions and notations

Notation	Corresponding object
$\mathcal{A}$	Generic algorithm
A	Update function of the algorithm $\mathcal{A}$
(HB)	Heavy-ball
(NAG)	Nesterov accelerated gradient
(IGD)	Inexact GD
(TOS)	Three operator splitting
β, γ	Algorithm parameters
$(x_t)_t$	Sequence of iterates generated by $\mathcal{A}$
x_*	Optimal point
V	Lyapunov function
f	Objective function
$\mathcal{F}$	Generic class of functions
$\mathcal{F}_{\mu,L}$	Class of L-smooth and μ-strongly convex functions
$\mathcal{Q}_{\mu,L}$	Class of L-smooth and μ-strongly convex quadratic functions
ℓ	Order of the algorithm $\mathcal{A}$
K	Length of the considered cycle
$\mathcal{O}^{(f)}$	Generic oracle applied on f
u, F, G	Linearization variables (after SPD lifting)
d	Dimension
s_K	Score

In this section, we consider a subclass of first-order methods, tailored for our analysis. It is chosen to ensure the periodicity of an algorithm that cycles once (see Proposition 7.3.1). The class reduces to "*p-stationary canonical linear iterative optimization algorithms*" (p-SCLI, see (Arjevani et al., 2016, Definition 1)) when the dependency to the previous iterates and gradients is linear which is a particular case of "*fixed-step first-order methods*" (*FSFOM*, see in (Taylor et al., 2017c, Definition 4)). Here, we consider *stationary first-order methods (SFOM)*, whose iterates are defined as a fixed function of a given number of lastly observed iterates, as well as output of some oracles called on those iterates. Examples of such oracles include gradients, approximate gradients, function evaluations, proximal step, exact line-search, Frank-Wolfe-type steps (see Taylor et al. (2017a); Goujaud et al. (2022a) for lists of oracles that can be handled using PEPs). The oracles we use depend on the setting under consideration.

Definition 7.2.1 (Stationary first-order method (SFOM))**.** *A method $\mathcal{A}$ is called order-ℓ* stationary first-order method *if there exists a deterministic first-order oracle $\mathcal{O}^{(f)}$ and a function A such that the sequence generated on the function f verifies $\forall t \geqslant \ell$,*

$$x_t = A((x_{t-s}, \mathcal{O}^{(f)}(x_{t-s}))_{s\in[\![1,\ell]\!]}). \tag{SFOM}$$

For any given function of interest f and any initialization $(x_t)_{t\in[\![0,\ell-1]\!]}$, an order-$\ell$ (SFOM) $\mathcal{A}$ iteratively generates a sequence $(x_t)_{t\in\mathbb{N}}$ that we denote $\mathcal{A}(f, (x_t)_{t\in[\![0,\ell-1]\!]})$.

Definition 7.2.1 above is very similar to the definition of a general first-order method. However, the key assumption here is that the operation A does not depend on the iteration counter t: the algorithm is *stationary*. While this assumption is restrictive, many first-order methods are of the form (SFOM), including (but not limited to): GD, HB Polyak (1963) and NAG Nesterov (2003) with constant step-sizes. On the other hand, any strategy involving decreasing step-size (e.g. for GD), or increasing momentum parameter (e.g. for NAG on $\mathcal{F}_{0,L}$ as in Nesterov (1983)) are not in the scope of this definition. Note that the aforementioned examples use the first-order oracle $\mathcal{O}^{(f)}(x) \triangleq (\nabla f(x), f(x))$, although our methodology applies beyond this simple setting, as previously discussed. As an example, 7.4.4 considers an algorithm relying on the resolvent (or proximal operation).

Stationarity is essential for being able to prove existence of a cyclical behavior in a finite number of steps. Next, we define a cyclic sequence.

Definition 7.2.2 (Cycle). *For any positive integer $K \geqslant 2$, a sequence $(x_t)_{t\geqslant 0}$ is said to be K-cyclic if $\forall t \geqslant 0, x_t = x_{t+K}$. A sequence x is said to be cyclic if there exists $K \geqslant 2$ such that x is K-cyclic.*

For any given order-ℓ (SFOM) $\mathcal{A}$, and any function class $\mathcal{F}$, we want to address the question

> Does there exist a function $f \in \mathcal{F}$ and an initialization $(x_t)_{t\in[\![0,\ell-1]\!]}$ such that $\mathcal{A}(f, (x_t)_{t\in[\![0,\ell-1]\!]})$ is cyclic?

Example 7.2.3. *In (Lessard et al., 2016, Equation 4.11), the authors answer positively to this question by providing a cycle of length 3, on the class $\mathcal{F}_{\mu,L}$ with $(\mu, L) = (1, 25)$, and for $\mathcal{A}$ the heavy-ball method with step-size $\gamma = (\frac{2}{\sqrt{L}+\sqrt{\mu}})^2$ and momentum parameter $\beta = (\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}})^2$. Those parameters are natural candidates, that correspond to the limit of the step-size and momentum in Chebychev acceleration Flanders and Shortley (1950); Lanczos (1952); Young (1953), and result in an acceleration for quadratic functions.*

In Section 7.4, we extend this result to more parameters.

7.3 Searching for cycles

In this section, we show how to find cyclic trajectories.

7.3.1 Motivation

Finding diverging trajectories for an algorithm $\mathcal{A}$ might be challenging. We thus focus on cycles, as they allow to focus on a finite sequences of iterates only. Indeed, for an SFOM, once we observe the cycle to be repeated once, we can easily extrapolate: this same cycle is repeated again and again. This statement is formalized in the following proposition.

Proposition 7.3.1. *Let $\mathcal{A}$ be a order-ℓ (SFOM), and $(x_t)_{t\in\mathbb{N}}$ be any sequence generated by $\mathcal{A}$. Then the sequence $(x_t)_{t\in\mathbb{N}}$ is cyclic if and only if there exists $K \geqslant 2$ such that*

$\forall t \in [\![0, \ell-1]\!], x_t = x_{t+K}.$

Proof. Let $\mathcal{A}$ be a order-ℓ (SFOM), and $(x_t)_{t\in\mathbb{N}}$ be any sequence generated by $\mathcal{A}$. The method $\mathcal{A}$ is cyclic if and only if there exists $K \geqslant 2$, such that the translated sequence $(\tilde{x}_t)_{t\in\mathbb{N}} := (x_{t+K})_{t\in\mathbb{N}}$, is identical to $(x_t)_{t\in\mathbb{N}}$. Proposition 7.3.1 states that those two sequences are identical if and only if their ℓ first terms are. It is clear that if the sequences x and $\tilde{x}$ are identical, their ℓ first terms also are. Reciprocally, let's assume that their ℓ first terms are identical and let's introduce the function f and associated oracles $\mathcal{O}^{(f)}$ defined such that $x = \mathcal{A}(f, (x_t)_{t\in[\![0,\ell-1]\!]})$. Then $\forall t \geqslant 0, x_t = A((x_{t-s}, \mathcal{O}^{(f)}(x_{t-s}))_{s\in[\![1,\ell]\!]})$. In particular $\forall t \geqslant 0$

$$x_{t+K} = A((x_{t+K-s}, \mathcal{O}^{(f)}(x_{t+K-s}))_{s\in[\![1,\ell]\!]}),$$

thereby reaching

$$\tilde{x}_t = A((\tilde{x}_{t-s}, \mathcal{O}^{(f)}(\tilde{x}_{t-s}))_{s\in[\![1,\ell]\!]}).$$

Consequently, $\tilde{x} = \mathcal{A}(f, (\tilde{x}_t)_{t\in[\![0,\ell-1]\!]})$, and since the ℓ first terms of x and $\tilde{x}$ are identical, $\tilde{x} = \mathcal{A}(f, (\tilde{x}_t)_{t\in[\![0,\ell-1]\!]}) = \mathcal{A}(f, (x_t)_{t\in[\![0,\ell-1]\!]}) = x$. ■

7.3.2 Approach

We now present the approach used to search for cycles, based on performance estimation problems (PEPs) Drori and Teboulle (2014); Taylor et al. (2017c). We consider an algorithm $\mathcal{A}$, a function f and initial points $(x_t)_{t\in[\![0,\ell-1]\!]}$, and run $\mathcal{A}$ on f starting on $(x_t)_{t\in[\![0,\ell-1]\!]}$. This generates the sequence $x = \mathcal{A}(f, (x_t)_{t\in[\![0,\ell-1]\!]})$. For any positive integer K, we then define the non-negative score

$$s_K(\mathcal{A}, f, (x_t)_{t\in[\![0,\ell-1]\!]}) = \sum_{t=0}^{\ell-1} \|x_t - x_{t+K}\|^2.$$

From Proposition 7.3.1, this score is identically zero if and only if $\mathcal{A}$ cycles on f when starting from $(x_t)_{t\in[\![0,\ell-1]\!]}$. This suggests that one can search for cycles of length K by minimizing the score $s_K(\mathcal{A}, f, (x_t)_{t\in[\![0,\ell-1]\!]})$ w.r.t. the function f and the initialization $(x_t)_{t\in[\![0,\ell-1]\!]}$.

Observe that fixed points of $\mathcal{A}$, that correspond to cycles of length 1, also cancel this score. Our goal is to search for cycles of length at least $K \geqslant 2$, that entail that the algorithm diverges for a particular function and initialization. As any convergent algorithm must admit the optimizer of f as fixed point, we have to exclude fixed points. To do so, we add the constraint that the two first iterates are far from each other. In most cases of interest, making this constraint can be done without loss of generality due to the homogeneity of the underlying problems. We arrive to the following formulation:

$$\left| \begin{array}{ll} \underset{d\geqslant 1, f\in\mathcal{F}, x\in\left(\mathbb{R}^d\right)^{\mathbb{N}}}{\text{minimize}} & \sum_{t=0}^{\ell-1} \|x_t - x_{t+K}\|^2 \\ \text{subject to} & \begin{cases} x = \mathcal{A}(f, (x_t)_{t\in[\![0,\ell-1]\!]}) \\ \|x_1 - x_0\|^2 \geqslant 1. \end{cases} \end{array} \right. \tag{$\mathcal{P}$}$$

As we see in the next sections, this problem can be used to answer the question of interest by testing the nullity of the solution of ($\mathcal{P}$).

As is, ($\mathcal{P}$) looks intractable due to the minimization over the infinite-dimensional space $\mathcal{F}$ and its non-convexity. This can be handled using the techniques proposed in Taylor et al. (2017c,a), developed for PEP. It consists in reformulating ($\mathcal{P}$) into a semi-definite program (SDP) using interpolation / extension properties for the class $\mathcal{F}$, together with SDP lifting.

Indeed, ($\mathcal{P}$) does not fully depend on f, but only on $\mathcal{O}^{(f)}(x_t)$ where $t \in [\![0; K+\ell-2]\!]$. By introducing the variables $\mathcal{O}_t \triangleq \mathcal{O}^{(f)}(x_t)$, we can replace the constraint $x = \mathcal{A}(f, (x_t)_{t\in[\![0,\ell-1]\!]})$ of ($\mathcal{P}$) by

$$\begin{cases} x_\ell &= A((x_{\ell-s}, \mathcal{O}_{\ell-s})_{s\in[\![1,\ell]\!]}), \\ &\vdots \\ x_{K+\ell-1} &= A((x_{K+\ell-1-s}, \mathcal{O}_{K+\ell-1-s})_{s\in[\![1,\ell]\!]}), \end{cases}$$

and minimize over the finite dimensional variables $(\mathcal{O}_t)_{0\leqslant t\leqslant K+\ell-2}$ instead of f, under the constraint that there exists a function $f \in \mathcal{F}$ that interpolates those values, i.e. that verifies $\mathcal{O}^{(f)}(x_t) = \mathcal{O}_t$ for all $t \in [\![0; K+\ell-2]\!]$. For some classes $\mathcal{F}$, those interpolation property are equivalent to tractable inequalities, as in the following example.

Example 7.3.2 (L-smooth convex functions)**.** *If the oracles are only the gradients and the function values of the objective function f, denoting $f_i \triangleq f(x_i)$ and $g_i \triangleq \nabla f(x_i)$ (i.e. $\mathcal{O}_i \triangleq (g_i, f_i)$), the interpolation conditions of $\mathcal{F}_{0,L}$ are provided in Taylor et al. (2017c) as*

$$\forall i, j,\ f_i \geqslant f_j + \langle g_j, x_i - x_j\rangle + \tfrac{1}{2L}\|g_i - g_j\|^2. \tag{IC}$$

This function class is considered in three of the four examples under consideration in the next section (HB, NAG, inexact GD). However, the methodology described in this paper applies to many other classes beyond $\mathcal{F}_{\mu,L}$, see for instance (Goujaud et al., 2022a, Function classes): an example of such a class is used in the fourth example of the following section. Each class considered must be described by its interpolation conditions, similar to (IC). Other examples of known interpolations conditions are provided in (Taylor et al., 2017a, Th. 3.4-3.6), (Dragomir et al., 2021, Cor.1&2), (Guille-Escuret et al., 2022, Th.1) (Goujaud et al., 2022c, Th. 2.6). The key ingredient for a class to enter the scope of this paper, is that its interpolation conditions are expressed as a degree 2 polynomial in x_t and $\mathcal{O}_t$, and that a given variable is not involved both in a monomial of degree 1 and one of degree 2, as in (IC).

Then, the SDP lifting part consists in introducing a Gram matrix G (Taylor et al., 2017c, Theorem 5) of vectors among x_t and $\mathcal{O}_t$ that are involved in degree 2 monomials, so that those quadratic expressions of x_t and $\mathcal{O}_t$ are then expressed linearly in term of $G \succcurlyeq 0$. Thereby, the problem can be cast a standard SDP.

In the case where the oracle is $\mathcal{O}_t = (g_t, f_t)$, and the class of interest $\mathcal{F}$ is the class of L-smooth convex functions $\mathcal{F}_{0,L}$, the objective and all the constraints of ($\mathcal{P}$) are written linearly in terms of $(f_t)_t$ and quadratically in terms of $(x_t, g_t)_t$. Therefore, we define G as the Gram matrix of $(x_t, g_t)_t$ and F as a vector storing all the values f_t leading to an SDP reformulation of the problem. See, e.g., Taylor (2020) for a detailed derivation on a simple example.

Setting $u \triangleq (G, F)$, ($\mathcal{P}$) is generally rewritten, under above-mentioned key ingredients, as

$$\left| \begin{array}{ll} \underset{u}{\text{minimize}} & \langle u, v_{obj} \rangle \\ \text{subject to} & \begin{cases} \langle u, v_1 \rangle \geqslant 0 \\ \cdots \\ \langle u, v_n \rangle \geqslant 0 \\ \langle u, v_{\text{aff}} \rangle \geqslant 1 \\ u \in \mathcal{C}. \end{cases} \end{array} \right. \tag{SDP-$\mathcal{P}$}$$

The objective is linear, as well as the first n constraints. The affine constraint $\langle u, v_{\text{aff}} \rangle \geqslant 1$ enables to discard the trivial solution $u = 0$ and corresponds in ($\mathcal{P}$) to the constraint $\|x_1 - x_0\|^2 \geqslant 1$. Finally, the constraint $u \in \mathcal{C}$ corresponds to the constraint $G \succcurlyeq 0$. $\mathcal{C}$ is then a closed convex semi-cone.

By definition, if there exists a feasible vector u such that the objective of (SDP-$\mathcal{P}$) is zero, then it describes a cycle. Moreover (SDP-$\mathcal{P}$) is convex and efficiently solvable (due to the existence of a Slater point (Taylor et al., 2017c, Theorem 6)).

In the next sections, we numerically apply this methodology through the `Pepit` python package Goujaud et al. (2022a) which takes care about the tractable reformulations of ($\mathcal{P}$) into (SDP-$\mathcal{P}$). (SDP-$\mathcal{P}$) is then solved using through a standard solver MOSEK (2019) to determine the infimum value of $\langle u, v_{\text{obj}} \rangle$ over the feasible set of (SDP-$\mathcal{P}$). The next theorem allows to conclude about the existence of cycles.

Theorem 7.3.3. *Assuming the infimum value of ($\mathcal{P}$) to be 0, then there exists a cycle.*

Proof. By equivalence between ($\mathcal{P}$) and (SDP-$\mathcal{P}$), we assume that there exists a sequence of feasible vectors u_i with $\langle u_i, v_{\text{obj}} \rangle \to 0$.

The constraint $\langle u_i, v_{\text{aff}} \rangle \geqslant 1$, guarantees that none of the u_i is equal to 0. Considering any norm (all equivalent to each other in finite dimension) and projecting u_i on the associated sphere defines $s_i \triangleq \frac{u_i}{\|u_i\|}$. The other n *linear* constraints still hold after the scaling and $s_i \in \mathcal{C}$ since $\mathcal{C}$ is a semi-cone.

Moreover, the norms of all u_i are lower bounded by the distance from 0 to the affine hyperplan $\{w | \langle w, v_{\text{aff}} \rangle = 1\}$. Hence $|\langle s_i, v_{\text{obj}} \rangle| = \frac{|\langle u_i, v_{\text{obj}} \rangle|}{\|u_i\|} \to 0$.

Finally by compacity of the sphere, there exists a subsequent limit s of the sequence $(s_i)_i$ and by continuity of the linear operator $\langle \cdot, v_{\text{obj}} \rangle$, $\langle s, v_{\text{obj}} \rangle = 0$.

We conclude that s is a vector the objective reaches 0 on, that verifies all the constraints of (SDP-$\mathcal{P}$) but the affine one.

Note the affine constraint only aimed at discarding the trivial solution 0 in a linear way (for solver purpose), and that s is not 0. Then s describes a cycle. ■

7.4 Application to four different (SFOM)s

In this section we illustrate this methodology on four examples: heavy-ball (HB), Nesterov accelerated gradient (NAG), Gradient descent (GD) with inexact gradients, and three-operator splitting (TOS). For each, we apply the methodology proposed in Section 7.3. The code is available in the public GitHub repository `https://github.com/bgoujaud/`

`cycles`. Since ingredients are the same as those of classical PEPs, we also use the python package `Pepit` Goujaud et al. (2022a). We perform a grid search over the spaces of *parameters of interest* Ω, described in the respective subsections. We compare the parameter region where Lyapunov functions can be obtained with the region in which we establish that the method cannot have a guaranteed worst-case convergence (due to the existence of cycles). More precisely, in Figures 7.1 to 7.4 below, green regions correspond to parameter choices for which the methods converge (existence of a Lyapunov function, found using the technique described in Taylor et al. (2018a)). Conversely, in the red regions, our methodology establishes that the method cycles on at least one function of $\mathcal{F}$. In short, the algorithms converge in the green regions and do not converge in the worst-case in the red ones.

Note that some parameters for which the algorithm $\mathcal{A}$ admits a worst-case convergence guarantee could theoretically exist outside the green region: indeed, in Taylor et al. (2018a), the authors do not guarantee that they necessarily find convergence. Similarly, parameters for which $\mathcal{A}$ does *not* admit a worst-case convergence guarantee could theoretically exist outside the red region: indeed ($\mathcal{P}$) is defined for a fixed cycle length K, and we therefore run it several times with different values of K. Longer cycles are therefore not detected. Moreover, the non-existence of cycles does not necessarily imply that the algorithm always converges.

Interestingly, in practice, we observe on Figures 7.2 and 7.3 that the set Ω of parameters of interest is completely filled by the union of those 2 regions and that it is almost the case on Figure 7.1 (it may have been if we had searched for cycles of all lengths). As a consequence, we fully characterize the tunings for which the algorithms admit a guaranteed worst-case convergence. On the contrary, there remains a significant gap between the red and the green regions in our last example, see Figure 7.4.

7.4.1 Heavy-ball

The HB algorithm, as introduced by Polyak (1963), corresponds to the following update, for a step-size parameter γ and a momentum parameter β:

$$x_{t+1} = x_t + \beta(x_t - x_{t-1}) - \gamma\nabla f(x_t). \tag{HB}$$

Therefore (HB) is an order-2 (SFOM).

Set Ω_{HB} of parameters of interest: HB converges on the set $\mathcal{Q}_{0,L}$ if and only if the parameters γ, β verify $0 \leqslant \gamma \leqslant 2(1+\beta)/L \leqslant 4/L$ Polyak (1963). Note this condition enforces $-1 \leqslant \beta \leqslant 1$. Moreover, we restrict to $\beta \geqslant 0$ as $\beta < 0$ is not an interesting setting (slowing down convergence with respect to GD). Therefore, we limit our analysis to this set of parameters.

Interpretation. The red area in Figure 7.1 shows parameters where cycles of length $K \in [\![2; 25]\!]$ are found by our methodology. The red color intensity indicates the length of the shortest cycle.

A striking observation is that the space Ω_{HB} is almost filled by the union of the red area and green one (where Lyapunov functions exist). Thereby, for almost all values of the parameters, we have a definitive answer on the existence of a certificate of convergence in the worst-case. That being said, there exists a small unfilled region in the top left corner (see the zoom on Figure 7.1) In this region, we do not know how HB behaves, and whether it accelerates. However, adding longer cycle length may enable to obtain cycles in that

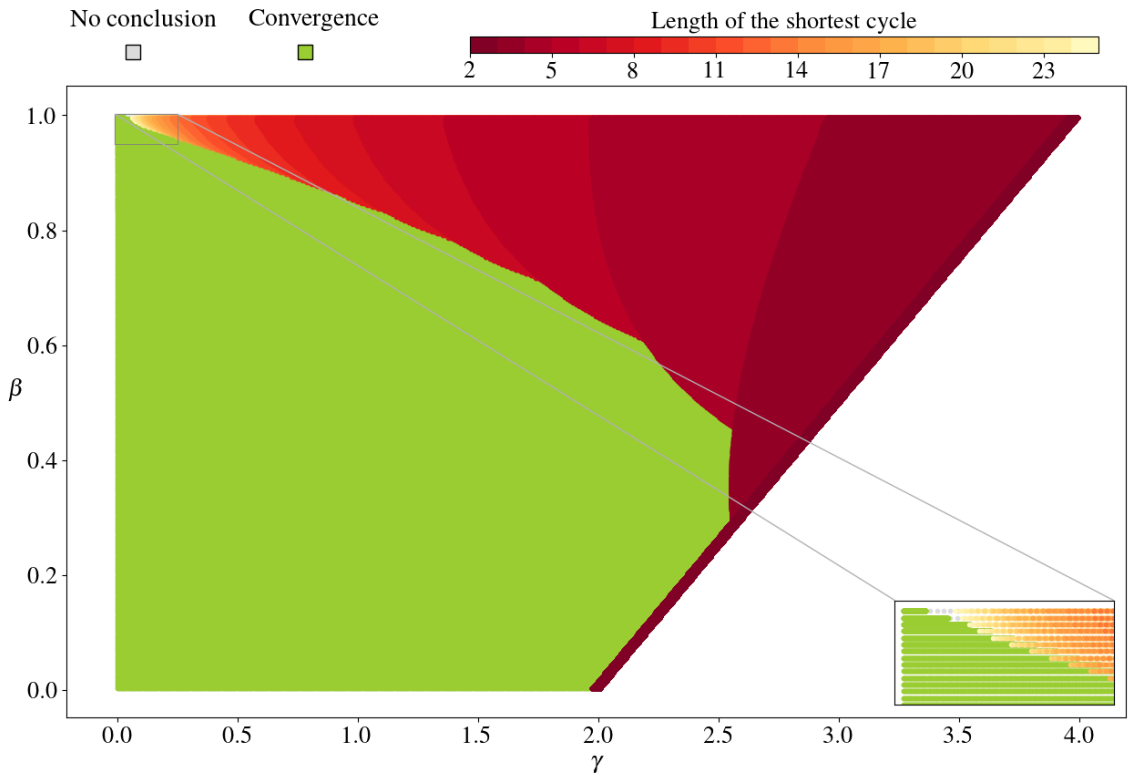


Figure 7.1: Heavy-ball (HB). Green area: set of parameters $(\gamma, \beta) \in \Omega_{\text{HB}}$ for which a Lyapunov function has been found using the technique described in Taylor et al. (2018a); Red area: set of parameters $(\gamma, \beta) \in \Omega_{\text{HB}}$ for which solving (SDP-$\mathcal{P}$) shows that (HB) cycles on at least one function in $\mathcal{F}_{0,L}$.

area. Indeed we considered only cycles of length $K \leqslant 25$, for computational reasons. Recently, Goujaud et al. (2023c) computed the analytical region of cycles of (HB) showing non-acceleration of the latter.

7.4.2 Nesterov accelerated gradient

NAG (also known as the *fast gradient method*) was introduced by Nesterov (2003) and corresponds to the following update, for a step-size parameter γ and a momentum parameter β:

$$\left\{ \begin{array}{rcl} y_t & = & x_t + \beta(x_t - x_{t-1}), \\ x_{t+1} & = & y_t - \gamma \nabla f(y_t). \end{array} \right. \tag{NAG}$$

(NAG) is also written as follows $y_{t+1} = (1+\beta)(y_t - \gamma\nabla f(y_t)) - \beta(y_{t-1} - \gamma\nabla f(y_{t-1}))$, and is therefore also an order-2 (SFOM).

Set Ω_{NAG} of parameters of interest: As for HB, we consider the set of β and γ for which (NAG) converges on $\mathcal{Q}_{0,L}$. This corresponds to considering all β, γ verifying $0 \leqslant \beta \leqslant 1$ and $0 \leqslant \gamma \leqslant \frac{2}{L}\frac{1+\beta}{1+2\beta}$.

Interpretation: (NAG) is known to converge, with an accelerated rate, on $\mathcal{F}_{\mu,L}$, for the tuning $(\gamma, \beta) = (\frac{1}{L}, \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}})$, that optimizes the convergence rate on $\mathcal{Q}_{\mu,L}$. For this reason, (NAG) is considered to be more "robust" than HB.

Figure 7.2 shows that (NAG) admits a Lyapunov function for almost any parameters in Ω_{NAG}. Moreover, our methodology does not detect any set of parameters at which a

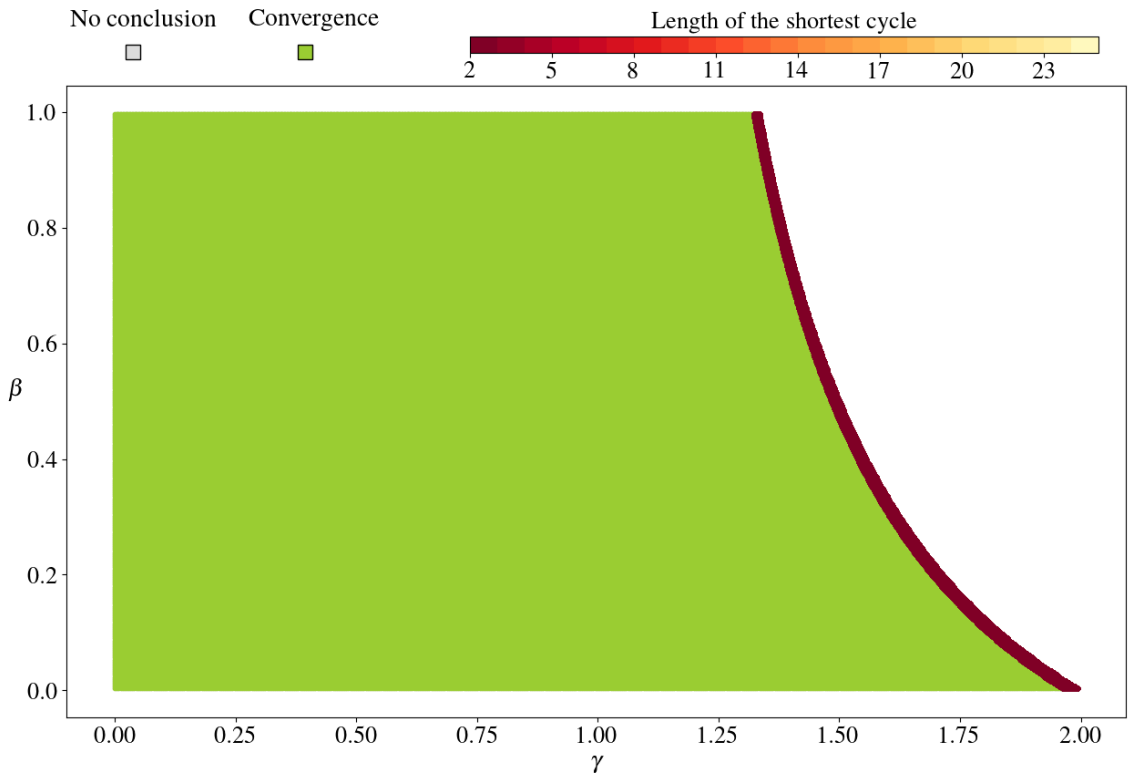


Figure 7.2: Nesterov Accelerated gradient (NAG). Green area: set of parameters $(\gamma, \beta) \in \Omega_{\text{NAG}}$ for which a Lyapunov function has been found using the technique described in Taylor et al. (2018a); Red area: set of parameters $(\gamma, \beta) \in \Omega_{\text{NAG}}$ for which solving (SDP-$\mathcal{P}$) shows that (NAG) cycles on at least one function in $\mathcal{F}_{0,L}$.

cycle of length $K \in [\![2; 25]\!]$ exists, apart on the boundary $\{(\gamma, \beta), \gamma = \frac{2}{L}\frac{1+\beta}{1+2\beta}\}$. On the boundary, cycles of length 2 are observed, whose existences are theoretically verified on one-dimensional quadratic functions. This illustrates the robustness of our methodology when very few cycles exist.

7.4.3 Inexact gradient method

Next, we consider the inexact gradient method, parameterized by γ and ε, and the update

$$\begin{aligned} \text{Get } \mathcal{O}(x_t) &= d_t \text{ such that } \|d_t - \nabla f(x_t)\| \leqslant \varepsilon \|\nabla f(x_t)\|, \\ x_{t+1} &= x_t - \gamma d_t. \end{aligned} \tag{IGD}$$

(IGD) is thus an (SFOM) of order 1.

Set Ω_{IGD} of parameters of interest: Since the exact gradient method converges only for $\gamma < \frac{2}{L}$, we only consider such steps-sizes. Moreover, $\varepsilon \geqslant 1$ allows $d_t = 0$ and thereby does not make much sense. This motivates considering the set $\Omega_{\text{IGD}} = \{(\gamma, \varepsilon) \in [0; \frac{2}{L}] \times [0, 1]\}$.

Interpretation: We search for cycles of length $K \in [\![2; 25]\!]$ and use color intensity to show the minimal cycle length. (IGD) is known to converge for any $\gamma \leqslant \frac{2}{L(1+\varepsilon)}$ (see De Klerk et al. (2020); Gannot (2021)). Figure 7.3 shows that the complementary of this region of convergence is completely filled by parameters allowing cycles, showing that no other parameters values than the known ones allow obtaining worst-case convergence of (IGD).

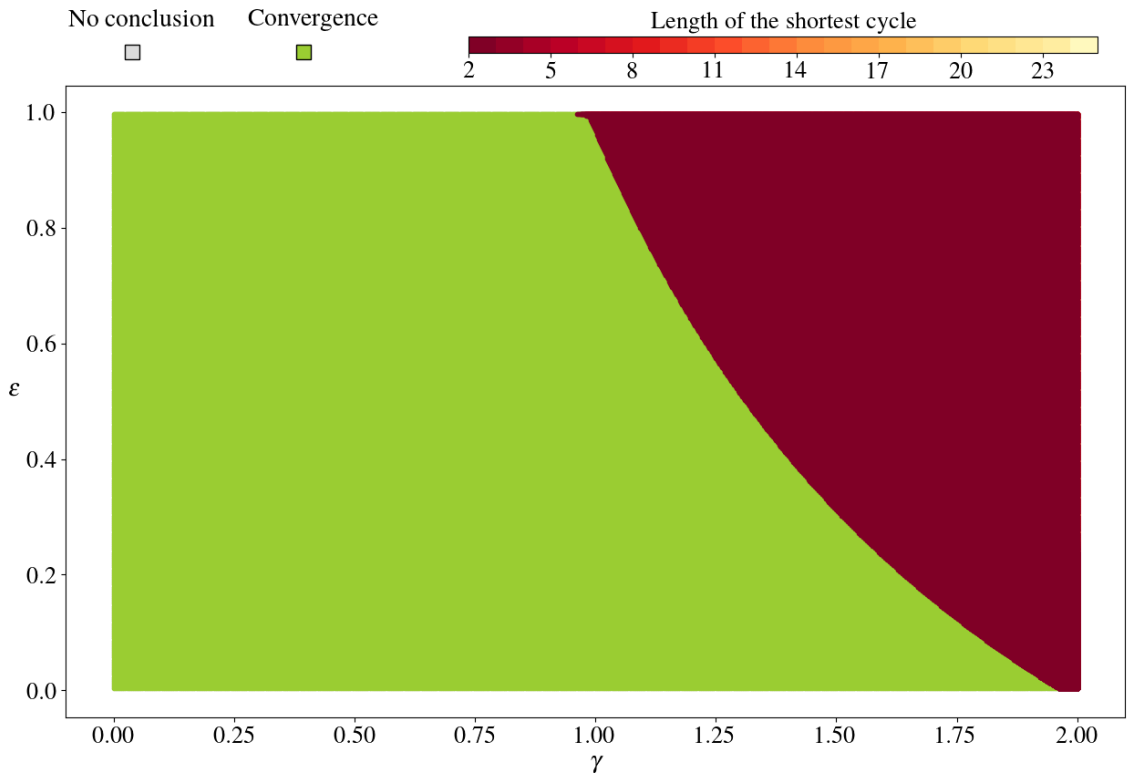


Figure 7.3: Inexact GD (IGD). Green area: set of parameters $(\gamma, \varepsilon) \in \Omega_{\text{IGD}}$ for which a Lyapunov function has been found using the technique described in Taylor et al. (2018a); Red area: set of parameters $(\gamma, \varepsilon) \in \Omega_{\text{IGD}}$ for which solving (SDP-$\mathcal{P}$) shows that (IGD) cycles on at least one function in $\mathcal{F}_{0,L}$.

7.4.4 Three-operator splitting

The three-operator splitting (TOS) method, introduced by Davis and Yin (2017), aims at solving the *inclusion problem* $0 \in Ax + Bx + \partial f(x)$, where A is a monotone operator, B is a co-coercive operator and ∂f denotes the differential of the smooth (strongly) convex function f. It corresponds to the following update, for a step-size parameter γ, a smoothing parameter α, and an update parameter β:

$$\left\{ \begin{array}{rcl} x_{t+1} & = & J_{\alpha B}(w_t), \\ y_{t+1} & = & J_{\alpha A}\left(2x_{t+1} - w_t - \frac{\gamma}{\beta}\nabla f(x_{t+1})\right), \\ w_{t+1} & = & w_t - \beta(x_{t+1} - y_{t+1}), \end{array} \right. \tag{TOS}$$

where J_O denotes the resolvent of the operator O, i.e. $J_O = (I + O)^{-1}$. Note that (TOS) is therefore an order-1 (SFOM).

Set Ω_{TOS} of parameters of interest: When considering A, B and ∇f to be linear symmetric and co-diagonalizable operators, the set of convergence of (TOS) is $\Omega_{\text{TOS}} = \left\{(\gamma, \beta) \in \left[0, \frac{2}{L}\right] \times [0, 2]\right\}$, which we therefore consider.

Interpretation: We search for cycles of length $K \in [\![2, 25]\!]$ and use color intensity to show the minimal cycle length. Interestingly, there is a gap between the green region and the red ones. Unlike for (HB), it seems that increasing the length of the cycle does not help covering this gap and shows that some algorithms might have no Lyapunov function while

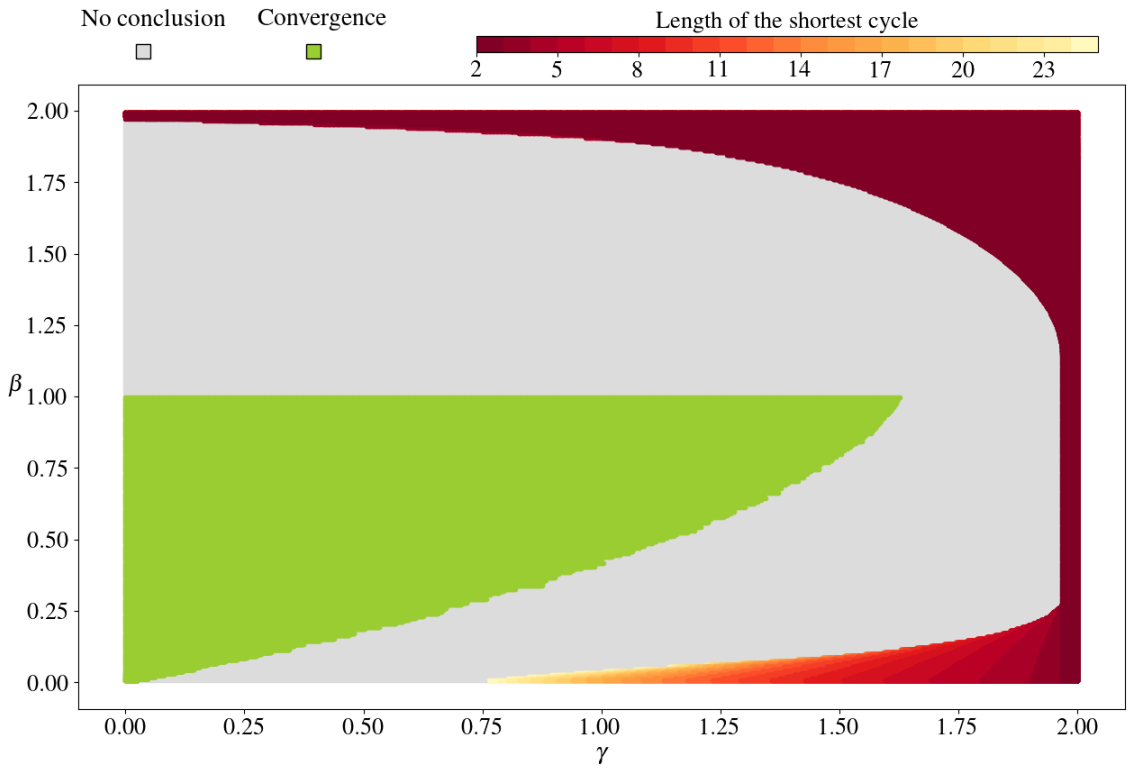


Figure 7.4: Three-operator splitting (TOS). Green area: set of parameters $(\gamma, \beta) \in \Omega_{\text{TOS}}$ for which a Lyapunov function has been found using the technique described in Taylor et al. (2018a); Red area: set of parameters $(\gamma, \beta) \in \Omega_{\text{TOS}}$ for which solving (SDP-$\mathcal{P}$) shows that (TOS) cycles on at least a triplet of operators.

not cycling. Understanding the behavior of (TOS) in the grey region is therefore still an open question.

7.5 Conclusions

Summary. This work proposes a systematic approach for finding counter-examples to convergence of first-order methods, bringing a complementary tool to the existing systematic techniques for finding convergence guarantees (that include certifications through the existence of Lyapunov functions). Our approach is based on now classical tools and techniques used in the field of first-order optimization and a few existing packages Goujaud et al. (2022a); Taylor et al. (2017b) allow for straightforward implementations of our methodology.

Discussion and Future works. While our analysis complements the Lyapunov one, the existence of a Lyapunov function or the existence of a cycle are not the only 2 options.

Indeed, the sequence of iterates produced by an algorithm on a given function may diverge by (i) tending to infinity, (ii) simply growing unbounded, or even (iii) showing a chaotic behavior, while staying in a compact set.

Being capable of detecting those three divergence cases is therefore an open but interesting question.

An interesting example is $\mathcal{A}$ being GD with step-size 1 on the $1 + \rho$-smooth and non-

convexe function f_ρ, such that $f_\rho(x)$ is equal to:

$$\begin{cases} \frac{\rho}{3}|x|^3 + \frac{1-\rho}{2}x^2 + \frac{(\rho-1)^3}{6\rho^2} & \text{if } |x| \leqslant 1, \\ -\frac{\rho}{3}|x|^3 + \frac{1+\rho}{2}x^2 - 2\rho|x| + \frac{(\rho-1)^3+4\rho^3}{6\rho^2} & \text{if } 1 \leqslant |x| \leqslant \frac{3}{2}, \\ \frac{1}{2}x^2 + \frac{\rho}{4}|x| + \frac{4(\rho-1)^3+11\rho^3}{24\rho^2} & \text{if } \frac{3}{2} \leqslant |x|, \end{cases}$$

for $\rho \in [0, 4]$.

From any point in the interval $(1, \infty)$, the next iterate is in $[-1, 0]$. Similarly, starting in the interval $(-\infty, -1)$, the second iterate is in $[0, 1]$. Those 2 intervals are stable and, by symmetry of the function, the dynamics in those 2 intervals are themselves symmetric. Note that for any $x \in [0, 1]$, $f'(x) = \rho x^2 + (1 - \rho)x$, leading to the dynamic $x_{t+1} = x_t - 1 \times \nabla f_\rho(x_t) = \rho x_t(1 - x_t)$, known as *logistic map*. The behavior of this dynamic is highly dependent on the value of ρ. On the first hand, for $\rho < 3$, $\mathcal{A}(f_\rho, x_0)$ converges for any x_0. On the other hand, for almost all values of ρ close enough to 4, this dynamic is chaotic. Note however, that for any $\rho_0 < 4$, there exists $\rho > \rho_0$ such that Gradient descent with step-size 1 cycles on f_ρ.

Therefore, on the class $\{f_\rho, \rho \in [0, 4]\}$ we have: functions over which $\mathcal{A}$ converges, functions over which it diverges because it is chaotic, but also functions over which it cycles. This example thus shows that those behaviors can co-exist, but does not provide an answer to the open question.

8

Provable non-accelerations of the heavy-ball method

In this work, we show that the Heavy-ball (HB) method provably does not reach an accelerated convergence rate on smooth strongly convex problems. More specifically, we show that for any condition number and any choice of algorithmic parameters, either the worst-case convergence rate of HB on the class of L-smooth and μ-strongly convex *quadratic* functions is not accelerated (that is, slower than $1 - \mathcal{O}(\kappa)$), or there exists an L-smooth μ-strongly convex function and an initialization such that the method does not converge.

To the best of our knowledge, this result closes a simple yet open question on one of the most used and iconic first-order optimization technique.

Our approach builds on finding functions for which HB fails to converge and instead cycles over finitely many iterates. We analytically describe all the parametrizations of HB that exhibit this cycling behavior on a particular cycle shape, whose choice is supported by a systematic and constructive approach to the study of cycling behaviors of first-order methods. We show the robustness of our results to perturbations of the cycle and extend them to classes of functions that also satisfy higher-order regularity conditions.

This chapter is based on our work "Provable non-accelerations of the heavy-ball method" (co-authored with A. Taylor, and A. Dieuleveut), currently under review.

Contents

8.1 Introduction

In this paper, we consider the unconstrained minimization of a convex function $f : \mathbb{R}^d \to \mathbb{R}$:

$$x_\star \triangleq \arg\min_{x \in \mathbb{R}^d} f(x), \tag{OPT}$$

where f belongs to a given class of functions $\mathcal{F}$ (e.g., the set of convex quadratic functions, or the set of strongly convex and smooth functions). First-order optimization methods have recently attracted a lot of attention due to their generally low cost per iteration and their practical success in many applications. They are particularly relevant in applications not requiring very accurate solutions, such as in machine learning (see, e.g., Bottou and Bousquet (2007)).

A major weakness of those methods is that their convergence speed is typically slow, and crucially affected by the so-called *conditioning* of the function to be minimized (more in the sequel). In this context, the theoretical foundations for first-order methods played a crucial role in their success, among others by enabling the development of momentum-type methods for mitigating the impact of the conditioning on the convergence rates. That is, momentum-type methods usually behave much better both theoretically and practically as compared to the vanilla *Gradient descent* (GD), which is probably the most well-known and iconic first-order method. Momentum-type methods are usually classified in two categories depending on how the momentum appears in the iterative update equations. As a reference, the update rule for the vanilla *Gradient descent* (GD) method for solving (OPT) is

$$x_{t+1} = x_t - \gamma \nabla f(x_t), \tag{GD}$$

for some step-size γ. In (Polyak, 1964), Polyak introduced the first momentum-type strategy, the celebrated *Heavy-ball* (HB) update rule:

$$x_{t+1} = x_t - \gamma \nabla f(x_t) + \beta(x_t - x_{t-1}). \tag{HB}$$

HB is notorious for being, among others, an *optimal* method for minimizing convex quadratic functions, as its computational complexity matches that of the corresponding lower complexity bounds (Nemirovskii, 1994). A few years later, Nesterov introduced the *accelerated gradient* method (Nesterov, 1983) which consists in iterating

$$x_{t+1} = x_t - \gamma \nabla f(x_t + \beta(x_t - x_{t-1})) + \beta(x_t - x_{t-1}), \tag{NAG}$$

which is often referred to as Nesterov's accelerated gradient (NAG).

In words, the Gradient descent update is complemented with a momentum term $\beta(x_t - x_{t-1})$ being either applied after the gradient evaluation (for HB) or before it (for NAG) at each iteration. As compared to the HB method, NAG is an *optimal* algorithm on the class of smooth strongly convex functions f (i.e., beyond quadratics). Yet, HB is known to be asymptotically twice faster than Nesterov on the class of quadratic functions. Moreover, HB is among the most prevalent practical first-order optimization paradigms used in optimization software (e.g., for machine learning): it works well in many situations, though the question of its theoretical convergence speed is still open beyond quadratics.

Conditioning and its effects. This work primarily focuses on a particular classical set of functions, namely the set of *smooth and strongly convex functions* (additional higher-order regularity assumptions are considered later in Section 8.6). This set is very standard in the

first-order optimization literature; see, e.g., Polyak (1987); Nemirovskii (1994); Nesterov (2003).

Definition 8.1.1 (Set $\mathcal{F}_{\mu,L}$)**.** *Let $0 \leqslant \mu \leqslant L < \infty$. A continuously differentiable function $f : \mathbb{R}^d \mapsto \mathbb{R}$ is L-smooth and μ-strongly convex (notation $f \in \mathcal{F}_{\mu,L}$) if:*

- *(L-smoothness) for all $x, y \in \mathbb{R}^d$, it holds that*

$$f(x) \leqslant f(y) + \langle \nabla f(y),\, x - y \rangle + \frac{L}{2}\|x - y\|^2,$$

- *(μ-strong convexity) for all $x, y \in \mathbb{R}^d$, it holds that*

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

In particular, twice differentiable L-smooth μ-strongly convex functions corresponds to functions whose Hessian have bounded eigenvalues between μ and L (i.e., $\mu\, I \preccurlyeq \nabla^2 f(x) \preccurlyeq L\, I$, or $\mathrm{Sp}\left(\nabla^2 f(x)\right) \subseteq [\mu,\, L]$, for all $x \in \mathbb{R}^d$). A particular subclass of $\mathcal{F}_{\mu,L}$ is that of quadratic functions (i.e., with constant Hessian).

Definition 8.1.2. *Let $0 \leqslant \mu \leqslant L < \infty$. A continuously differentiable function $f(x) = \langle x,\, Hx \rangle + \langle b,\, x \rangle + c$ is an L-smooth μ-strongly convex quadratic function (notation $f \in \mathcal{Q}_{\mu,L}$) if and only if $\mathrm{Sp}\,(H) \subseteq [\mu,\, L]$.*

Those sets of functions are very standard for the analyses of first-order methods and are characterized by the (inverse) condition number $\kappa \triangleq \frac{\mu}{L}$. In this context, it is known that GD converges exponentially fast both for minimizing functions in $\mathcal{Q}_{\mu,L}$ and $\mathcal{F}_{\mu,L}$. More precisely, $\|x_t - x_\star\| \leqslant \left(\frac{1-\kappa}{1+\kappa}\right)^t \|x_0 - x_\star\|$ for suitable choices of the step-size γ. On $\mathcal{Q}_{\mu,L}$, an optimally-tuned HB has $\|x_t - x_\star\| \leqslant C\left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^t \|x_0 - x_\star\|$ (for some $C > 0$) and is therefore (much) faster for small values of κ. However, the optimal behavior of HB beyond quadratics, on $\mathcal{F}_{\mu,L}$, is unknown to the best of our knowledge. Finally, NAG has $\|x_t - x_\star\| \leqslant C\,(1 - \sqrt{\kappa})^{t/2} \|x_0 - x_\star\|$ on $\mathcal{F}_{\mu,L}$ but is suboptimal compared to HB when working on $\mathcal{Q}_{\mu,L}$, though much faster than GD.

In short, albeit generally good practical performances, it remains unclear for which choices of (γ, β) (if any), HB allows obtaining *fast* convergence speeds on the standard class of L-smooth μ-strongly convex problems. In this work, we answer this question by proving that it is suboptimal for minimizing smooth strongly convex functions. More precisely, we show that the only choices of (γ, β) for which HB is guaranteed to converge on $\mathcal{F}_{\mu,L}$ only marginally improve upon the convergence speed of Gradient descent.

8.1.1 Related works

As HB is one of the most widely used methods, understanding its worst-case convergence rate on larger class of functions is a natural problem.

Behavior of HB on quadratics. The HB method was originally coined for optimizing quadratic functions Polyak (1964). A classical approach to the analysis of (HB) on $\mathcal{Q}_{\mu,L}$ consists in exploiting links between first-order methods and polynomials (see e.g. Fischer (2011); Nemirovskii (1994) or d'Aspremont et al. (Chapter 2 of 2021) for a recent introduction–for more recent exploitation of those links, see, e.g., Berthier et al. (2020);

Pedregosa and Scieur (2020); Goujaud et al. (2022b,d); Kim et al. (2022)). In this context, a standard choice of HB parameters is $(\gamma^\star(\mathcal{Q}_{\mu,L}), \beta^\star(\mathcal{Q}_{\mu,L})) = ((\frac{2}{\sqrt{L}+\sqrt{\mu}})^2, (\frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}})^2)$ and is often referred to as the *optimal tuning* for quadratics.

Behavior of HB beyond quadratics. The class of L-smooth and μ-strongly convex functions (notation $\mathcal{F}_{\mu,L}$) is extensively studied in the first-order optimization literature (see, e.g., (Polyak, 1987; Nesterov, 2003; Bubeck, 2015)), and is a natural candidate for exploring the performance of HB beyond quadratics. Notable results on HB beyond quadratics include the work of Ghadimi et al. (2015) who provide non-accelerated convergence results of HB for a large set of parameters (γ, β) (more details in the subsequent sections). Also, Lessard et al. (2016) provide a non-convergence result of HB (with the optimal parameter choice for quadratics) on $\mathcal{F}_{\mu,L}$, based on a counter-example, i.e., some $f \in \mathcal{F}_{\mu,L}$ for which given a particular initialization, the HB algorithm cycles over a finite number of values, without ever approaching the set of minimizers of f. Moreover, other attempts were made to obtain an accelerated rate on HB on the class $\mathcal{F}_{\mu,L}$, without definitive results Dobson et al. (2023); or on even larger class of functions (Goujaud et al., 2022c).

As a conclusion, despite the existence of both positive and negative convergence results for HB on $\mathcal{F}_{\mu,L}$, the optimal tuning of HB on this specific class as well as its worst-case convergence rate, remain unknown.

Erroneous convergence results on HB. A few recent works either claim or use the fact that HB does converge with an accelerated convergence rate. For instance, Wang et al. (2022) prove convergence of HB by implicitly assuming co-diagonalisation, and hence their results is actually only valid in dimension one. This result is thereby complemented by this work, which shows we cannot achieve acceleration as soon as the dimension is at least two. Another example is that of Gupta et al. (2021, Corollary 22) whose proof relies on an hypothetical accelerated convergence rate of HB.

Optimal methods on $\mathcal{F}_{\mu,L}$. Whereas convergence rates for HB were unclear on $\mathcal{F}_{\mu,L}$, there exist alternative optimal methods (which are, however, slower than HB on $\mathcal{Q}_{\mu,L}$) on this class, commonly referred to as *accelerated gradient methods*, such as (NAG), see Nesterov (1983, 2003). Whereas the dependency w.r.t. κ is essentially optimal for NAG (whose convergence rate is $(1-\sqrt{\kappa})^{1/2}$), it can be improved to $(1-\sqrt{\kappa})$ (see (Taylor and Drori, 2022) for the optimal algorithm and (Van Scoy et al., 2017) for its stationary version) thereby matching the exact lower complexity bound for $\mathcal{F}_{\mu,L}$ (Drori and Taylor, 2022) (which is more technical than that for $\mathcal{Q}_{\mu,L}$, whose lower bound on the rate is $\left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)$ which is thereby more often used).

8.1.2 Contributions

In this work, we show that the heavy-ball (HB) cannot attain an accelerated (worst-case) convergence rate on the standard class of smooth strongly convex problems in general. We further show that this result is stable to additional assumptions. In particular, it remains true even under higher-order regularity assumptions, and for perturbed initial conditions and gradient computations.

More precisely, in Section 8.2, we recall a few known results on HB, and set up a few key concepts for building up the following sections, including those of *cycling behaviors*. Next,

in Section 8.3, we show our main result on HB, namely that we cannot obtain accelerated convergence guarantees for HB on standard classes of problems beyond quadratics. To obtain that result, we analyze parameters resulting in a simple cycle shape.

Then, in Section 8.4 details we detail a constructive approach for finding counter-examples to the convergence of HB, and more generally of any *stationary* first-order method. We demonstrate that if a parametrization results in a cycle, then it also results in a cycle having a very specific shape, and that numerically, this can be solved as a linear problem. Then, in Section 8.5 we show that our non-acceleration results are stable to small perturbations of the initial iterates, of the parameters and of the gradients. Finally, in Section 8.6, we also show that adding additional natural regularity assumptions does not result in acceleration either.

8.1.3 Key concepts

The following definition introduces the concept of *asymptotic* convergence rate and of convergence rate of a method over a class of functions. Formally, this definition is necessary because we are looking for negative results on the value of ρ throughout the paper. Furthermore, momentum-type methods such as HB or NAG (contrary to GD) are not monotone method (i.e., $(\|x_t - x_\star\|)_t$ is not a decreasing sequence in general).

Definition 8.1.3 ((Asymptotic) convergence rate)**.** *Let $\mathcal{F}$ be a class of functions. We say that a given first-order method has a worst-case asymptotic convergence rate ρ over $\mathcal{F}$ if for all $\varepsilon > 0$, there exists T_0 such that for all $f \in \mathcal{F}, x_0$ and $T \geqslant T_0$, for $(x_t)_{t\geqslant 0}$ the sequence of iterates generated from x_0 by running the method on the function f, we have*

$$\|x_T - x_\star\| \leqslant (\rho + \varepsilon)^T \|x_0 - x_\star\|.$$

Informally, this means that $\|x_T - x_\star\|$ is (almost) of the order of ρ^T uniformly over the class $\mathcal{F}$. In the sequel, we may refer to the worst-case asymptotic rate as the *rate*. Our interest is to understand the impact of parameters (γ, β) in HB. For clarity, when necessary, we denote $(\text{HB})_{\gamma,\beta}$ the heavy-ball method with coefficients (γ, β), and $(\text{HB})_{\gamma,\beta}(f)$ the heavy-ball method with coefficients (γ, β) applied to the function f. We then introduce the following two definitions.

Definition 8.1.4 (Rate $\rho_{\gamma,\beta}(\mathcal{F})$)**.** *For any class $\mathcal{F}$ and any $(\gamma, \beta) \in \mathbb{R} \times \mathbb{R}$, we denote $\rho_{\gamma,\beta}(\mathcal{F})$ the smallest worst-case asymptotic rate of $(\text{HB})_{\gamma,\beta}$ on $\mathcal{F}$.*

Definition 8.1.5 (Convergence region $\Omega_{\text{cv}}(\mathcal{F})$)**.** *We denote $\Omega_{\text{cv}}(\mathcal{F})$ the set of parameters $(\gamma, \beta) \in \mathbb{R} \times \mathbb{R}$ for which $(\text{HB})_{\gamma,\beta}$ has a worst-case asymptotic convergence rate $\rho_{\gamma,\beta}(\mathcal{F})$ strictly below 1.*

In the sequel, if $(\gamma, \beta) \in \Omega_{\text{cv}}(\mathcal{F})$, we may abusively say that $(\text{HB})_{\gamma,\beta}$ *converges* on $\mathcal{F}$ (instead of "has a worst-case asymptotic convergence rate $\rho_{\gamma,\beta}(\mathcal{F})$ strictly below 1"), and conversely, that $(\text{HB})_{\gamma,\beta}$ *does not converge* if $(\gamma, \beta) \notin \Omega_{\text{cv}}(\mathcal{F})$ (i.e., there exists $f \in \mathcal{F}$ on which $\text{HB}_{\gamma,\beta}$ does not converge).

Before giving preliminary results on (HB), let us recall a few notations.

Notation. For $x \in \mathbb{R}^d$ and $r > 0$, $B(x, r)$ is the Euclidean ball with center x and radius r, and $\|x\|$ the Euclidean norm of x. We denote $\text{convh}(x_i, i \in I)$ the convex hull of a family

of points $(x_i)_{i\in I}$ indexed by a set I. We denote $\mathrm{Int}(A)$ the interior of a set $A \subseteq \mathbb{R}^d$. For vectors $x, y \in \mathbb{R}^d$, we $\langle x, y\rangle = x^\top y$ is the Euclidean inner product.

We denote I_d the identity matrix in dimension d. We denote $\mathcal{S}_K(\mathbb{R})$ the set of symmetric matrices in dimension $K \in \mathbb{N}$, and $\mathcal{S}_K^+(\mathbb{R})$ the set of positive semi-definite matrices. For M, N in $\mathcal{S}_K(\mathbb{R})$, we denote $M \preccurlyeq N$ if $N - M \in \mathcal{S}_K^+(\mathbb{R})$. For a matrices $M, N \in \mathbb{R}^{K\times K'}$, we denote $\langle M, N\rangle = \mathrm{Tr}(M^\top N)$ the standard inner product, with Tr the trace operator. We denote $\|M\|_{\mathrm{op}}$ the operator norm of the matrix M, and $\mathrm{Sp}(M)$ the spectrum of M.

We denote $\mathcal{C}^k(\mathbb{R}^d)$ the set of k times continuously differentiable functions from $\mathbb{R}^d \to \mathbb{R}$. We denote f^* the Fenchel-transform of a function f. Note that all classes of functions considered hereafter belong to the set of closed proper convex functions, and hence satisfy $f = f^{**}$.

Finally, for any integer $K \geqslant 2$, we denote by θ_K the angle $\frac{2\pi}{K}$. By convention, and to avoid unnecessary case disjunctions, for $\beta < 0$, we use the convention $\sqrt{\beta} = \mathrm{NAN}$ (Not A Number), and $\min(a, \mathrm{NAN}) = \max(a, \mathrm{NAN}) = a$.

8.2 Preliminary results on heavy-ball

This section summarizes a few well-known results and open questions regarding the heavy-ball method. We start by describing a link between convergence guarantees and the choice of parameters (γ, β) on the class of quadratic functions $\mathcal{Q}_{\mu,L}$. Those results are well-known nowadays and date back to Polyak (1964). We leverage them in Section 8.4.

8.2.1 Known behavior of the heavy-ball method on quadratics ($\mathcal{Q}_{\mu,L}$)

In this section, we consider a function $f_H \in \mathcal{Q}_{\mu,L}$ parameterized by its Hessian matrix H, i.e., such that $f(x) - f_\star = \frac{1}{2}(x - x_\star)^\top H(x - x_\star)$ with $\mu\mathrm{I} \preccurlyeq H \preccurlyeq L\mathrm{I}$—or equivalently $\mathrm{Sp}(H) \in [\mu, L]$. The heavy-ball update is:

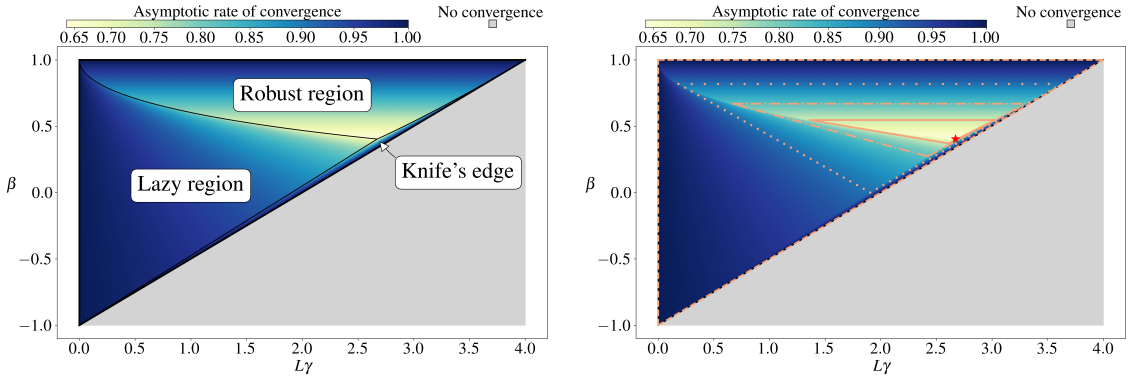
$$x_{t+1} = x_t - \gamma\nabla f(x_t) + \beta(x_t - x_{t-1}) = x_t - \gamma H(x_t - x_\star) + \beta(x_t - x_{t-1}). \tag{HB-Q}$$

The worst-case asymptotic convergence rate (see Definition 8.1.3) of (HB-Q) is provided by the following.

Proposition 8.2.1. (Polyak (1964)) *Consider $\beta \in \mathbb{R}$ and $\gamma \in \mathbb{R}$. The worst-case asymptotic convergence rate $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L})$ of (HB-Q)$_{\gamma,\beta}$, over the class $\mathcal{Q}_{\mu,L}$ is:*

1. ***Lazy region**: If $0 < \gamma \leqslant \min\left(\frac{2(1+\beta)}{L+\mu}, \frac{\left(1-\sqrt{\beta}\right)^2}{\mu}\right)$ then $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L}) = \frac{1+\beta-\mu\gamma}{2} + \sqrt{\left(\frac{1+\beta-\mu\gamma}{2}\right)^2 - \beta}$.*
2. ***Robust region**: If $\beta \geqslant 0$, and $\frac{\left(1-\sqrt{\beta}\right)^2}{\mu} \leqslant \gamma \leqslant \frac{\left(1+\sqrt{\beta}\right)^2}{L}$ then $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L}) = \sqrt{\beta}$.*
3. ***Knife's edge**: If $\max\left(\frac{2(1+\beta)}{L+\mu}, \frac{\left(1+\sqrt{\beta}\right)^2}{L}\right) \leqslant \gamma < \frac{2(1+\beta)}{L}$, then $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L}) = \frac{L\gamma-(1+\beta)}{2} + \sqrt{\left(\frac{L\gamma-(1+\beta)}{2}\right)^2 - \beta}$.*
4. ***No convergence**: if $\frac{\gamma}{1+\beta} \geqslant \frac{2}{L}$ or $\gamma \leqslant 0$ then $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L}) > 1$.*

Sketch of proof. The complete proof is provided in Section 8.A. In short, we rewrite (HB-Q)



(a) Schematic view of the three convergence regions described by Proposition 8.2.1 together with the asymptotic worst-case convergence rate $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L})$ as a color-scale, with respect to γ and β.

(b) Level sets $\mathrm{LS}_{\mu,L}(\rho)$ of $\gamma, \beta \mapsto \rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L})$ as orange triangles. Levels ρ correspond to $\rho = 1$ (- -), $\rho = \frac{1-\kappa}{1+\kappa}$ ($\cdots$) corresponding to the rate of GD, then $\rho = \frac{1-2\kappa}{1+2\kappa}$ (— · —), and $\rho = \frac{1-3\kappa}{1+3\kappa}$ (—), and finally $\rho = \frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}$ (⋆), i.e., $\rho^\star(\mathcal{Q}_{\mu,L})$.

Figure 8.1: Asymptotic convergence rate $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L})$ of (HB-Q), for $\kappa = 1/20$.

as a linear system and decompose it over the eigenspaces of H. Ultimately, the asymptotic convergence rate is given by $\max_{\lambda\in[\mu,L]} \bar{\rho}(P_{\beta,\lambda,\gamma})$ with $P_{\beta,\lambda,\gamma} \triangleq \begin{pmatrix} 1+\beta-\gamma\lambda & -\beta \\ 1 & 0 \end{pmatrix}$. The four cases arise from the nature of the eigenvalues of $P_{\beta,\lambda,\gamma}$, that can either be two complex conjugates number with modulus $\sqrt{\det(P_{\beta,\lambda,\gamma})} = \sqrt{\beta}$ or two real numbers, and the fact the $\max$ may be attained on either μ or L. ■

Comments on Proposition 8.2.1. Figure 8.1b illustrates the asymptotic rate of the heavy-ball method for each value of the parameters γ, β. It shows the three parameter regions resulting in convergence, as given by Proposition 8.2.1. First, the **left** region, called the **lazy region** where the step-size is small, thus the rate is driven by the convergence of the iterates' component aligned with the eigenvector of H associated with μ. Second, the **right** region, called the **knife's edge** where the step-size is large and the rate is driven by the oscillations of the iterates' component aligned with the eigenvector of H associated with L. Third, the **upper** region, called the **robust region** where the step-size does not impact the convergence rate.

In particular, Proposition 8.2.1 enables to define the set of parameters for which (HB-Q) converges. Following Definition 8.1.5, we denote by $\Omega_{\mathrm{cv}}(\mathcal{Q}_{\mu,L})$ the set of parameters (γ, β) for which (HB-Q) has a worst-case asymptotic convergence rate strictly below 1 (i.e., for which HB converges on any function of $\mathcal{Q}_{\mu,L}$). This set is naturally the union of three regions of convergence provided by Proposition 8.2.1.

Corollary 8.2.2. *By Proposition 8.2.1, we have*

$$\Omega_{\mathrm{cv}}(\mathcal{Q}_{\mu,L}) = \left\{(\gamma,\beta) \in \mathbb{R}\times\mathbb{R} \text{ s.t. } \beta \in (-1;1), 0 < \gamma < \frac{2}{L}(1+\beta)\right\}.$$

Furthermore, the optimal parameter choice is achieved at the intersection of the three regions (or equivalently at point of the robust region with the smallest β), as provided by the following result.

Corollary 8.2.3. *The optimal worst-case asymptotic rate of* (HB-Q) *on* $\mathcal{Q}_{\mu,L}$*, for parameters* $(\gamma,\beta)\in\Omega_{\mathrm{cv}}(\mathcal{Q}_{\mu,L})$ *is*

$$\rho^\star(\mathcal{Q}_{\mu,L})\triangleq\min_{(\gamma,\beta)\in\Omega_{\mathrm{cv}}(\mathcal{Q}_{\mu,L})}\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L})=\sqrt{\beta^\star(\mathcal{Q}_{\mu,L})}=\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}},$$

which is achieved for $\beta^\star(\mathcal{Q}_{\mu,L})\triangleq\left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^2$, $\gamma^\star(\mathcal{Q}_{\mu,L})\triangleq\frac{2}{L+\mu}(1+\beta^\star(\mathcal{Q}_{\mu,L}))$.

In a nutshell, on the one hand, for $\beta\leqslant\left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^2$, the optimal rate is achieved for a single value of γ, such that $\gamma=\frac{2}{L+\mu}(1+\beta)$, that corresponds to the limit between the lazy region and the knife's edge. In this region $\beta\in\left[0;\left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^2\right]$, the rate decreases (improves) as β increases. On the other hand, in the robust region, i.e., when $\beta>\left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^2$, the asymptotic rate $\sqrt{\beta}$ is achieved for any $\gamma\in\left[\frac{\left(1-\sqrt{\beta}\right)^2}{\mu},\frac{\left(1+\sqrt{\beta}\right)^2}{L}\right]$ (which allows to use any value in this set). In this region $\beta\in\left[\left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^2;1\right]$, the rate increases (degrades) with β. The optimal rate is thus achieved at the limit, $\beta=\left(\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}\right)^2$.

This asymptotic rate matches the lower complexity bound provided by Nemirovskii and Nesterov (1985). As $\kappa\to0$, $\rho^\star(\mathcal{Q}_{\mu,L})\sim1-2\sqrt{\kappa}$ which is commonly referred to as an accelerated convergence rate, as compared to that of the classical Gradient descent algorithm, obtained with $(\gamma=2/(L+\mu),\beta=0)$, whose rate is $\rho_{\gamma=2/(L+\mu),\beta=0}(\mathcal{Q}_{\mu,L})\sim1-2\kappa$ (as $\kappa\to0$).

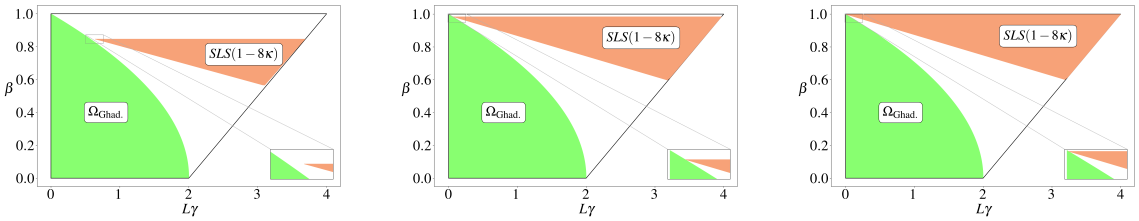
The level sets of the asymptotic convergence rate of (HB) are triangles, as stated in the following lemma and illustrated on Figure 8.1b. This property will be used in the proof of our main result, in Section 8.3.

Lemma 8.2.4 (Sublevel set $\mathrm{SLS}_{\mu,L}(\rho)$ of the heavy-ball convergence rates)**.** *Let* $0<\mu\leqslant L$. *The level sets of* $\gamma,\beta\mapsto\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L})$ *are triangles. More precisely, for any* $\rho\in[\rho^\star(\mathcal{Q}_{\mu,L}),1]$, *the set of parameters* γ,β *for which* $(\mathrm{HB})_{\gamma,\beta}$ *has rate* ρ *is the union of the three segments parametrized by:*

- *Segment in the Lazy Region:* $\beta\in\left[\frac{\frac{1-\kappa}{1+\kappa}-\rho}{\frac{1}{\rho}-\frac{1-\kappa}{1+\kappa}},\rho^2\right]$ *and* $\gamma=\frac{(1-\rho)(1-\beta/\rho)}{\mu}$.
- *Segment in the Robust Region:* $\gamma\in\left[\frac{(1-\rho)^2}{\mu},\frac{(1+\rho)^2}{L}\right]$ *and* $\beta=\rho^2$.
- *Segment in the Knife Edge:* $\beta\in\left[\frac{\frac{1-\kappa}{1+\kappa}-\rho}{\frac{1}{\rho}-\frac{1-\kappa}{1+\kappa}},\rho^2\right]$ *and* $\gamma=\frac{(1+\rho)(1+\beta/\rho)}{L}$.

We denote this level set by $\mathrm{LS}_{\mu,L}(\rho)$ *(which is a triangle), and the corresponding sublevel set (that is extensively used in the sequel) by* $\mathrm{SLS}_{\mu,L}(\rho)=\cup_{\rho'\leqslant\rho}\mathrm{LS}_{\mu,L}(\rho')$.

As a summary, this section provided a complete picture of the behavior of (HB) over $\mathcal{Q}_{\mu,L}$. As the convergence rate for $\beta<0$ is never better than the one for $\beta=0,\gamma=\frac{2}{L+\mu}$, we restrict the analysis to $\beta\geqslant0$ in the following. Next, we move to existing results on the class $\mathcal{F}_{\mu,L}$.



(a) $\kappa = 0.01$ (b) $\kappa = 0.001$ (c) $\kappa = 0.0001$

Figure 8.2: Illustration of Lemma 8.2.5. Region of parameters $\Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L})$ and sublevel set $\text{SLS}_{\mu,L}(1-8\kappa)$ for three values of κ: the rate of $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L})$ on $\Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L})$ is at best $\frac{1-4\kappa}{1+4\kappa}$.

8.2.2 Known behaviors of the heavy-ball method on $\mathcal{F}_{\mu,L}$

Convergence of (HB) on the set of L-smooth and μ-strongly convex functions $\mathcal{F}_{\mu,L}$ has attracted a lot of attention over the last decade. First, recall that we denote $\Omega_{\text{cv}}(\mathcal{F}_{\mu,L})$ the set of parameters (γ,β) for which (HB) has a worst-case asymptotic convergence rate strictly below 1 on $\mathcal{F}_{\mu,L}$ (see Definition 8.1.5).

Convergence results on $\mathcal{F}_{\mu,L}$

Ghadimi et al. (2015) establish that (HB) converges on $\mathcal{F}_{\mu,L}$ when

$$\gamma \in (0, \tfrac{2}{L}) \text{ and } 0 \leqslant \beta < \tfrac{1}{2}\left(\frac{\mu\gamma}{2} + \sqrt{\left(\frac{\mu\gamma}{2}\right)^2 + 4(1-\tfrac{L\gamma}{2})}\right),$$

see (Theorem.4, Ghadimi et al., 2015). For comparison purposes, we denote this set of parameters $\Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L})$ in what follows. Unfortunately, this result does not lead to an acceleration of (HB) on $\mathcal{F}_{\mu,L}$. Indeed, the following lemma shows that the best rate for $(\gamma,\beta) \in \Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L})$ parameters is not accelerated, *even on* $\mathcal{Q}_{\mu,L}$.

Lemma 8.2.5 (Optimal asymptotic rate of (HB-Q) on $\mathcal{Q}_{\mu,L}$ for $(\gamma,\beta) \in \Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L})$)**.** *The optimal worst-case asymptotic rate of* (HB-Q) *on* $\mathcal{Q}_{\mu,L}$*, for parameters* $(\gamma,\beta) \in \Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L})$ *is*

$$\rho^\star_{\text{Ghad.}}(\mathcal{Q}_{\mu,L}) \triangleq \min_{(\gamma,\beta)\in\Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L})} \rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L}) = \sqrt{\beta^\star_{\text{Ghad.}}(\mathcal{Q}_{\mu,L})} \underset{\kappa\to 0}{=} 1 - 8\kappa + o(\kappa),$$

which is achieved for

$$\sqrt{\beta^\star_{\text{Ghad.}}(\mathcal{Q}_{\mu,L})} = (\kappa^{-1}-1)^{1/3}\left[\left(\sqrt{\frac{\kappa^{-1}+26}{27}}+1\right)^{1/3} - \left(\sqrt{\frac{\kappa^{-1}+26}{27}}-1\right)^{1/3}\right] - 1,$$

$$\gamma^\star_{\text{Ghad.}}(\mathcal{Q}_{\mu,L}) = \frac{2(1+\beta^\star_{\text{Ghad.}}(\mathcal{Q}_{\mu,L}))}{L+\mu}.$$

Sketch of proof. The result of Ghadimi et al. (2015) corresponds to using a Lyapunov function of the form $V_t = f(x_t) - f_\star + A(f(x_{t-1}) - f_\star) + B\|x_t - x_{t-1}\|^2$ with $A, B \geqslant 0$. ■

In short, the set $\Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L})$ of parameters covered from (Ghadimi et al., 2015) is not large enough to guarantee acceleration of (HB): indeed for any $C > 8$, there exists κ_0

such that for all $\kappa < \kappa_0$, $\Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L})$ does not intersect the sublevel set $\text{SLS}_{\mu,L}(1 - C\kappa)$ given by Lemma 8.2.4. This is illustrated in Figure 8.2, with $C = 8$ and for 3 different κ. Therefore, a natural question is that of properly identifying the set $\Omega_{\text{cv}}(\mathcal{F}_{\mu,L})$. There exist a few approaches for trying to get better approximations to this set; see, e.g., the work Taylor et al. (2018a) which provides a tool to numerically identify valid Lyapunov functions—see Goujaud et al. (2023a) for a detailed treatment of HB with this technique.

Non-convergence results on $\mathcal{F}_{\mu,L}$

This section summarizes a few negative convergence results for HB on $\mathcal{F}_{\mu,L}$.

Non-convergence for the optimal tuning $(\gamma^\star(\mathcal{Q}_{\mu,L}), \beta^\star(\mathcal{Q}_{\mu,L}))$ on quadratics. Lessard et al. (2016) prove that for the optimal tuning $(\gamma^\star(\mathcal{Q}_{\mu,L}), \beta^\star(\mathcal{Q}_{\mu,L}))$ on $\mathcal{Q}_{\mu,L}$ given by Corollary 8.2.3, for $\kappa = 1/25$, there exist an $L = 25$-smooth and $\mu = 1$-strongly convex function such that if x_0 is in a specific neighborhood, then the iterates generated by (HB) oscillate between the neighborhoods of three values, and thereby never converge towards $x_\star$.

At this stage, it is important to note that this counter-example does not exclude the existence of another tuning (γ, β) for which an accelerated convergence rate is achieved. Indeed, there is no reason for the optimal tuning on $\mathcal{F}_{\mu,L}$ to correspond to that on $\mathcal{Q}_{\mu,L}$.

Non-convergence on multiple tunings. Recently, Goujaud et al. (2023a) proposed a numerical approach to compute cyclic trajectories of various first-order methods that include HB. For (HB), given the period $K \geqslant 2$ of the cycles, this technique consists in solving the following optimization problem which can be cast and solved as an SDP:

$$\left| \begin{array}{ll} \underset{\substack{d \geqslant 1, f \in \mathcal{F}_{\mu,L} \\ (x_t)_t \text{ is generated by (HB)}}}{\text{minimize}} & \|x_0 - x_K\|^2 + \|x_1 - x_{K+1}\|^2 \\ \text{subject to} & \|x_1 - x_0\|^2 \geqslant 1. \end{array} \right. \tag{$\mathcal{P}$}$$

Goujaud et al. (2023a) prove that the value of the optimization problem ($\mathcal{P}$) is 0 if and only if there exists a function $f \in \mathcal{F}_{\mu,L}$ and an initialization $(x_0, x_1) \in (\mathcal{X})^2$ such that the method (HB)(f) initialized at (x_0, x_1) cycles on K values, i.e. that the sequence of iterates generated is $(x_0, \dots, x_{K-1}, x_0, \dots, x_{K-1}, x_0, \dots)$.

Although these negative results are limited to either a single κ and tuning $(\gamma^\star(\mathcal{Q}_{\mu,L}), \beta^\star(\mathcal{Q}_{\mu,L}))$ for (Lessard et al., 2016), or only numerical in (Goujaud et al., 2023a), analyzing the parameter choices for which the worst-case uniform convergence of (HB) on $\mathcal{F}_{\mu,L}$ can be disproved by establishing the existence of a cycle appears to be a promising direction.

8.2.3 Our approach to comprehensive behaviors of heavy-ball

In this section we therefore introduce $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$ as the set of parameter values (γ, β) for which (HB) cycles on a function of $\mathcal{F}_{\mu,L}$.

Definition 8.2.6 (Cycles)**.** *Let $(\gamma, \beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$, and $\mathcal{F}$ a class of functions.*

1. *For K a positive integer, referred to as the period, $(x_t)_{t \in [\![0, K-1]\!]} \neq (x_0, \dots, x_0)$ a family of K points not all-equal, and f a function, we say that*

$$\text{(HB)}_{\gamma,\beta}(f) \text{ cycles on } (x_t)_{t \in [\![0, K-1]\!]}$$

if the sequence $(z_t)_{t\in\mathbb{N}}$ generated by (HB) applied on f with initial points $z_0 = x_0$ and $z_1 = x_1$ cycles on $(x_t)_{t\in[\![0,K-1]\!]}$, i.e., verifies $\forall t \geqslant 0, z_t = x_{t \pmod K}$.

2. *Moreover, for such a K and family of K points $(x_t)_{t\in[\![0,K-1]\!]} \neq (x_0,\dots,x_0)$, we say that*

$$\text{(HB)}_{\gamma,\beta} \text{ cycles on } (x_t)_{t\in[\![0,K-1]\!]} \text{ on } \mathcal{F}$$

if there exists an $f \in \mathcal{F}$, for which $\text{(HB)}_{\gamma,\beta}(f)$ cycles on $(x_t)_{t\in[\![0,K-1]\!]}$.

3. *Finally, we say that*

$$\text{(HB)}_{\gamma,\beta} \text{ has a cycle on } \mathcal{F}$$

if there exist such a period K, and cycle $(x_t)_{t\in[\![0,K-1]\!]} \neq (x_0,\dots,x_0)$, and $f \in \mathcal{F}$, for which $\text{(HB)}_{\gamma,\beta}(f)$ cycles on $(x_t)_{t\in[\![0,K-1]\!]}$.

Note that we exclude the constant cycle $(x_0,\dots,x_0)$, that would correspond to a trivial cycle $(x_\star,\dots,x_\star)$ of $\text{(HB)}(f)$ for any function f such that $x_0 = x_\star = \arg\min f$. This corresponds to non-problematic situations as the algorithm already converged. We underline the following equivalent point of view on a cycle.

Remark 8.2.7. *$\text{(HB)}_{\gamma,\beta}(f)$ cycles on $(x_t)_{t\in[\![0,K-1]\!]}$ if and only if for any $s \in [\![0, K-1]\!]$, $x_{s+1} = x_s - \gamma\nabla f(x_s) + \beta(x_s - x_{s-1})$, where the sequence $(x_t)_t$ is extended K-periodically to $t \in \mathbb{Z}$ as $(x_t)_{t\in\mathbb{Z}} \triangleq (x_{t \pmod K})_{t\in\mathbb{Z}}$ (in particular as $x_K \triangleq x_0$ and $x_{-1} \triangleq x_{K-1}$).*

From Definition 8.2.6, we define the region $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$.

Definition 8.2.8 (Cycling region $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$)**.** *For any $0 < \mu \leqslant L$, we denote:*

1. *$\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$ the subset of $\Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$ for which (HB) has a cycle on $\mathcal{F}_{\mu,L}$.*
2. *$\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})^c$ the complementary of $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$ in $\Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$.*

In the following, we leverage that for any parameters (γ,β) for which $\text{(HB)}_{\gamma,\beta}$ has a cycle on $\mathcal{F}_{\mu,L}$, then the method does not converge.

Fact 8.2.9. *The set of parameters for which (HB) has a worst-case (asymptotic) convergence rate (strictly below 1) on $\mathcal{F}_{\mu,L}$ is included in $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})^c$:*

$$\Omega_{\text{cv}}(\mathcal{F}_{\mu,L}) \subseteq \Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})^c$$

In Section 8.3, we demonstrate that (HB) cannot accelerate by focusing on a particular cycle shape. We study the set of parameters such that there exists a function in $\mathcal{F}_{\mu,L}$ that cycles over that particular set of iterates. Section 8.4 explains why such a choice of a cycle is in fact natural.

8.3 Non-acceleration of heavy-ball on $\mathcal{F}_{\mu,L}$ via simple two-dimensional cycles

In this section, we demonstrate the main result of the paper, which is that HB method does not accelerate on the class $\mathcal{F}_{\mu,L}$. To obtain this result, we introduce in Subsection 8.3.1 a simple two-dimensional cycle of length K and study the set of (γ,β) such that there exists a function in $\mathcal{F}_{\mu,L}$ that cycles over those specific iterates. Then, in Subsection 8.3.2, for some appropriate $C > 0$, we show that the sublevel set of level $1 - C\kappa$ of (HB) on the set of quadratic function is excluded from the parameters that do not have such a cycle.

8.3.1 Studying a specific type of cycling behavior

We focus on the cycles that are supported by the K-th roots of unity.

Definition 8.3.1 (Roots-of-unity cycle)**.** *For $K \in \mathbb{N}$, and $\theta_K \triangleq \frac{2\pi}{K}$, we define the roots-of-unit cycle as*

$$\bigcirc_K = (x_0^\circ, x_1^\circ, \dots, x_t^\circ \dots, x_{K-1}^\circ) \triangleq \left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} \cos\theta_K \\ \sin\theta_K \end{pmatrix}, \dots, \begin{pmatrix} \cos t\theta_K \\ \sin t\theta_K \end{pmatrix}, \dots, \begin{pmatrix} \cos(K-1)\theta_K \\ \sin(K-1)\theta_K \end{pmatrix} \right).$$

We introduce the rotation operator $R = \begin{pmatrix} \cos\theta_K & -\sin\theta_K \\ \sin\theta_K & \cos\theta_K \end{pmatrix}$ such that for any $t \in [\![1; K-1]\!]$, $x_t^\circ = Rx_{t-1}^\circ = R^t x_0^\circ$ and $R^K = \mathrm{I}$.

This corresponds to a completely symmetrical cycle shape. Such a cycle is pictured in Figure 8.3. We now introduce the set of parameters (γ, β) for which (HB) results in such a cycle on at least one function in $\mathcal{F}_{\mu,L}$.

Definition 8.3.2 (Roots-of-unity cycling region $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$)**.** *For any $0 < \mu \leqslant L$, we define*

1. *for any $K \in \mathbb{N}$, $\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ the subset of $\Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$ for which (HB) cycles on $\bigcirc_K$ on $\mathcal{F}_{\mu,L}$ (in the sense of Definition 8.2.6, item 2).*
2. $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}) = \bigcup_{K=2}^{\infty} \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$.
3. $(\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}))^c = \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}) \backslash \Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ *the complementary of $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ in $\Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$.*

In other words, $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ is a subset of $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$ for which we limit the cycles to be (i) in dimension $d = 2$ (there was no restriction on the dimension earlier on) and (ii) with a specific shape (cycling over the roots of unity). The fact that limiting ourselves to $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ is a reasonable restriction will be discussed in Section 8.4. For clarity, notations of the various regions are summarized in Table 8.1. Putting things together and leveraging Fact 8.2.9, we have the following fact.

Fact 8.3.3. *For any $0, \mu \leqslant L$:*

$$\Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L}) \subseteq \Omega_{\text{cv}}(\mathcal{F}_{\mu,L}) \subseteq (\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L}))^c \subseteq (\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}))^c \subseteq \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}).$$

Table 8.1: Summary of the parameter regions for which convergence is established or disproved.

Notation	Region
$\Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$	Convergence on $\mathcal{Q}_{\mu,L}$.
$\Omega_{\text{cv}}(\mathcal{F}_{\mu,L})$	Convergence on $\mathcal{F}_{\mu,L}$
$\Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L})$	Convergence is established by Ghadimi et al. (2015)
$\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$	Subset of $\Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$ where (HB) has a cycle on $\mathcal{F}_{\mu,L}$
$\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$	Subset of $\Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$ where (HB) has a roots-of-unity cycle.

Next, we observe that (HB)(f) cycles over $\bigcirc_K$, if and only if we have a simple expression for $(\nabla f(x_t^\circ))_{t\in[\![0,K-1]\!]}$. Indeed, given the iterates in (HB)(f), the value of the gradients to obtain those iterates are uniquely obtained.

Lemma 8.3.4. *Let $K \geqslant 2$ an integer and $\theta_K \triangleq \frac{2\pi}{K}$. Let $(x_t^\circ)_{t\in[\![0,K-1]\!]} = \bigcirc_K$ be the roots-of-unity cycle of length K. Let $\gamma, \beta \in \Omega_{\mathrm{cv}}(\mathcal{Q}_{\mu,L})$. For any differentiable function f, $\mathrm{HB}_{\gamma,\beta}(f)$ cycles on $\bigcirc_K$ if and only if*

$$\forall t \in [\![0, K-1]\!],\ \nabla f(x_t^\circ) = g_t \triangleq \frac{(1+\beta)I_2 - R - \beta R^{-1}}{\gamma} x_t^\circ. \tag{8.1}$$

Proof. Let f any differentiable function. By Definition 8.2.6 and Remark 8.2.7, $(\mathrm{HB})_{\gamma,\beta}(f)$ cycles on $(x_t^\circ)_{t\in[\![0,K-1]\!]}$ if and only if for any $t \in [\![0, K-1]\!]$, $x_{t+1}^\circ = x_t^\circ - \gamma\nabla f(x_t^\circ) + \beta(x_t^\circ - x_{t-1}^\circ)$, with x_t° extended K-periodically (i.e., $x_{-1}^\circ \triangleq x_{K-1}^\circ$ and $x_K^\circ \triangleq x_0^\circ$). Since $\gamma \neq 0$, this system is equivalently written as, for any $t \in [\![0, K-1]\!]$,

$$\nabla f(x_t^\circ) = g_t \triangleq \frac{(1+\beta)x_t^\circ - x_{t+1}^\circ - \beta x_{t-1}^\circ}{\gamma}. \tag{8.2}$$

Replacing the expressions $x_{t+1}^\circ = Rx_t^\circ$ and $x_{t-1}^\circ = R^{-1}x_t^\circ$, we obtain the desired result. ■

The values of the gradients at points $(x_t^\circ)_{t\in[\![0,K-1]\!]}$ are depicted as red arrows on Figure 8.3. We now use Lemma 8.3.4 to obtain an analytical form of $\Omega_{\circ\text{-Cycle}}(\mathcal{F}_{\mu,L})$.

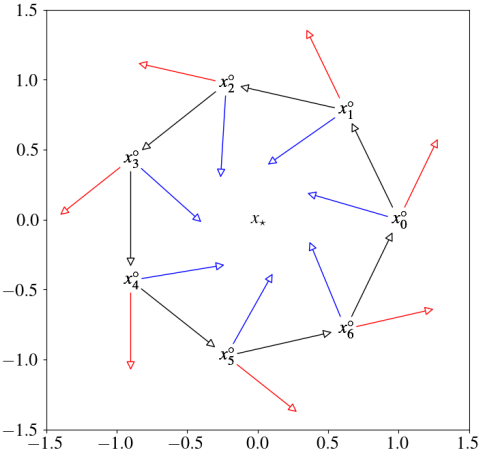


Figure 8.3: Cycle $\bigcirc_7 = (x_0^\circ, \dots, x_6^\circ)$ of the $K = 7^{\text{th}}$-roots-of-unity. For $(\gamma, \beta) \in \Omega_{7\text{-}\circ\text{-Cycle}}(\mathcal{F}_{\mu,L})$, the red arrows (→) correspond momentum component of $(\mathrm{HB})_{\gamma,\beta}$ and the blue arrows (→) to the gradients of $\psi^K_{\gamma,\beta,\mu,L}$ such the $(\mathrm{HB})_{\gamma,\beta}(\psi^K_{\gamma,\beta,\mu,L})$ cycles over $\bigcirc_7$. Here, $L = 1$, $\mu = 0.005$, $\gamma = 3.5$ and $\beta = 0.75$.

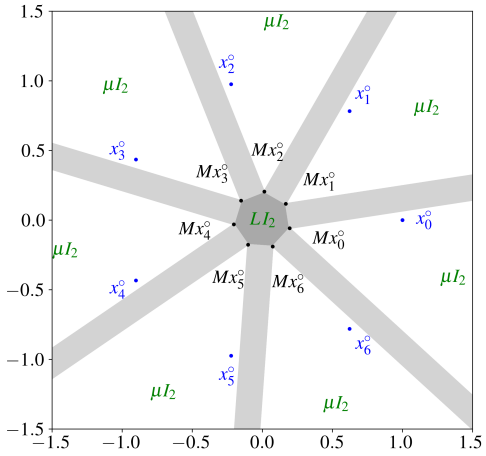


Figure 8.4: Shape of the counter-example function $\psi^K_{\gamma,\beta,\mu,L}$ given by (8.3), for $(\mu, L) = (0.005, 1)$ and $(\gamma, \beta) = (3.3, 0.75) \in \Omega_{7\text{-}\circ\text{-Cycle}}(\mathcal{F}_{\mu,L})$. The function is locally quadratic, with Hessian LI_2 inside $\mathrm{convh}\{Mx_t^\circ, t \in [\![0, 6]\!]\}$ (gray background), μI_2 in the white-background around $\bigcirc_7$ and quadratic with Hessian spectrum $\{\mu, L\}$ in the light gray area.

Theorem 8.3.5. (Analytical form of Roots-of-unity cycle region) *For any $K \geqslant 2$, the K^{th}-roots-of-unity cycling region is, for $\theta_K = \frac{2\pi}{K}$:*

$$\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) = \Big\{ (\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}) \mid (\mu\gamma)^2 - 2\left[\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)\right](\mu\gamma) + 2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K) \leqslant 0. \Big\}$$

Moreover for any $K \geqslant 2$, and any $(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$,

$$\psi^K_{\gamma,\beta,\mu,L} : x \mapsto \frac{L}{2}\|x\|^2 - \frac{L-\mu}{2} d(x, \operatorname{convh}\{Mx_t^\circ, t \in [\![0, K-1]\!]\})^2 \tag{8.3}$$

is a function such that (HB)$_{\gamma,\beta}(\psi^K_{\gamma,\beta,\mu,L})$ *cycles on* $\bigcirc_K$*, with M the linear operator $M \triangleq \frac{(1+\beta-\mu\gamma)I_2 - R - \beta R^{-1}}{(L-\mu)\gamma}$.*

For given K, μ, L, Theorem 8.3.5 provides a second-order equation on (γ,β), such that (HB)$_{\gamma,\beta}$ cycles over $\bigcirc_K$ on $\mathcal{F}_{\mu,L}$. We use this formula to plot the regions $\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ in Figure 8.5, for increasing cycle length K, for two values of κ. Equation (8.3) gives an explicit formula for the function that realizes the cycle: this function is a quadratic by part: its shape is described in Figure 8.4.

Sketch of proof. By Lemma 8.3.4, $(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ if and only if, there exists a function $f \in \mathcal{F}_{\mu,L}$ such that (8.1) holds. Establishing the *existence* of a function in the class $\mathcal{F}_{\mu,L}$ having specific gradient values at a finite number of specific points can be cast as verifying a finite number of simple inequalities. Those conditions, often referred to as *interpolation conditions* (see, e.g., (Taylor et al., 2017c)) and come along with a systematic construction of the given function as a Moreau envelope (similar in spirit with the proof of Taylor et al. (2017c, Theorem 4)). The complete proof is given in Subsection 8.B.1. ■

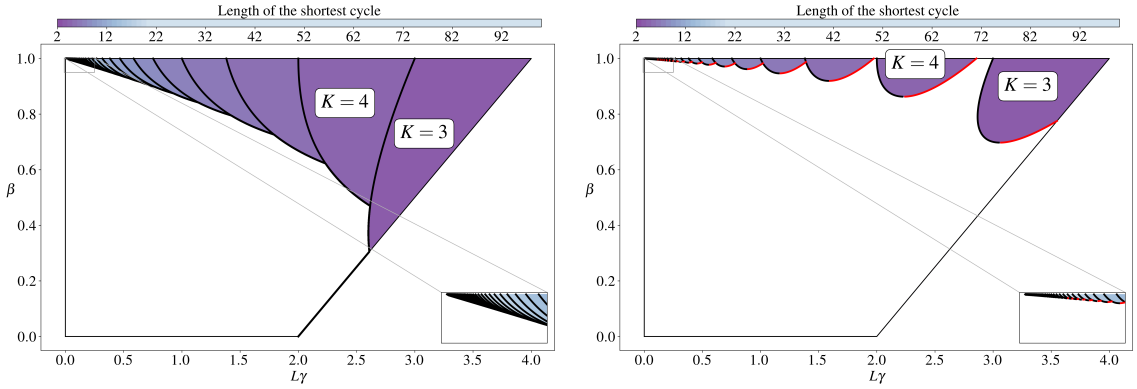
Theorem 8.3.5 shows that, (γ,β) is in $\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ if and only if

$$\gamma \in [\gamma_-(\beta,K,\mu,L), \gamma_+(\beta,K,\mu,L)]\,, \tag{8.4}$$

with $(\gamma_-(\beta,K,\mu,L), \gamma_+(\beta,K,\mu,L))$ obtained as the roots of the second order polynomial given in Theorem 8.3.5, i.e., $\gamma \mapsto (\mu\gamma)^2 - 2\left[\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)\right](\mu\gamma) + 2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K)$ – and the set is empty if this polynomial is always positive. Hence, for any β, the set of all γ such that $(\gamma,\beta) \in \Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ is the union (over $K \geqslant 2$) of intervals given by (8.4), that are not necessarily connected. In the proof of the next result, we will rely on the fact that for $\kappa \leqslant \left(\frac{3-\sqrt{5}}{4}\right)^2$, this union actually is a *unique* interval of the form $[\gamma_{\min}(\beta,\mu,L), \frac{2(1+\beta)}{L}]$, as illustrated in Figure 8.5a. For larger values of κ, the union is not a single interval, which can be expected as the region shrinks as κ approaches 1. Such a behavior is illustrated in Figure 8.5b. However, large values of κ are not problematic as for those, the difference between $\sqrt{\kappa}$ and κ is not significant. This is made formal in Theorem 8.B.4, stated in Subsection 8.B.2 is essential for the next section.

8.3.2 Non-acceleration on $\mathcal{F}_{\mu,L}$

In this section, we finally obtain the main non-acceleration result of the paper, by comparing $\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ to the sublevel sets of the convergence rate of (HB) on $\mathcal{Q}_{\mu,L}$.



(a) $\kappa = 0.01$. Typical region shape for small κ (b) $\kappa = 0.7$. Typical region shape shape for $\kappa \simeq 1$.

Figure 8.5: Regions $\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ for increasing values of K between 2 and 100. The limit of the regions is obtained from the analytical formula given by Theorem 8.3.5.

Theorem 8.3.6. *There exists an absolute constant $C > 0$ (any $C > \frac{50}{3}$), such that for any $0 < \mu < L$, we have:*

$$(\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}))^c \cap \text{SLS}_{\mu,L}\left(\frac{1-C\kappa}{1+C\kappa}\right) = \emptyset. \tag{8.5}$$

Sketch of proof. The complete proof is provided in Subsection 8.B.3. In short, we show that, if $(\gamma,\beta) \in (\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}))^c$, then using Theorem 8.B.4 for any $\kappa \leqslant \left(\frac{3-\sqrt{5}}{4}\right)^2$, necessarily, $\mu < C\kappa(1-\beta)$ for any constant $C > \frac{50}{3}$ which is excluded from $\text{SLS}_{\mu,L}\left(\frac{1-C\kappa}{1+C\kappa}\right)$ by Lemma 8.2.4. For $\kappa \geqslant \left(\frac{3-\sqrt{5}}{4}\right)^2$, we have $\sqrt{\kappa} \leqslant (3+\sqrt{5})\kappa \leqslant C\kappa$ for any $C > \frac{50}{3}$, hence the result. ■

Theorem 8.3.6 is illustrated on Figure 8.6: we represent, for three values of κ in decreasing order, the set $(\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}))^c$ and the set $\text{SLS}_{\mu,L}\left(\frac{1-C\kappa}{1+C\kappa}\right)$. This means that for any (γ,β) such that (HB) does not cycle over a roots-of-unity cycle on $\mathcal{F}_{\mu,L}$, the asymptotic convergence rate of (HB-Q) over $\mathcal{Q}_{\mu,L}$, is worse (i.e., larger) than $\frac{1-C\kappa}{1+C\kappa}$. Formally, we have the following corollary.

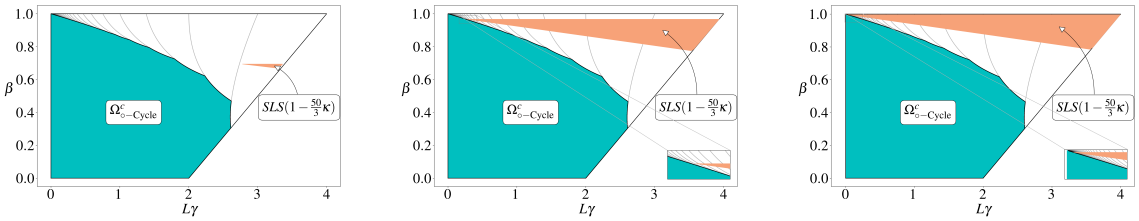
Corollary 8.3.7. (HB) *does not accelerate over on the class $(\mathcal{F}_{\mu,L})_{0<\mu<L}$.*

More precisely, there exists a constant C such that for all $0 < \mu < L$, for all $(\gamma,\beta) \in \mathbb{R}\times\mathbb{R}$ the worst-case asymptotic convergence rate (HB) *over $\mathcal{F}_{\mu,L}$ is lower bounded by $\frac{1-C\kappa}{1+C\kappa}$:*

$$\forall 0 < \mu < L, \rho^\star(\mathcal{F}_{\mu,L}) \triangleq \min_{(\gamma,\beta)\in\mathbb{R}\times\mathbb{R}} \rho_{\gamma,\beta}(\mathcal{F}_{\mu,L}) \geqslant \frac{1-C\kappa}{1+C\kappa},$$

Proof. [Corollary 8.3.7] Let $(\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$. Possibilities are twofold:

- if $(\gamma,\beta) \in \text{SLS}_{\mu,L}\left(\frac{1-C\kappa}{1+C\kappa}\right)$, we know from Theorem 8.3.6, that $(\gamma,\beta) \in \text{SLS}_{\mu,L}\left(\frac{1-C\kappa}{1+C\kappa}\right) \subseteq \Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}) \subseteq \Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L}) \subseteq \Omega_{\text{cv}}(\mathcal{F}_{\mu,L})^c$, i.e. there exists a function $f \in \mathcal{F}_{\mu,L}$ such that $(\text{HB})_{\gamma,\beta}(f)$ does not converge.
- if $(\gamma,\beta) \in \text{SLS}_{\mu,L}\left(\frac{1-C\kappa}{1+C\kappa}\right)^c$, then by definition, there exists a function $f \in \mathcal{Q}_{\mu,L} \subseteq \mathcal{F}_{\mu,L}$ such that $(\text{HB})_{\gamma,\beta}(f)$ achieves an asymptotic rate larger that $\frac{1-C\kappa}{1+C\kappa}$.



(a) $\kappa = 0.01$ (b) $\kappa = 0.001$ (c) $\kappa = 0.0001$

Figure 8.6: Illustration of the incompatibility result Theorem 8.3.6. For three values of κ, we represent the sublevel set $\text{SLS}_{\mu,L}\left(\frac{1-C\kappa}{1+C\kappa}\right)$ and the region $(\Omega_{\circ\text{-Cycle}}(\mathcal{F}_{\mu,L}))^c$ and notice that their intersections are empty.

■

This result concludes the first part of our study of (HB), as Corollary 8.3.7 closes a long-standing open question on the behavior of (HB).

In the next section, we provide an in-depth analysis of the structure of the cycles, beyond dimension 2, that supports the seemingly arbitrary choice of roots-of-unity cycles made in this section.

8.4 General study of cycles for stationary first-order methods

In this section, we investigate the set $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$ of (γ, β) for which $(\text{HB})_{\gamma,\beta}$ has a cycle (in the sense of Definition 8.2.6), without specifically focusing on roots-of-unity cycles $\bigcirc_K$. We explain why this particular cycle shape, that led to the main result in Section 8.3, arises as a natural candidate when studying cycles for stationary first-order methods. Informally, we show that: *if* $(\text{HB})_{\gamma,\beta}$ *has a cycle, then* $(\text{HB})_{\gamma,\beta}$ *has a symmetric cycle*.

Remark 8.4.1. *The arguments made in this section directly apply to any* stationary *first-order method, that is, to any method whose dynamic does not change along the iterations (see Definition 2.1, Goujaud et al., 2023a). For simplicity and coherence, we only instantiate the construction on* (HB).

Our approach is decomposed into three steps. First, in Subsection 8.4.1, we show that the existence of a cycle can be cast as a SDP: to that end, we follow the classical *performance estimation* approach (Drori and Teboulle, 2014; Taylor et al., 2017c). Second, in Subsection 8.4.2, we leverage the convexity of the problem in its SDP form and the structure of the cycle-existence problem to obtain a solution that admits particular symmetries: first on the space of Gram matrices in $\mathcal{S}^+_K$ in Subsection 8.4.2, then to decompose the cycle onto low-dimensional subspaces in Subsection 8.4.2, and ultimately, to rewrite the problem of finding a cycle as a linear problem in Subsection 8.4.2. Finally, in Subsection 8.4.3, we establish numerically that, for (HB), the set of parameters for which there exists a cycle in dimension 2 is the same as the one for which there exists any cycle.

8.4.1 Casting the existence of a cycle as a convex feasibility problem

Let $(\gamma, \beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$. In this section we approach cycles from an optimization point of view, similar in spirit with Goujaud et al. (2023a), see ($\mathcal{P}$). However, we here choose to

directly write the existence of a cycle with period K as the following feasibility problem:

$$\exists (x_0, \dots, x_{K-1}) \neq (\bar{x}, \dots, \bar{x}),\ \exists f \in \mathcal{F}_{\mu,L} |\ \mathrm{HB}_{\gamma,\beta}(f) \text{ cycles on } (x_t)_{0 \leqslant t \leqslant K-1}, \tag{$\mathcal{P}'_K$}$$

where $\bar{x} \triangleq \sum_{i=0}^{K-1} x_i/K$ is used in place of x_0 to avoid constant cycles, while preserving symmetry. By Definition 8.2.6 and Definition 8.2.8, $(\gamma, \beta) \in \Omega_{\mathrm{Cycle}}(\mathcal{F}_{\mu,L})$ if and only if there exists $K \geqslant 2$ such that ($\mathcal{P}'_K$) holds.

We now fix $(\gamma, \beta) \in \Omega_{\mathrm{cv}}(\mathcal{Q}_{\mu,L})$, and $K \geqslant 2$ in the rest of the section. We denote $\mathcal{S}_K^+(\mathbb{R})$ the cone of symmetric positive semi-definite (p.s.d.) matrices. We prove the following result.

Theorem 8.4.2 (Cycle as an SDP)**.** *The feasibility problem ($\mathcal{P}'_K$) is equivalent to the following feasibility problem:*

$$\exists G \in \mathcal{S}_K^+(\mathbb{R}), G \neq \mathbf{0}_K, G\mathbf{1}_K = \mathbf{0}_K,\ \exists F \in \mathbb{R}^K \mid \forall i, j \in [\![0, K-1]\!], \langle F, e_i - e_j \rangle \geqslant \langle G, M_{i,j} \rangle \tag{$\mathcal{P}_{K-\mathrm{SDP}}$}$$

where (e_i) corresponds to the $(i+1)^{th}$ canonical vector in $\mathbb{R}^K$ and the the matrices $M_{i,j}$ are fixed.

In words, the feasibility problem is equivalent to the existence of a p.s.d. matrix G and a vector F satisfying a list of linear inequalities, which is usually referred to as a semi-definite program (SDP) (Vandenberghe and Boyd, 1996). The proof, which is given below, is decomposed into three steps. In short, first, we give a necessary and sufficient condition on the gradients of f for (HB)$_{\gamma,\beta}(f)$ to cycle over a $(x_0, \dots x_{K-1})$. Second, we characterize by a list of inequalities the *existence* of a function f in the class $\mathcal{F}_{\mu,L}$ that admits those specific gradients. Those inequalities are referred to as *interpolation conditions*. Third, we rewrite the feasibility problem in terms of the p.s.d. Gram matrix of the translated iterates $(x_0 - \bar{x}, \dots x_{K-1} - \bar{x})$ and the vector of function values $(f(x_0), \dots f(x_{K-1}))$: all constraints are then linear, and the problem writes as an SDP.

Remark 8.4.3 (Link with performance estimation problems (PEPs))**.** *This approach is essentially the one systematically used in performance estimation (see, e.g. (Drori and Teboulle, 2014; Taylor et al., 2017a,c) for details), to cast the derivation of worst-case guarantees of first-order optimization methods as SDPs. As proposed by (Goujaud et al., 2023a), formulating and solving ($\mathcal{P}$) through SDP formulations can be done numerically with appropriate PEP software (Goujaud et al., 2022a; Taylor et al., 2017b) and SDP software (MOSEK, 2019).*

Proof. We fix $(\gamma, \beta) \in \Omega_{\mathrm{cv}}(\mathcal{Q}_{\mu,L})$, and $K \geqslant 2$ in the proof. We first observe that for any cycle under consideration $(x_0, \dots, x_{K-1})$, and any function $f \in \mathcal{F}_{\mu,L}$, $\mathrm{HB}_{\gamma,\beta}(f)$ cycles on $(x_t)_{0 \leqslant t \leqslant K-1}$ if an only if, for any $t \in [\![0; K-1]\!]$

$$\nabla f(x_t) = \frac{(1+\beta)x_t - x_{t+1} - \beta x_{t-1}}{\gamma}, \tag{8.6}$$

where the sequence $(x_t)_t$ is extended to $t \in \mathbb{Z}$ in this proof by K-periodicity as $x_t \triangleq x_{t \pmod K}$ (in fact, (8.6) only requires to introduce x_{-1} and x_K). This directly follows from inverting (HB) recursion to obtain the unique value of the gradients that result in a particular cycle.[1]

[1] This derivation generalizes the one of (8.2) in the proof of Lemma 8.3.4 to any cycle.

Therefore, for $(x_0, \dots, x_{K-1}) \neq (\bar{x}, \dots, \bar{x})$ there exists a function $f \in \mathcal{F}_{\mu,L}$ such that (HB)$_{\gamma,\beta}(f)$ cycles over $(x_0, \dots, x_{K-1})$ if and only if there exists a function $f \in \mathcal{F}_{\mu,L}$ verifying (8.6). This problem is known as *interpolation problem* of the class $\mathcal{F}_{\mu,L}$ and a necessary and sufficient condition is given by (Taylor et al., 2017c, Theorem 4), which is recalled below.

Lemma 8.4.4. *[$\mathcal{F}_{\mu,L}$-interpolation, see (Taylor et al., 2017c)] Let $\mathcal{I}$ a set of indices and $(x_i, g_i, f_i)_{i\in\mathcal{I}}$ a family of triplets. There exists a function $f \in \mathcal{F}_{\mu,L}$ verifying $\forall i \in \mathcal{I},\ f(x_i) = f_i$, and $\nabla f(x_i) = g_i$, if and only*

$$\forall i, j \in \mathcal{I},\ f_i - f_j \geqslant \langle g_j, x_i - x_j\rangle + \frac{1}{2L}\|g_i - g_j\|^2 + \frac{\mu}{2(1-\kappa)}\|x_i - \tfrac{1}{L}g_i - x_j + \tfrac{1}{L}g_j\|^2.$$

Consequently, using (8.6) in Lemma 8.4.4, we obtain that ($\mathcal{P}'_K$) is equivalent to

$$\begin{aligned}
&\exists (x_0, \dots, x_{K-1}) \neq (\bar{x}, \dots, \bar{x}),\ \exists (f_0, \dots, f_{K-1}) \mid \forall i, j \in [\![0, K-1]\!],\\
&\quad f_i - f_j \geqslant \left\langle \frac{(1+\beta)x_j - x_{j+1} - \beta x_{j-1}}{\gamma}, x_i - x_j \right\rangle \qquad (\mathrm{IC}_{i,j})\\
&\qquad + \frac{1}{2L}\left\| \frac{(1+\beta)x_i - x_{i+1} - \beta x_{i-1}}{\gamma} - \frac{(1+\beta)x_j - x_{j+1} - \beta x_{j-1}}{\gamma} \right\|^2\\
&\qquad + \frac{\mu}{2(1-\kappa)}\left\| \frac{(1+\beta-L\gamma)x_i - x_{i+1} - \beta x_{i-1}}{L\gamma} - \frac{(1+\beta-L\gamma)x_j - x_{j+1} - \beta x_{j-1}}{L\gamma} \right\|^2.
\end{aligned}$$

Under this form, this feasibility problem is not convex due to quadratic terms in $(x_t)_{0\leqslant t\leqslant K-1}$. However, all terms involving $(x_t)_{0\leqslant t\leqslant K-1}$ are exactly quadratic. We therefore introduce the Gram matrix G of vectors $(x_t - \bar{x})_{0\leqslant t\leqslant K-1}$, and the vector $F = (f_0, \dots, f_{K-1})^T$:

$$G = \Big(x_0 - \bar{x}, \dots, x_{K-1} - \bar{x}\Big)^T \Big(x_0 - \bar{x}, \dots, x_{K-1} - \bar{x}\Big) = \big(\langle x_i - \bar{x}, x_j - \bar{x}\rangle\big)_{0\leqslant i,j\leqslant K-1}.$$

The matrix G is symmetric positive semi-definite. Moreover, for any (i, j), (1) the right hand side of ($\mathrm{IC}_{i,j}$) can be written as $\langle G, M_{i,j}\rangle$, that is, as a linear combination of the coefficients of G, for a matrix M_{ij} obtained from ($\mathrm{IC}_{i,j}$); and (2) the left hand side of ($\mathrm{IC}_{i,j}$) can be written as $\langle F, e_i - e_j\rangle$, where e_i denotes the $(i+1)^{\text{th}}$ vector of the canonical basis. This method, referred to as *SDP lifting* thus linearizes the above problem as

$$\exists G \in \mathcal{S}_K^+(\mathbb{R}), G \neq \mathbf{0}_K, G\mathbf{1}_K = \mathbf{0}_K,\ \exists F \in \mathbb{R}^K \mid \forall i, j \in [\![0, K-1]\!], \langle F, e_i - e_j\rangle \geqslant \langle G, M_{i,j}\rangle$$

for some matrices $(M_{i,j})_{i,j}$ independent of the variables of the problem (in particular independent of the cycle itself). Note that the constraint $G \neq \mathbf{0}_K$ comes from the condition $(x_0, \dots, x_{K-1}) \neq (\bar{x}, \dots, \bar{x})$ and the constraint $G\mathbf{1}_K = \mathbf{0}_K$ comes from the fact that $\frac{1}{K}\sum_{k=0}^{K-1}(x_k - \bar{x}) = 0$. Overall, this corresponds to the feasibility problem given as ($\mathcal{P}_{K-\mathrm{SDP}}$).

■

This proof also provides a slightly stronger result, that will be leveraged in the following.

Theorem 8.4.5. *The feasibility problem ($\mathcal{P}'_K$) is equivalent to ($\mathcal{P}_{K-\mathrm{SDP}}$). Moreover,*

1. *a) For any solution $(F, G) \in \mathbb{R}^K \times \mathcal{S}_K^+(\mathbb{R})$ of ($\mathcal{P}_{K-\mathrm{SDP}}$), there exist points $(x_0, \dots, x_{K-1})$ in dimension at most $K-1$ such that G is the Gram matrix of $(x_0, \dots, x_{K-1})$.*
   b) Moreover for all such $(x_0, \dots, x_{K-1})$, there exists a function $f \in \mathcal{F}_{\mu,L}$ such that (HB)$_{\gamma,\beta}(f)$ cycles over $(x_0, \dots, x_{K-1})$, $F = (f(x_k))_{k\in[\![0,K-1]\!]}$ and G is the Gram matrix of the vectors $(x_0, \dots, x_{K-1})$.

2. *For any points* $(x_0, \dots, x_{K-1})$ *and function* $f \in \mathcal{F}_{\mu,L}$ *solution of* $(\mathcal{P}'_K)$ *then with* $F = (f(x_k))_{k\in[\![0,K-1]\!]}$ *and* G *the Gram matrix of the vectors* $(x_0, \dots, x_{K-1})$*, we have that* (F, G) *is a solution of* $(\mathcal{P}_{K-\mathrm{SDP}})$*,*

This results thus links the solution of the two problems.

8.4.2 Building a symmetric feasible point from a given feasible point

Circulant solution to $(\mathcal{P}_{K-\mathrm{SDP}})$

In this section, we leverage simultaneously the existence of a cycle under the SDP form given by Theorem 8.4.2 and the initial form of the problem as the existence of a cycle for a *stationary* first-order method – which is the case for (HB). We consider K, γ, β to be fixed in what follows.

In short, the proof builds upon the intuition that all iterates within the cycle play a symmetric role. As a consequence, from a given cycle $\mathcal{C}_0 = (x_0, x_1, \dots, x_{K-1})$ with G_0 the Gram matrix of $(x_k)_{k\in[\![0,K-1]\!]}$, we have access to $K-1$ other cycles $\mathcal{C}_1 \triangleq (x_1, x_2, \dots, x_{K-1}, x_0)$, $\dots, \mathcal{C}_s = (x_s, x_{s+1} \dots, x_{s-1}), \dots$, for which the Gram matrix $(G_s)_{s\in[\![0,K-1]\!]}$ is obtained by applying a circular permutation to the rows and columns of G_0. We then average $G_0, \dots, G_{K-1}$: $\bar{G} = \frac{1}{K}\sum_{s=0}^{K-1} G_s$ is a solution to the problem, and a circulant matrix.

Definition 8.4.6 (Circulant matrix)**.** *We denote*

$$J_K \triangleq (\delta_{i+1-j \pmod K})_{1\leqslant i,j\leqslant K} = \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & & & & 1 \\ 1 & 0 & 0 & \dots & 0 \end{pmatrix}.$$

A matrix M *of dimension* $(K \times K)$ *is said to be circulant if it is equal to a polynomial in* J_K*, i.e. there exist* $(c_0, c_1, \dots, c_{d-1})$ *such that*

$$M = c_0 I_d + c_1 J + \dots + c_{K-1} J^{K-1} = \begin{pmatrix} c_0 & c_1 & c_2 & \dots & c_{K-1} \\ c_{K-1} & c_0 & c_1 & & c_{K-2} \\ c_{K-2} & c_{K-1} & c_0 & & c_{K-3} \\ \vdots & & & \ddots & \vdots \\ c_1 & c_2 & c_3 & \dots & c_0 \end{pmatrix}.$$

As Gram matrices are also symmetric, we will have an additional constraint that $c_1 = c_{K-1}$, $c_2 = c_{K-2}$, etc. We establish the following result describing symmetric solutions to $(\mathcal{P}_{K-\mathrm{SDP}})$.

Theorem 8.4.7 (Symmetries of the cycle)**.** *If* $(\mathcal{P}_{K-\mathrm{SDP}})$ *admits a solution, then* $(\mathcal{P}_{K-\mathrm{SDP}})$ *admits a solution* $(\bar{F}, \bar{G})$ *with* $\bar{F} = \mathbf{0}_K$ *and* $\bar{G}$ *a (symmetric) circulant matrix.*

In short, from a solution to $(\mathcal{P}_{K-\mathrm{SDP}})$, we build $K-1$ other solutions by performing a circular permutation of the elements of the cycle. This corresponds to applying a circular permutation matrix to the Gram matrix of the iterates. We can then *average* the K solutions of the problem: by convexity of the set of solutions of the SDP, the resulting Gram matrix is still a solution to the problem. We make this argument precise in the following proof.

Proof. Assume there exists a solution $(F_0, G_0) \in \mathbb{R}^K \times \mathcal{S}_K^+(\mathbb{R})$ of $(\mathcal{P}_{K-\mathrm{SDP}})$, i.e., that (F_0, G_0)

are such that $G_0 \neq 0_K$, $G\mathbf{1}_K = 0_K$ and for all $i, j \in [\![0, K-1]\!]$, $\langle F_0, e_i - e_j\rangle \geqslant \langle G_0, M_{i,j}\rangle$. From Theorem 8.4.5-1 there exists a cycle $\mathcal{C}_0 = (x_0, \dots, x_{K-1})$ and a function $f \in \mathcal{F}_{\mu,L}$ such that $\mathrm{HB}_{\gamma,\beta}(f)$ cycles on $\mathcal{C}_0$, and $G_0 = (\langle x_i, x_j\rangle)_{0\leqslant i,j\leqslant K-1}$ is the Gram matrix of the iterates and $F_0 = (f(x_i))_{0\leqslant i\leqslant K-1}$ the vector of function values.

Furthermore, for a stationary method, cycling over $\mathcal{C}_0 = (x_0, \dots, x_{K-1})$ is equivalent to cycling over $\mathcal{C}_s = (x_s, x_{s+1} \dots, x_{s-1})$ for any $s \in [\![0, K-1]\!]$. This leads to $K-1$ other solutions to ($\mathcal{P}_{K-\mathrm{SDP}}$):

$$F_s \triangleq (f(x_{i+s}))_{0\leqslant i\leqslant K-1}, \quad G_s \triangleq (\langle x_{i+s}, x_{j+s}\rangle)_{0\leqslant i,j\leqslant K-1}.$$

Remarkably, the solution (F_s, G_s) is obtained from F_0, G_0 by cyclically permuting the elements of F_0 as well as the rows and columns of G_0. Indeed, for F_s, we start with the $(s+1)^{\text{th}}$ element of F, and end with its s^{th} element. Similarly, we start in G_s with the $(s+1)^{\text{th}}$ row and column of G and end with the its s^{th} row and column. Mathematically, for all $s \in [\![0, K-1]\!]$, we have that $F_s = J_K^{-s} F_0$ and $G_s = J_K^{-s} G_0 J_K^s$. And (F_s, G_s) is a solution to ($\mathcal{P}_{K-\mathrm{SDP}}$). By convexity of the set of solutions to ($\mathcal{P}_{K-\mathrm{SDP}}$),

$$(\bar{F}, \bar{G}) \triangleq \left(\frac{1}{K}\sum_{s=0}^{K-1} F_s, \frac{1}{K}\sum_{s=0}^{K-1} G_s\right) = \left(\frac{1}{K}\sum_{s=0}^{K-1} J_K^{-s} F_0, \frac{1}{K}\sum_{s=0}^{K-1} J_K^{-s} G_0 J_K^s\right)$$

is also solution to ($\mathcal{P}_{K-\mathrm{SDP}}$). Moreover, note that the vector $\bar{F}$ is colinear with $\mathbf{1}_K$ and that it only appears in ($\mathcal{P}_{K-\mathrm{SDP}}$) via inner products with vectors orthogonal to $\mathbf{1}_K$ (only differences between 2 components matters). Therefore, $(\mathbf{0}_K, \frac{1}{K}\sum_{s=0}^{K-1} J_K^{-s} G J_K^s)$ is also solution to ($\mathcal{P}_{K-\mathrm{SDP}}$). We use the following fact to conclude.

Fact 8.4.8. *A matrix M is circulant if and only if $M = J_K^{-1} M J_K$.*

Thus $\bar{G} = \frac{1}{K}\sum_{s=0}^{K-1} J_K^{-s} G J_K^s$ is a circulant matrix, as

$$J_K^{-1}\bar{G}J_K = J_K^{-1}\left(\frac{1}{K}\sum_{s=0}^{K-1} J_K^{-s} G J_K^s\right) J_K = \frac{1}{K}\sum_{s=1}^{K} J_K^{-s} G J_K^s = \frac{1}{K}\sum_{s=0}^{K-1} J_K^{-s} G J_K^s = \bar{G},$$

thereby arriving to the desired claim. ■

In the next section, we rely on the symmetries of the Gram matrix to gain insights on the shape of a corresponding cycle $(x_k)_{k\in[\![0,K-1]\!]}$.

From symmetries on the Gram matrix to symmetric cycle shapes

Leveraging Theorem 8.4.5, we obtain that any cycle $(x_0, \dots, x_{K-1})$ obtained from a circulant matrix $\bar{G}$ (i.e., such that $\bar{G}$ is its Gram matrix), has multiple symmetries. This is formalized in the following corollary.

Corollary 8.4.9. *If* (HB)$_{\gamma,\beta}$ *has a cycle*[2] *on $\mathcal{F}_{\mu,L}$ then* (HB)$_{\gamma,\beta}$ *cycles on a* symmetric cycle *$(x_i)_{i\in[\![0,K-1]\!]}$ on $\mathcal{F}_{\mu,L}$. A symmetric cycle is such that its Gram matrix is symmetric circulant, i.e.:*

1. *for all $t \in [\![0, K-1]\!]$, $\|x_t\|^2 = c_0$, i.e. all iterates are on a sphere,*
2. *for all $t \in [\![0, K-1]\!]$, $\langle x_t, x_{t+1}\rangle = c_1$, i.e. the inner product between two consecutive iterates is constant along the cycle,*

[2] in the sense of Definition 8.2.6, items 2 and 3.

3. *more generally, there exist* $(c_s)_{s\in[\![0,K-1]\!]}$*, such that for all* $s,t \in [\![0,K-1]\!]$*,* $\langle x_t, x_{t+s}\rangle = c_s$*, i.e. the inner product between two* s*-separated iterates is constant along the cycle.*

Proof. If $(\text{HB})_{\gamma,\beta}$ has a cycle on $\mathcal{F}_{\mu,L}$ then by Theorem 8.4.2 then Theorem 8.4.7, there exists a circulant solution to $(\mathcal{P}_{K-\text{SDP}})$. A symmetric cycle is obtained from Theorem 8.4.5-1a on the circulant solution. ■

Motivating the roots-of-unity cyclic structure. An example of such a symmetric cycle is the roots-of-unity cycle in dimension 2 (see Definition 8.3.1), that was the focus of Section 8.3. It corresponds to the arguably simplest solution to obtain the symmetries mentioned above. Corollary 8.4.9 thus supports the idea of looking for simple two-dimensional roots-of-unity cycles: the study of those particular cycles, that are sufficient to demonstrate the main non-acceleration result Theorem 8.3.6, takes its roots in this higher-level analysis of the cycles as an SDP and the inherent symmetries of the problem.

Example 8.4.10. *A straightforward application of the symmetrization process given in the proof of Theorem 8.4.7 is the symmetrization of the cycle provided by Lessard et al. (2016), which is a one-dimensional cycle over the three iterates:* $(x_0,x_1,x_2) = \frac{1}{1225}(792,-2208,2592)$*. The Gram matrix of the centered iterates* $(x_0-\bar{x},x_1-\bar{x},x_2-\bar{x})$ *is*

$$G_0 = \left(\frac{8}{49}\right)^2 \begin{pmatrix} 4 & -26 & 22 \\ -26 & 169 & -143 \\ 22 & -143 & 121 \end{pmatrix}$$

After the circulation process described in the proof on Theorem 8.4.7, we obtain that $\bar{G} = \frac{8^2}{49}\begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$*. This Gram matrix is (proportional to) the one of the* 3^{rd}*-roots-of-unity cycle* $\bigcirc_3$*.*

Although we cannot prove that in full generality, if $(\text{HB})_{\gamma,\beta}$ has a cycle on $\mathcal{F}_{\mu,L}$ then $(\text{HB})_{\gamma,\beta}$ cycles over a roots-of-unity cycle, the next result provides a generic decomposition, beyond dimension 2, of the symmetric cycles.

Proposition 8.4.11. *If* $(\text{HB})_{\gamma,\beta}$ *has a cycle on* $\mathcal{F}_{\mu,L}$ *then* $(\text{HB})_{\gamma,\beta}$ *cycles on* K *points* $(x_k)_{k\in[\![0,K-1]\!]}$ *on* $\mathcal{F}_{\mu,L}$*, for which there exist* $\tilde{\nu}_1,\dots,\tilde{\nu}_{\lfloor K/2\rfloor} \geqslant 0$*, such that for all* $k \in [\![0,K-1]\!]$

$$x_k = \begin{pmatrix} \tilde{\nu}_1 & \begin{pmatrix} \cos\left(k\times\frac{2\pi\times 1}{K}\right) \\ \sin\left(k\times\frac{2\pi\times 1}{K}\right)\end{pmatrix} & \text{[block 1]} \\ & \vdots & \\ \tilde{\nu}_\ell & \begin{pmatrix} \cos\left(k\times\frac{2\pi\times \ell}{K}\right) \\ \sin\left(k\times\frac{2\pi\times \ell}{K}\right)\end{pmatrix} & \text{[block } \ell\text{]} \\ & \vdots & \\ \tilde{\nu}_{\lfloor\frac{K-1}{2}\rfloor} & \begin{pmatrix} \cos\left(k\times\frac{2\pi\times \lfloor\frac{K-1}{2}\rfloor}{K}\right) \\ \sin\left(k\times\frac{2\pi\times \lfloor\frac{K-1}{2}\rfloor}{K}\right)\end{pmatrix} & \text{[block } \lfloor\tfrac{K-1}{2}\rfloor\text{]} \\ \tilde{\nu}_{\frac{K}{2}} & \left((-1)^k\right) & \text{[block } \tfrac{K}{2}\text{, only if } K \text{ is even]} \end{pmatrix} \in \mathbb{R}^{K-1}$$

(Symmetric Cycle)

All points $(x_k)_{k\in[\![0,K-1]\!]}$ are in dimension $K-1$, as mentioned in Theorem 8.4.5-1. The points $(x_k)_{k\in[\![0,K-1]\!]}$ are decomposed over a Cartesian product of $\lfloor\frac{K-1}{2}\rfloor$ independent two-dimensional spaces: on each block, the cycle is perfectly regular. For example, on block 1, one recognizes the roots-of-unity shape. If $K-1$ is even, there are exactly $(K-1)/2$ blocks of dimension 2, and if K is even there are exactly $(K-2)/2$ blocks of dimension 2, and one block of dimension 1.

Proof. First, recall that by Theorem 8.4.2 then Theorem 8.4.7, if $(\mathrm{HB})_{\gamma,\beta}$ has a cycle on $\mathcal{F}_{\mu,L}$, then $(\mathcal{P}_{K-\mathrm{SDP}})$ admits a solution $(0,\bar{G})$ with $\bar{G}$ a (symmetric) circulant matrix.

Second, circulant matrices constitute a long standing object of interest in linear algebra, and their reduction properties are well understood (Gray, 2006). We use the following lemma.

Lemma 8.4.12. *A matrix $\bar{G}$ is symmetric and circulant such that $\bar{G}\mathbf{1}_K = 0$, if and only is there exist non-negative $\nu_1,\dots,\nu_{\lfloor K/2\rfloor}$ such that $\bar{G} = \sum_{\ell=1}^{\lfloor K/2\rfloor}\nu_\ell H_\ell$, with $H_\ell \triangleq \left(\cos\left(\frac{2\pi\ell}{K}|i-j|\right)\right)_{i,j}$.*

This is a classical result, whose proof is recalled for completeness in Section 8.C. For each $\ell\in[\![0,\lfloor\frac{K}{2}\rfloor]\!]$, the matrix H_ℓ is a rank 2 matrix and is the Gram matrices of the family of vectors $\begin{pmatrix}\cos\left(k\times\frac{2\pi\ell}{K}\right)\\ \sin\left(k\times\frac{2\pi\ell}{K}\right)\end{pmatrix}_{k\in[\![0,K-1]\!]}$ that corresponds to the ℓ^{th} block.

Overall, considering $\nu_1,\dots,\nu_{\lfloor K/2\rfloor}$ that provide a decomposition $\bar{G}$ as in Lemma 8.4.12, and defining $(x_k)_{k\in[\![0,K-1]\!]}$ by (Symmetric Cycle) with $\tilde{\nu}_\ell = \sqrt{\nu_\ell}$, we obtain that the Gram matrix of $(x_k)_{k\in[\![0,K-1]\!]}$ is $\bar{G}$. Finally, Theorem 8.4.5-1b provides the desired claim. ■

Overall, this provides a complete picture of shape of all cycles of period K. There is no apparent reason for the (Symmetric Cycle) to further reduce to dimension 2, and the proof of such a result is left as an open question. In Subsection 8.4.3, a numerical comparison of the cycles obtained analytically as roots-of-unity cycles in dimension 2 and the ones obtained numerically by solving directly $(\mathcal{P}_{K-\mathrm{SDP}})$. Before turning to this numerical study, we underline that our analysis enables to rewrite the existence of a cycle as a linear feasibility problem.

Casting the problem of finding cycles as a linear feasibility problem

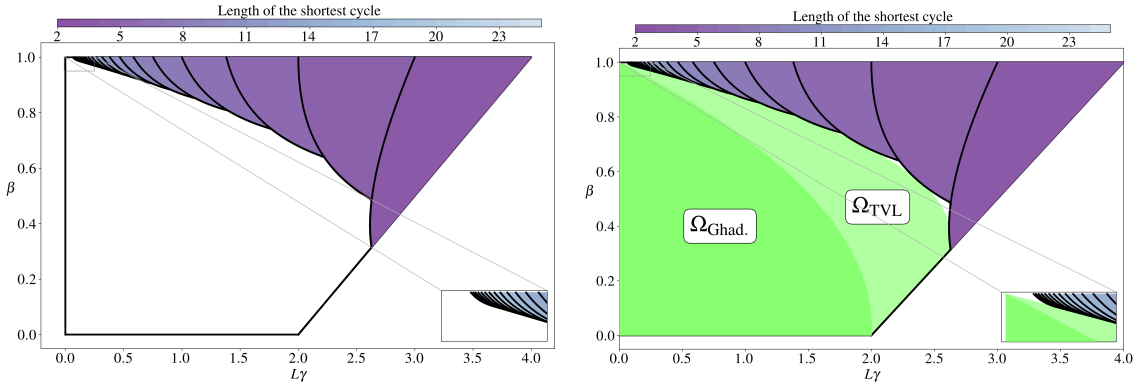
A notable byproduct of Lemma 8.4.12 is the decomposition of any circulant matrix as a positive linear combination of elementary matrices. This enables to reparametrize $(\mathcal{P}_{K-\mathrm{SDP}})$ as a linear problem:

Theorem 8.4.13. *$(\mathcal{P}'_K)$ and $(\mathcal{P}_{K-\mathrm{SDP}})$ are equivalent to the following linear problem:*

$$\exists\nu\in\mathbb{R}^{\lfloor K/2\rfloor}_{\geqslant 0}, \nu\neq\mathbf{0}_{\lfloor K/2\rfloor},\ \forall i\in[\![0,K-1]\!], 0\geqslant\sum_{\ell=1}^{\lfloor K/2\rfloor}\nu_\ell\langle H_\ell, M_{i,j=0}\rangle \qquad (\mathcal{P}_{K-\mathrm{LP}})$$

Equivalently, introducing the matrix $P = \left(\langle M_{i,j=0}, H_\ell\rangle_{i\in[\![0,K-1]\!],\ell\in[\![1,\lfloor K/2\rfloor]\!]}\right)\in\mathbb{R}^{(K-1)\times\lfloor K/2\rfloor}$, $(\mathcal{P}_{K-\mathrm{LP}})$ writes, for $P\nu\in\mathbb{R}^{K-1}$, as:

$$\exists\nu\in\mathbb{R}^{\lfloor K/2\rfloor}_{\geqslant 0},\quad P\nu\leqslant 0.$$



(a) For $K \in [\![2, 25]\!]$, comparison between $\Omega_{K\text{-}\circ\text{-Cycle}}(\mathcal{F}_{\mu,L})$ (analytically obtained in Section 8.3), which borders are represented as black lines, and $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$, represented as the set of purple points, obtained by solving Equation ($\mathcal{P}_{K-\text{SDP}}$).

(b) Comparison between the hyper-parameter regions $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$ for which we obtain cycles, $\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L})$ for which a Lyapunov is found numerically, and $\Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L})$ for which a Lyapunov is known analytically

Figure 8.7: Numerical results on the behavior of (HB) as a function of (γ, β).

Proof. We use Theorem 8.4.7 to obtain a circulant symmetric solution from ($\mathcal{P}_{K-\text{SDP}}$), then by Lemma 8.4.12 all circulant symmetric matrices $\bar{G}$ are written as a linear combination $\sum_{\ell=1}^{\lfloor K/2 \rfloor} \nu_\ell H_\ell$, with $(\nu_\ell)_{\ell \in [\![1, \lfloor K/2 \rfloor]\!]} \in \mathbb{R}_{\geqslant 0}^{\lfloor K/2 \rfloor}$. We obtain ($\mathcal{P}_{K-\text{LP}}$) by parametrizing the problem by ν, as for any (i,j), $\langle \bar{G}, M_{i,j} \rangle = \sum_{\ell=1}^{\lfloor K/2 \rfloor} \nu_\ell \langle H_\ell, M_{i,j} \rangle$, and by observing that for any (i,j), $\langle \bar{G}, M_{i,j} \rangle \leqslant 0$ if and only if $\langle \bar{G}, M_{i-j,0} \rangle \leqslant 0$, as $\bar{G}$ is circulant. ■

In conclusion, we observe that a cycle can either be parametrized by

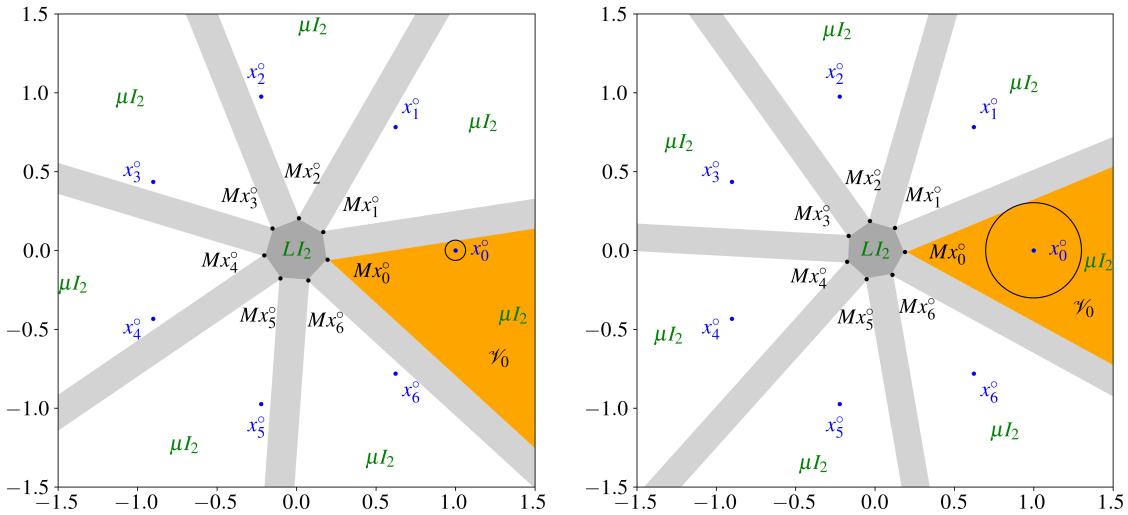
1. Initially, in Definition 8.2.6 and ($\mathcal{P}'_K$), by $(x_k)_{k \in [\![0,K-1]\!]} \in (\mathbb{R}^d)^K$ and $f \in \mathcal{F}_{\mu,L}$.
2. Second, by $G \in \mathcal{S}_K^+(\mathbb{R})$ in ($\mathcal{P}_{K-\text{SDP}}$) and Theorem 8.4.2, with $(K-1)^2$ constraints.
3. Then, using Theorem 8.4.7, by $(c_k)_{k \in [\![0,K-1]\!]}$ the first row of the circulant matrix $\bar{G}$. But the constraints on $c_0, \dots c_{K-1}$ correspond to an SDP type constraint.
4. Finally, in Proposition 8.4.11 and ($\mathcal{P}_{K-\text{LP}}$) , by $(\tilde{\nu}_\ell)_{\ell \in [\![0, \lfloor K/2 \rfloor]\!]}$, onto which only $K-$ *linear* constraints hold.

The latest parametrization, as ($\mathcal{P}_{K-\text{LP}}$), naturally provides the best numerical results, that are given in the next section.

8.4.3 Numerical results on (HB)

In this section, we provide a comparison between the roots-of-unity cycling region $\Omega_{\circ\text{-Cycle}}(\mathcal{F}_{\mu,L})$ obtained analytically in Section 8.3 and $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$ obtained numerically by solving ($\mathcal{P}_{K-\text{SDP}}$). On Figure 8.7a, we observe that the two sets appear to be identical, i.e., numerically $\Omega_{\circ\text{-Cycle}}(\mathcal{F}_{\mu,L}) = \Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$.

Secondly, we compare in Figure 8.7b the hyper-parameter regions $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$ for which we obtain cycles, $\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L})$ for which a Lyapunov is found numerically in Goujaud et al. (2023a) using the approach of Taylor et al. (2018a), and $\Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L})$ for which a Lyapunov is known analytically (see Lemma 8.2.5). We observe that numerically,



(a) $(\gamma, \beta) = (3.3, 0.75)$, $(\mu, L) = (.005, 1)$, $K = 7$ (b) $(\gamma, \beta) = (3.8, 0.95)$, $(\mu, L) = (.005, 1)$, $K = 7$

Figure 8.8: Shape of the counter-example function $\psi^K_{\gamma,\beta,\mu,L}$ (See Figure 8.4), locally quadratic neighborhood $(\mathcal{V}_k)_{k\in[\![0,K-1]\!]}$ (white background, highlighted in orange for $\mathcal{V}_0$), and ball $B(x_0^\circ, r_{\max})$.

$(\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L}))^c$ and $\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L})$ are nearly similar, apart from some small neighborhoods (left white on Figure 8.7b). These observations can be summarized as follows.

Fact 8.4.14. *For any $0 < \mu \leqslant L$:*

$$\Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L}) \subseteq \Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L}) \subseteq \Omega_{\text{cv}}(\mathcal{F}_{\mu,L}) \subseteq (\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L}))^c \subseteq (\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}))^c \subseteq \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}).$$

Conjecture 8.4.15. *For any $0 < \mu \leqslant L$:*

$$\Omega_{\text{Ghad.}}(\mathcal{F}_{\mu,L}) \subsetneq \Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L}) \left\{ \begin{array}{c} \subset \Omega_{\text{cv}}(\mathcal{F}_{\mu,L}) = \\ \text{or} \\ = \Omega_{\text{cv}}(\mathcal{F}_{\mu,L}) \subset \end{array} \right\} (\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L}))^c = (\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}))^c \subsetneq \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}).$$

Although the proof of this conjecture is left open, it is strongly supported by the numerical experiments given above.

In the next section, we demonstrate the cycles obtained in Section 8.3 are robust to a perturbation and to small variations of γ, β, thereby naturally strengthening our results.

8.5 Robustness of the roots-of-unity cycle

In Theorem 8.3.5 we proved that, for any $K \geqslant 2$, for any $(\gamma, \beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$, $(\text{HB})_{\gamma,\beta}(\psi^K_{\gamma,\beta,\mu,L})$ cycles over $\bigcirc_K$, where $\psi^K_{\gamma,\beta,\mu,L}$ is defined in (8.3) as

$$\psi^K_{\gamma,\beta,\mu,L}(x) = \frac{L}{2}\|x\|^2 - \frac{L-\mu}{2} d(x, \text{convh}\{Mx_t^\circ, t \in [\![0, K-1]\!]\})^2.$$

A natural concern is the robustness of this result to an initial perturbation of the starting points, or a random or adversarial perturbations of the gradients or the hyperparameters. In this section, building on the properties of $\psi^K_{\gamma,\beta,\mu,L}$, we establish that the cycle is indeed robust to an initial perturbation of (x_0, x_1) in a neighborhood, to small (random or adversarial) variations of the parameters γ and β at each iteration, and to a small (random or

adversarial) noise on the gradient. We explicitly quantify neighborhoods providing such stability.

To establish stability properties, we leverage the fact that the function $\psi^K_{\gamma,\beta,\mu,L}$ is locally quadratic around the iterates of $\bigcirc_K$. In the rest of the section and the proofs, we extend again (x_k°) to $k \in \mathbb{Z}$ by K-periodicity, as $(x_k^\circ) = (x^\circ_{k \pmod K})$. For $k \in [\![0, K-1]\!]$, we introduce the neighborhood $\mathcal{V}_k$ of x_k° as follows.

Definition 8.5.1 (Locally quadratic neighborhood $(\mathcal{V}_k)_{k\in[\![0,K-1]\!]}$)**.** *For any $k \in [\![0, K-1]\!]$, we denote $\mathcal{V}_k$ the largest neighborhood of x_k° over which $\psi^K_{\gamma,\beta,\mu,L}$ is quadratic with Hessian $\mu \mathrm{I}_2$.*

This neighborhood is represented on Figure 8.8 as the white area surrounding x_k°, highlighted in orange for $\mathcal{V}_0$. Formally, $\mathcal{V}_k$ is composed of all points that have the same projection on $\operatorname{convh}\{Mx_t^\circ, t \in [\![0, K-1]\!]\}$ as x_k°. Moreover, if we define $r_{\max} = -\left\langle (I-M)x_0^\circ, \frac{M(x_1^\circ - x_0^\circ)}{\|M(x_1^\circ - x_0^\circ)\|}\right\rangle$, as the distance between x_0° and the light gray area, as represented in Figure 8.8, we have that for any k, $B(x_k^\circ, r_{\max}) \subseteq \mathcal{V}_k$. Moreover, as the function is locally quadratic, for any $z \in \mathcal{V}_k$,

$$\begin{aligned}\psi^K_{\gamma,\beta,\mu,L}(z) &= \frac{L}{2}\|z\|^2 - \frac{L-\mu}{2}\|z - Mx_k^\circ\|^2 = \frac{\mu}{2}\|z\|^2 + (L-\mu)\langle Mx_k^\circ, z\rangle - \frac{L-\mu}{2}\|Mx_k^\circ\|^2 \\ \nabla\psi^K_{\gamma,\beta,\mu,L}(z) &= Lz - (L-\mu)(z - Mx_k^\circ) \qquad = \nabla\psi^K_{\gamma,\beta,\mu,L}(x_k^\circ) + \mu(z - x_k^\circ). \end{aligned} \tag{8.7}$$

We first consider the case of a perturbation of the initial point, for which we prove the following result:

Theorem 8.5.2. *Consider $0 < \mu < L$, $K \geqslant 2$, the roots-of-unity cycle $(x_k^\circ)_{k\in[\![0,K-1]\!]}$ and $(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$. Let $(z_t)_{t\in\mathbb{N}}$ be the sequence generated by running $(\mathrm{HB})_{\gamma,\beta}$ initialized at $(z_0, z_1) = (x_0^\circ + \delta_0, x_1^\circ + \delta_1)$. There exists κ_P such that if $\sqrt{\|\delta_0\|^2 + \|\delta_1\|^2} \leqslant \kappa_P r_{\max}$, the following properties hold:*

1. *for all $t \in \mathbb{N}$, $z_t \in \mathcal{V}_{t \pmod K}$,*
2. *the sequence $\delta_t \triangleq z_t - x_t^\circ$ follows the dynamic of $(\mathrm{HB})_{\gamma,\beta}(x \mapsto \frac{\mu}{2}\|x\|^2)$, initialized at (δ_0, δ_1),*
3. *consequently, $\|z_t - x_t^\circ\|$ converges to 0 as $t \to \infty$, at rate $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L=\mu})$.*

The first two points are proved simultaneously, and the third point is a consequence of the second. In words, we show that at all iterations, the iterate z_t remains in the locally quadratic neighborhood of x_t°, and that remarkably, the dynamic of the residual (δ_t) is then precisely the one of a (HB) dynamic *on an isotropic (i.e., with $\kappa = 1$) quadratic function*. Indeed, if $z_t \in \mathcal{V}_{t \pmod K}$, we have that:

$$z_{t+1} \overset{(\mathrm{HB})}{=} z_t - \gamma\nabla\psi^K_{\gamma,\beta,\mu,L}(z_t) + \beta(z_t - z_{t-1}).$$

And as $(z_t)_t = (x_t^\circ + \delta_t)_t$, using the formula for the gradient (8.7), we get

$$x_{t+1}^\circ + \delta_{t+1} \overset{(8.7)}{=} x_t^\circ + \delta_t - \gamma\nabla\psi^K_{\gamma,\beta,\mu,L}(x_t^\circ) - \gamma\mu\delta_t + \beta(x_t^\circ - x_{t-1}^\circ) + \beta(\delta_t - \delta_{t-1}).$$

Moreover, as $(\mathrm{HB})_{\gamma,\beta}(\psi^K_{\gamma,\beta,\mu,L})$ cycles over $\bigcirc_K$, $x_{t+1}^\circ = x_t^\circ - \gamma\nabla\psi^K_{\gamma,\beta,\mu,L}(x_t^\circ) + \beta(x_t^\circ - x_{t-1}^\circ)$ thus

$$\delta_{t+1} = \delta_t - \gamma\mu\delta_t + \beta(\delta_t - \delta_{t-1}). \tag{8.8}$$

Which means that as long as $z_t \in \mathcal{V}_{t \pmod K}$, δ_{t+1} is obtained by the dynamic of (HB) on the quadratic isotropic function $x \mapsto \frac{\mu}{2}\|x\|^2$. We now give the complete proof.

Proof. We introduce matrices P and D verifying $PDP^{-1} = \begin{pmatrix} (1+\beta)\mathrm{I}_2 - \mu\gamma \mathrm{I}_2 & -\beta \mathrm{I}_2 \\ \mathrm{I}_2 & 0 \end{pmatrix}$, and such that the operator norm ρ_D of the matrix D, $\rho_D = \|D\|_{\mathrm{op}}$ is smaller than 1, and set $\kappa_P = \frac{1}{\|P\|_{\mathrm{op}}\|P^{-1}\|_{\mathrm{op}}} \leqslant 1$. The existence of such matrices is guaranteed as $(\gamma, \beta) \in \Omega_{\mathrm{cv}}(\mathcal{Q}_{\mu,L})$. To ensure that $z_t \in \mathcal{V}_{t \pmod K}$ (point 1 in Theorem 8.5.2), we prove that $\forall t \geqslant 1, \left\| P^{-1} \begin{pmatrix} \delta_t \\ \delta_{t-1} \end{pmatrix} \right\| \leqslant \frac{r_{\max}}{\|P\|_{\mathrm{op}}}$. Indeed, this is a stronger statement, as it implies that $\forall t \geqslant 1, \left\| \begin{pmatrix} \delta_t \\ \delta_{t-1} \end{pmatrix} \right\| \leqslant r_{\max}$, thus $z_t \in B(x^\circ_{t \pmod K}, r_{\max}) \subseteq \mathcal{V}_{t \pmod K}$.
We prove simultaneously by induction point 2 of Theorem 8.5.2 and the condition $\forall t \geqslant 1, \left\| P^{-1} \begin{pmatrix} \delta_t \\ \delta_{t-1} \end{pmatrix} \right\| \leqslant \frac{r_{\max}}{\|P\|_{\mathrm{op}}}$.

Initialization: $\left\| \begin{pmatrix} \delta_1 \\ \delta_0 \end{pmatrix} \right\| = \sqrt{\|\delta_0\|^2 + \|\delta_1\|^2} \leqslant \kappa_P r_{\max} = \frac{r_{\max}}{\|P\|_{\mathrm{op}}\|P^{-1}\|_{\mathrm{op}}}$ implies $\left\| P^{-1} \begin{pmatrix} \delta_1 \\ \delta_0 \end{pmatrix} \right\| \leqslant \frac{r_{\max}}{\|P\|_{\mathrm{op}}}$.

Induction: By induction hypothesis $z_t \in \mathcal{V}_{t \pmod K}$, thus by (8.8), δ_{t+1} is obtained by (HB):

$$\begin{pmatrix} \delta_{t+1} \\ \delta_t \end{pmatrix} = \begin{pmatrix} (1+\beta)\mathrm{I}_2 - \mu\gamma \mathrm{I}_2 & -\beta \mathrm{I}_2 \\ \mathrm{I}_2 & 0 \end{pmatrix} \begin{pmatrix} \delta_t \\ \delta_{t-1} \end{pmatrix} = PDP^{-1} \begin{pmatrix} \delta_t \\ \delta_{t-1} \end{pmatrix}$$

$$\Rightarrow \quad \left\| P^{-1} \begin{pmatrix} \delta_{t+1} \\ \delta_t \end{pmatrix} \right\| = \left\| DP^{-1} \begin{pmatrix} \delta_t \\ \delta_{t-1} \end{pmatrix} \right\| \overset{\rho_D = \|D\|_{\mathrm{op}}}{\leqslant} \rho_D \left\| P^{-1} \begin{pmatrix} \delta_t \\ \delta_{t-1} \end{pmatrix} \right\| \overset{\text{by induction}}{\leqslant} \rho_D \frac{r_{\max}}{\|P\|_{\mathrm{op}}} \leqslant \frac{r_{\max}}{\|P\|_{\mathrm{op}}}.$$

This concludes the induction and proves the first two items of Theorem 8.5.2. Finally, Point 3 of Theorem 8.5.2 is thus a consequence of point 2 and Proposition 8.2.1.

■

We now give a more general theorem, that provides a stability result w.r.t. the initial points and (potentially adversarial) small perturbations of γ, β at each iteration, and of the gradient oracles. Furthermore, we give explicit neighborhoods preserving the non-convergence property of (HB) on $\psi^K_{\gamma,\beta,\mu,L}$. As in Theorem 8.5.2, we consider that the process starts at a perturbed point $(x^\circ_0 + \delta_0, x^\circ_1 + \delta_1)$. Moreover, instead of applying parameters (γ, β) at each step, we consider that step t is performed with parameters $(\gamma_t, \beta_t) = (\gamma + \delta_{\gamma_t}, \beta + \delta_{\beta_t})$, and rely on perturbed (or noised) gradients $\hat{g}_t(z) = \nabla \psi^K_{\gamma,\beta,\mu,L}(z) + \delta_{g_t}$.

Theorem 8.5.3. *Consider $0 < \mu < L$, $K \geqslant 2$, the roots-of-unity cycle $(x^\circ_k)_{k \in [\![0, K-1]\!]}$, and $(\gamma, \beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$. Let $(z_t)_{t \in \mathbb{N}}$ be the sequence generated by running (HB)$_{\gamma_t,\beta_t}$, with time varying parameters γ_t, β_t, initialized at $(z_0, z_1) = (x^\circ_0 + \delta_0, x^\circ_1 + \delta_1)$, with perturbed gradients $\hat{g}_t(z) = \nabla \psi^K_{\gamma,\beta,\mu,L}(z) + \delta_{g_t}$. There exist κ_P and $\rho_D < 1$ (made explicit in the proof) such that if the following three conditions hold,*

1. $\sqrt{\|\delta_0\|^2 + \|\delta_1\|^2} \leqslant \kappa_P r_{\max}$,
2. *for all* $t \in \mathbb{N}$, $\left(\frac{4}{\gamma} + \mu\kappa_P r_{\max}\right) |\delta_{\gamma_t}| + (2 + 2\kappa_P r_{\max}) |\delta_{\beta_t}| \leqslant \frac{1}{2}(1 - \rho_D)\kappa_P r_{\max}$,
3. *for all* $t \in \mathbb{N}$, $\frac{4}{L}\|\delta_{g_t}\| \leqslant \frac{1}{2}(1 - \rho_D)\kappa_P r_{\max}$,

then, $(z_t)_{t\geqslant 0}$ *keeps cycling in a neighborhood of* $\bigcirc_K$*:* $\|z_t - x^\circ_{t \pmod K}\| \leqslant r_{\max}$ *(and thus* $z_t \in \mathcal{V}_{t \pmod K}$*).*

The proof extends the one of Theorem 8.5.2 and is postponed to Subsection 8.D.1. The constants κ_P, ρ_D are identical to the ones exhibited in the proof of Theorem 8.5.2. Overall, we conclude that, as soon as $r_{\max} > 0$, i.e. as soon as (γ, β) belongs to the interior $\mathrm{Int}(\Omega_{\circ\text{-Cycle}}(\mathcal{F}_{\mu,L}))$ of $\Omega_{\circ\text{-Cycle}}(\mathcal{F}_{\mu,L})$, the cycle $\bigcirc$ is attractive and robust to small variations of the initialization, parameters and gradient oracles. These robustness results ensure that the counter-examples we provide cannot be circumvented by adding small perturbations or stochasticity, and therefore strongly reinforce our conclusions.

In the last section, we explore a different direction, which is the question of acceleration of (HB) if we restrict the class $\mathcal{F}_{\mu,L}$ to functions that have a Lipschitz-continuous Hessian (i.e., under higher-order regularity assumptions), or even any higher order regularity.

8.6 No acceleration of (HB) under higher-order regularity assumptions

A natural way to tentatively extend the proof of (HB)'s acceleration beyond quadratics, consists in assuming *Hessian Lipschitz continuity* (Wang et al., 2022). Indeed, for $0 < \mu < L$, $K \geqslant 2$ and $(\gamma, \beta) \in \Omega_{K\text{-}\circ\text{-Cycle}}(\mathcal{F}_{\mu,L})$ the counter-example function $\psi^K_{\gamma,\beta,\mu,L}$ used in Section 8.3 is quadratic by part, i.e., its hessian is constant by part. As its Hessian is not constant everywhere, $\psi^K_{\gamma,\beta,\mu,L}$ is not even 3 times differentiable, so our counter-example approach seemingly leaves room for improvement under a restricted class of more regular functions. We first focus on the class of Hessian-Lipschitz functions, and will extend the argument to higher-order regularity.

Definition 8.6.1. *Let* $\mathcal{F}^\tau_{\mu,L}$ *be the set of functions* f *in* $\mathcal{F}_{\mu,L}$ *that are three times differentiable, with* τ*-Lipschitz Hessian:*

$$\forall z, w, \|\nabla^2 f(z) - \nabla^2 f(w)\| \leqslant \tau \|z - w\|.$$

For any μ, L and τ, we have $\mathcal{Q}_{\mu,L} \subseteq \mathcal{F}^\tau_{\mu,L} \subset \mathcal{F}_{\mu,L}$. In the rest of the section we consider fixed $0 < \mu < L$, $K \geqslant 2$ and $(\gamma, \beta) \in \mathrm{Int}(\Omega_{K\text{-}\circ\text{-Cycle}}(\mathcal{F}_{\mu,L}))$. We thus omit the indices on the notion of ψ and simply use:

$$\psi := \psi^K_{\gamma,\beta,\mu,L}.$$

In this section, we prove that (HB) *does not accelerate* on $\mathcal{F}^\tau_{\mu,L}$.

Theorem 8.6.2. *For* $0 < \mu < L$*,* $K \geqslant 2$ *and* $(\gamma, \beta) \in \mathrm{Int}(\Omega_{K\text{-}\circ\text{-Cycle}}(\mathcal{F}_{\mu,L}))$*,*

1. *there exists* τ *and a function* $\varphi \in \mathcal{F}^\tau_{\mu,L}$ *such that* $(\mathrm{HB})_{\gamma,\beta}(\varphi)$ *cycles over* $\bigcirc_K$*,*
2. *moreover, for any* $\tau > 0$*,* $(\mathrm{HB})_{\gamma,\beta}$ *has a cycle on* $\mathcal{F}^\tau_{\mu,L}$*.*

The second point means that the set of parameters (γ, β) for which $(\mathrm{HB})_{\gamma,\beta}$ cycles over $\mathcal{F}^\tau_{\mu,L}$ contains the interior of the set of parameters for which $(\mathrm{HB})_{\gamma,\beta}$ cycles on a roots-of-unity cycle over $\mathcal{F}_{\mu,L}$:

$$\forall \tau > 0, \Omega_{\mathrm{Cycle}}(\mathcal{F}^\tau_{\mu,L}) \supseteq \mathrm{Int}(\Omega_{\circ\text{-Cycle}}(\mathcal{F}_{\mu,L})), \tag{8.9}$$

and a direct consequence is that the non-acceleration result Corollary 8.3.7 extends to $\mathcal{F}^{\tau}_{\mu,L}$: $\forall \tau > 0, \forall 0 < \mu < L$,

$$\begin{aligned}
\rho^\star(\mathcal{F}^{\tau}_{\mu,L}) &\triangleq \operatorname*{arg\,min}_{(\gamma,\beta)\in\mathbb{R}\times\mathbb{R}} \rho_{\gamma,\beta}(\mathcal{F}^{\tau}_{\mu,L}) \geqslant \operatorname*{arg\,min}_{(\gamma,\beta)\in(\Omega_{\text{Cycle}}(\mathcal{F}^{\tau}_{\mu,L}))^c} \rho_{\gamma,\beta}(\mathcal{F}^{\tau}_{\mu,L}) && \text{as } \Omega_{\text{cv}}(\mathcal{F}^{\tau}_{\mu,L}) \subseteq (\Omega_{\text{Cycle}}(\mathcal{F}^{\tau}_{\mu,L}))^c \\
&\geqslant \operatorname*{arg\,min}_{(\gamma,\beta)\in(\Omega_{\text{Cycle}}(\mathcal{F}^{\tau}_{\mu,L}))^c} \rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L}) && \text{as } \mathcal{Q}_{\mu,L} \subseteq \mathcal{F}^{\tau}_{\mu,L} \\
&\geqslant \operatorname*{arg\,min}_{(\gamma,\beta)\in(\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}))^c} \rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L}) && \text{by Equation (8.9)} \\
&\geqslant \frac{1-C\kappa}{1+C\kappa} && \text{by Theorem 8.3.6,}
\end{aligned}$$

for a constant C (in fact, any $C \geqslant 50/3$). The rate $\rho^\star(\mathcal{F}^{\tau}_{\mu,L})$ is thus lower bounded, independently of τ, by a non accelerated rate, which proves that Hessian regularity does not help to obtain acceleration.

Interpretation. This result is surprising as a discontinuity appears in $\tau = 0$. Indeed, for $\tau = 0$, $\mathcal{F}^{\tau}_{\mu,L} = \mathcal{Q}_{\mu,L}$, and acceleration is obtained. The mapping $\tau \mapsto \rho^\star(\mathcal{F}^{\tau}_{\mu,L})$ is thus not continuous in $\tau = 0$. A significant nuance to help grasp this phenomenon is that we do *not* prove that for all $\tau > 0$, $\Omega_{\text{o-Cycle}}(\mathcal{F}^{\tau}_{\mu,L}) \supset \Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$, that is that the existence of a roots-of-unity cycle on $\mathcal{F}_{\mu,L}$ implies the existence of a roots-of-unity cycle on $\mathcal{F}^{\tau}_{\mu,L}$ for any $\tau > 0$. Such a property is in fact not true. On the other hand, Item 1 in Theorem 8.6.2 means that $\cup_{\tau>0}\Omega_{\text{o-Cycle}}(\mathcal{F}^{\tau}_{\mu,L}) \supset \Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$. To obtain this result, we show that *there exists* a τ such that we can "regularize" the counter-example ψ to obtain a function in $\mathcal{F}^{\tau}_{\mu,L}$ that cycles over the roots-of-unity cycle. Then Item 2 in Theorem 8.6.2 establishes that there exist a cycle *for all* τ. But this cycle is not necessarily $\bigcirc_K$: on the contrary, the cycle corresponds to a dilated version of $\bigcirc_K$, with a radius that diverges as $\tau \to 0$. Such a cycle thus does not have a limit as $\tau \to 0$, which explains the discontinuity of $\tau \mapsto \rho^\star(\mathcal{F}^{\tau}_{\mu,L})$.

In order to prove Theorem 8.6.2, we establish the following two lemmas, that respectively result in Items 1 and 2 in Theorem 8.6.2, and are given in the next two sections.

8.6.1 Proof of Item 1 of Theorem 8.6.2

First, in order to obtain a more regular function from ψ, we use convolutions. We consider a infinitely differentiable convolution kernel u_ε with bounded support $B(0,\varepsilon)$, such that u_ε is the probability density function of a zero centered random variable. For any $\varepsilon \geqslant 0$, let

$$\varphi_\varepsilon \triangleq u_\varepsilon * \psi.$$

We obtain the following lemma.

Lemma 8.6.3. *Let $0 < \mu < L$, $K \geqslant 2$ and $(\gamma,\beta) \in \operatorname{Int}(\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}))$. Then we have:*

1. *φ_ε is infinitely differentiable and there exists τ such that $\varphi_\varepsilon \in \mathcal{F}^{\tau}_{\mu,L}$.*
2. *For any $\varepsilon \leqslant r_{\max}$ (with $r_{\max}$ defined in Section 8.5), the gradients of φ_ε and ψ coincide on $\bigcirc_K$, that is for all $k \in [\![0, K-1]\!]$,*

$$\nabla\varphi_\varepsilon(x_k^\circ) = \nabla\psi(x_k^\circ).$$

Proof. The first point results from properties of convolution against probability density functions, that are recalled in Lemma 8.E.5 in Section 8.E. The second point is a consequence of the fact that ψ is locally quadratic (thus its gradient $\nabla\psi$ is locally linear). Since ψ is differentiable, we have for all $z \in \mathbb{R}^2$

$$\nabla\varphi_\varepsilon(z) = \nabla\left(\psi * u_\varepsilon\right)(z) = \left((\nabla\psi) * u_\varepsilon\right)(z) = \int_{y\in B(0,\varepsilon)} \nabla\psi(z-y)\, u_\varepsilon(y)\, \mathrm{d}y,$$

where the last step uses the fact that u_ε has support $B(0,\varepsilon)$. Then, for $k \in [\![0, K-1]\!]$ and $z = x_k^\circ$ a point of $\bigcirc_K$, we have that for any $y \in B(0,\varepsilon)$, if $\varepsilon \leqslant r_{\max}$, $x_k^\circ - y \in \mathcal{V}_k$ and by (8.7), $\nabla\psi(x_k^\circ - y) = \nabla\psi(x_k^\circ) - \mu y$. Thus

$$\nabla\varphi_\varepsilon(x_k^\circ) = \nabla\psi(x_k^\circ) - \mu \int_{y\in B(0,\varepsilon)} y u_\varepsilon(y)\mathrm{d}y = \nabla\psi(x_k^\circ).$$

■

Proof of Theorem 8.6.2 (Item 1). Item 1 of Theorem 8.6.2 is a direct consequence of Lemma 8.6.3: we use $\varphi \triangleq \varphi_\varepsilon$ for any $\varepsilon \leqslant r_{\max}$, as $r_{\max} > 0$ for $(\gamma,\beta) \in \mathrm{Int}(\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}))$. ■

8.6.2 Proof of Item 2 of Theorem 8.6.2

As for Item 2 of Theorem 8.6.2, it can be obtained from a scaling argument provided by the following lemma.

Lemma 8.6.4. *Let $0 < \mu < L$, $K \geqslant 2$ and $(\gamma,\beta) \in \mathrm{Int}(\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}))$. Let $\varphi \in \mathcal{F}^\tau_{\mu,L}$ be a function that cycles over $\bigcirc_K$ obtained by Theorem 8.6.2. Define $\varphi_\varepsilon^{(\lambda)}$ as*

$$\varphi_\varepsilon^{(\lambda)}(x) \triangleq \lambda^2 \varphi_\varepsilon\left(\frac{1}{\lambda}x\right).$$

Then:

1. *$\varphi_\varepsilon^{(\lambda)} \in C^\infty$ and $\varphi_\varepsilon^{(\lambda)} \in \mathcal{F}^{\tau/\lambda}_{\mu,L}$.*
2. *$(\mathrm{HB})_{\gamma,\beta}(\varphi_\varepsilon^{(\lambda)})$ cycles on the scaled roots of unity cycle $\lambda \times \bigcirc_K$.*

The function $\varphi_\varepsilon^{(\lambda)}$ is chosen such that the gradients scale proportionally with λ, (as the cycle $\lambda \times \bigcirc_K$), the Hessian is un-scaled, and the third-order derivative scales with $1/\lambda$.

Proof. First, $\varphi_\varepsilon \in C^\infty$, $\varphi_\varepsilon^{(\lambda)} \in C^\infty$. Second, $\varphi_\varepsilon^{(\lambda)} \in \mathcal{F}_{\mu,L}$ as $\nabla^2\varphi_\varepsilon^{(\lambda)}(x) = \nabla^2\varphi_\varepsilon\left(\frac{1}{\lambda}x\right)$ and $\varphi_\varepsilon \in \mathcal{F}_{\mu,L}$. Third, for any $r \geqslant 3$, $\nabla^r\varphi_\varepsilon^{(\lambda)}(x) = \frac{1}{\lambda^{r-2}}\nabla^r\varphi_\varepsilon\left(\frac{1}{\lambda}x\right)$.

Furthermore, $\nabla\varphi_\varepsilon^{(\lambda)}(x) = \lambda\nabla\varphi_\varepsilon\left(\frac{1}{\lambda}x\right)$, hence $\nabla\varphi_\varepsilon^{(\lambda)}(\lambda x) = \lambda\nabla\varphi_\varepsilon(x)$. Therefore, $(\mathrm{HB})_{\gamma,\beta}(\varphi_\varepsilon^{(\lambda)})$ cycles on $\lambda\bigcirc_K$. Indeed, $(\mathrm{HB})_{\gamma,\beta}(\varphi_\varepsilon)$ cycles on $\bigcirc_K$, i.e. $\forall t \in [\![0, K-1]\!], x_{t+1}^\circ = x_t^\circ - \gamma\nabla\varphi_\varepsilon(x_t^\circ) + \beta(x_t^\circ - x_{t-1}^\circ)$. Multiplying by λ, we get $\forall t \in [\![0, K-1]\!], \lambda x_{t+1}^\circ = \lambda x_t^\circ - \gamma\lambda\nabla\varphi_\varepsilon(x_t^\circ) + \beta(\lambda x_t^\circ - \lambda x_{t-1}^\circ) = \lambda x_t^\circ - \gamma\nabla\varphi_\varepsilon^{(\lambda)}(\lambda x_t^\circ) + \beta(\lambda x_t^\circ - \lambda x_{t-1}^\circ)$, that is $(\mathrm{HB})_{\gamma,\beta}(\varphi_\varepsilon^{(\lambda)})$ cycles on $\lambda\bigcirc_K$. ■

Proof of Theorem 8.6.2 (Item 2). It is a direct consequence of Lemma 8.6.4, by choosing λ large enough, it shows that for all $\tau > 0$, $(\mathrm{HB})_{\gamma,\beta}$ has a cycle on $\mathcal{F}^\tau_{\mu,L}$. ■

8.6.3 Beyond third-order regularity

As a conclusion of this section, we showed that Hessian-Lipschitz continuity does not help to obtain acceleration of (HB). It turns out that the arguments work beyond third-order regularity, and that no Lipschitz argument on higher-order derivative can help improving the situation.

In short, the arguments used for controlling the third-order derivative in this section can be extended to impose any bound on higher-order derivatives of the function. In particular, the scaling argument made in Lemma 8.6.4 enables to arbitrarily reduce all derivatives of order higher than 2 of the $\mathcal{C}^\infty$ function obtained in Lemma 8.6.3, whose derivatives are all uniformly bounded. This extension is thus for free: we choose to focus on the third-order derivative in this section for simplicity of exposition, but the generalization follows naturally.

8.7 Concluding remarks

As a brief summary, this work provides a definitive and negative answer to the question of obtaining accelerated convergence rate by using the heavy-ball (HB) method on all smooth strongly convex functions beyond quadratics. In other words, for smooth strongly convex minimization, the complexity of HB is the same as the one of GD, up to at most a constant $50/3 \simeq 17$. Further, we show that this result is stable to reasonable additional assumptions, including that of Lipschitz conditions on the Hessian of the problem, or to functions in the set $\mathcal{C}^\infty \cap \mathcal{F}_{\mu,L}$, and also robust to perturbations to the initial conditions, as well as to parameter and gradient noise.

Furthermore, we propose two constructive approaches, based on the construction of cyclic trajectories, for disproving convergence of (stationary) optimization methods. The first one consists in constructing a two-dimensional cycles (namely *roots-of-unity cycles*), and the second one provides a linear program for testing the existence of higher-dimensional cycles.

Future work and open questions. There remain a few open questions related to the convergence of HB, or that are raised by our work. In particular, we highlight the following points.

First, we now have an upper bound and a lower bound on (HB)'s convergence rate, both in $1-\Theta(\kappa)$. However, those bounds do not match perfectly, as they differ by a constant $\simeq 17$. The theoretical question of the exact rate of (HB) thus remains open.

Second, our non-acceleration result of (HB) holds in any dimension larger than or equal to 2, as our roots-of-unity cycles and counter examples construction only hold in dimension 2 at least. The potential acceleration of (HB) in a one-dimensional space is thus left open.

Lastly, although we proved that acceleration cannot be achieved by assuming higher-order Lipschitz-type regularity, it remains an open question to see if acceleration can be obtained (a) on another intermediary functional class between $\mathcal{Q}_{\mu,L}$ and $\mathcal{F}_{\mu,L}$, (b) under additional information on $f \in \mathcal{F}_{\mu,L}$ – such refined information should go beyond the knowledge of μ and L and could potentially be based on adaptive tunings (e.g., based on online information) of the method.

Conjecture. Finally, we mention the following open conjecture, echoing Conjecture 8.4.15: *for any stationary first-order method, if there exists a cyclic trajectory, there exists a roots-of-unity two-dimensional cycle on the function* (8.3) *(where M depends on the stationary first-order method under consideration)*.

Proving such a conjecture only requires to show that the (Symmetric Cycle) shape obtained in Proposition 8.4.11, can always be reduced to roots-of-unity cycles for any first-order method, as this is the case numerically for HB. While open, this conjecture, if proven, constitutes a promising direction. Indeed, a practical consequence of this conjecture is that testing existence of a cyclic trajectory of a stationary algorithm would only require testing it on one function of the form (8.3) for each cycle length, making the exploration of cyclical behavior of stationary first-order algorithms straightforward.

This appendix is organized in six sections: Section 8.A to 8.E respectively contain complementary proof results to Section 8.2 to 8.6, and Section 8.F a summary table of convergence rates on $\mathcal{Q}_{\mu,L}$ and $\mathcal{F}_{\mu,L}$ for several first-order methods of interest.

8.A Auxiliary proofs from Section 8.2: Proof of Proposition 8.2.1

In this section, we give a complete proof of the asymptotic convergence rate of (HB) on sets of quadratic functions.

Proposition 8.2.1. (Polyak (1964)) *Consider $\beta \in \mathbb{R}$ and $\gamma \in \mathbb{R}$. The worst-case asymptotic convergence rate $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L})$ of (HB-Q)$_{\gamma,\beta}$, over the class $\mathcal{Q}_{\mu,L}$ is:*

1. ***Lazy region**: If $0 < \gamma \leqslant \min\left(\frac{2(1+\beta)}{L+\mu}, \frac{\left(1-\sqrt{\beta}\right)^2}{\mu}\right)$ then $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L}) = \frac{1+\beta-\mu\gamma}{2} + \sqrt{\left(\frac{1+\beta-\mu\gamma}{2}\right)^2 - \beta}$.*
2. ***Robust region**: If $\beta \geqslant 0$, and $\frac{\left(1-\sqrt{\beta}\right)^2}{\mu} \leqslant \gamma \leqslant \frac{\left(1+\sqrt{\beta}\right)^2}{L}$ then $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L}) = \sqrt{\beta}$.*
3. ***Knife's edge**: If $\max\left(\frac{2(1+\beta)}{L+\mu}, \frac{\left(1+\sqrt{\beta}\right)^2}{L}\right) \leqslant \gamma < \frac{2(1+\beta)}{L}$, then $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L}) = \frac{L\gamma-(1+\beta)}{2} + \sqrt{\left(\frac{L\gamma-(1+\beta)}{2}\right)^2 - \beta}$.*
4. ***No convergence**: if $\frac{\gamma}{1+\beta} \geqslant \frac{2}{L}$ or $\gamma \leqslant 0$ then $\rho_{\gamma,\beta}(\mathcal{Q}_{\mu,L}) > 1$.*

Since, this recursion is of second-order, a classical and convenient trick is to consider the variable $X_t \triangleq \begin{pmatrix} x_t - x_\star \\ x_{t-1} - x_\star \end{pmatrix} \in \left(\mathbb{R}^d\right)^2$. It follows the simplified recursion

$$X_{t+1} = \begin{pmatrix} (1+\beta)\mathrm{I} - \gamma H & -\beta \mathrm{I} \\ \mathrm{I} & 0 \end{pmatrix} X_t, \tag{8.10}$$

that can be enrolled to obtain

$$X_T = \begin{pmatrix} (1+\beta)\mathrm{I} - \gamma H & -\beta \mathrm{I} \\ \mathrm{I} & 0 \end{pmatrix}^T X_0,$$

hence the result

$$\|x_t - x_\star\| \leqslant \|X_t\| \leqslant \left\| \begin{pmatrix} (1+\beta)\mathrm{I} - \gamma H & -\beta \mathrm{I} \\ \mathrm{I} & 0 \end{pmatrix}^T \right\|_{\mathrm{op}} \|X_0\|.$$

Finally,

$$\|x_t - x_\star\|^{1/T} \leqslant \left\| \begin{pmatrix} (1+\beta)\mathrm{I} - \gamma H & -\beta \mathrm{I} \\ \mathrm{I} & 0 \end{pmatrix}^T \right\|_{\mathrm{op}}^{1/T} \|X_0\|^{1/T} \underset{T\to\infty}{\longrightarrow} \rho \begin{pmatrix} (1+\beta)\mathrm{I} - \gamma H & -\beta \mathrm{I} \\ \mathrm{I} & 0 \end{pmatrix}$$

where ρ here denotes the spectral radius of the matrix, i.e. the largest complex module of its eigenvalues. Since H is diagonalizable (as self-adjoint operator, or symmetric matrix), we can block-diagonalize the previous matrix as

$$\begin{pmatrix} (1+\beta)\mathrm{I} - \gamma H & -\beta \mathrm{I} \\ \mathrm{I} & 0 \end{pmatrix} \sim \left(\begin{pmatrix} 1+\beta-\gamma\lambda & -\beta \\ 1 & 0 \end{pmatrix}_{\lambda \in \mathrm{Sp}(H)} \right),$$

and then the worst-case asymptotic convergence rate is upper bounded by

$$\rho\begin{pmatrix}(1+\beta)\mathrm{I}-\gamma H & -\beta\mathrm{I}\\ \mathrm{I} & 0\end{pmatrix} = \max_{\lambda\in[\mu,L]}\rho\begin{pmatrix}1+\beta-\gamma\lambda & -\beta\\ 1 & 0\end{pmatrix}.$$

The first thing to notice is that the determinant of $\begin{pmatrix}1+\beta-\gamma\lambda & -\beta\\ 1 & 0\end{pmatrix}$ is β. Therefore, when $\beta \geqslant 1$, at least one eigenvalue has a module larger than or equal to 1 and (HB) provably diverges on some function of $\mathcal{Q}_{\mu,L}$. Then, from now, we only consider $\beta \in (-1, 1)$.

Note that, when $\beta = 0$, we recover (GD) and its convergence rate $\max_{\lambda\in[\mu,L]}|1-\gamma\lambda|$. As for (GD), (HB)'s asymptotic convergence rate is given by the extreme eigenvalue μ or L, depending on the value of $\tilde{\gamma} \triangleq \frac{\gamma}{1+\beta}$. Indeed,

- when $\tilde{\gamma} = \frac{\gamma}{1+\beta} \leqslant \frac{2}{L+\mu}$, then $\max_{\lambda\in[\mu,L]}\rho\begin{pmatrix}1+\beta-\gamma\lambda & -\beta\\ 1 & 0\end{pmatrix} = \rho\begin{pmatrix}1+\beta-\gamma\mu & -\beta\\ 1 & 0\end{pmatrix}$,
- and when $\tilde{\gamma} = \frac{\gamma}{1+\beta} \geqslant \frac{2}{L+\mu}$, then $\max_{\lambda\in[\mu,L]}\rho\begin{pmatrix}1+\beta-\gamma\lambda & -\beta\\ 1 & 0\end{pmatrix} = \rho\begin{pmatrix}1+\beta-\gamma L & -\beta\\ 1 & 0\end{pmatrix}$.

  Moreover, when $\tilde{\gamma} = \frac{\gamma}{1+\beta} \geqslant \frac{2}{L}$, then $\rho\begin{pmatrix}1+\beta-\gamma L & -\beta\\ 1 & 0\end{pmatrix} \geqslant 1$, which cannot guarantee convergence.

Table 8.2: Classification of (HB)'s behavior in three regions, see Figure 8.1 for a graphical description.

Region's name	Range of γ's values	Asymptotic convergence rate
Lazy region	$0 < \gamma \leqslant \min\left(\frac{2(1+\beta)}{L+\mu}, \frac{\left(1-\sqrt{\beta}\right)^2}{\mu}\right)$	$\frac{1+\beta-\mu\gamma}{2} + \sqrt{\left(\frac{1+\beta-\mu\gamma}{2}\right)^2 - \beta}$
Robust region	$\frac{\left(1-\sqrt{\beta}\right)^2}{\mu} \leqslant \gamma \leqslant \frac{\left(1+\sqrt{\beta}\right)^2}{L}$	$\sqrt{\beta}$
Knife's edge	$\max\left(\frac{2(1+\beta)}{L+\mu}, \frac{\left(1+\sqrt{\beta}\right)^2}{L}\right) \leqslant \gamma < \frac{2(1+\beta)}{L}$	$\frac{L\gamma-(1+\beta)}{2} + \sqrt{\left(\frac{L\gamma-(1+\beta)}{2}\right)^2 - \beta}$

Remark 8.A.1. *This proof is based on linear algebra, relying on writing the system as (8.10). Another classical approach to the analysis of (HB) on $\mathcal{Q}_{\mu,L}$ consists in exploiting links between first-order methods and polynomials (see e.g. Fischer (2011); Nemirovskii (1994) or d'Aspremont et al. (Chapter 2 of 2021) for a recent introduction–for more recent exploitation of those links, see, e.g., Berthier et al. (2020); Pedregosa and Scieur (2020); Goujaud et al. (2022b,d); Kim et al. (2022)).*

8.B Auxiliary proofs from Section 8.3

8.B.1 Proof of Theorem 8.3.5

Theorem 8.3.5. (Analytical form of Roots-of-unity cycle region) *For any $K \geqslant 2$, the K^{th}-roots-of-unity cycling region is, for $\theta_K = \frac{2\pi}{K}$:*

$$\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) = \Big\{ (\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}) \mid (\mu\gamma)^2 - 2\left[\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)\right](\mu\gamma) + 2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K) \leqslant 0. \Big\}$$

Moreover for any $K \geqslant 2$, and any $(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$,

$$\psi^K_{\gamma,\beta,\mu,L} : x \mapsto \frac{L}{2}\|x\|^2 - \frac{L-\mu}{2} d(x, \text{convh}\left\{Mx_t^\circ, t \in [\![0,K-1]\!]\right\})^2 \tag{8.3}$$

is a function such that (HB)$_{\gamma,\beta}(\psi^K_{\gamma,\beta,\mu,L})$ *cycles on* $\bigcirc_K$*, with M the linear operator $M \triangleq \frac{(1+\beta-\mu\gamma)I_2 - R - \beta R^{-1}}{(L-\mu)\gamma}$.*

Proof. Let $(\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$. First we prove the expression of $\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$. Then we will prove that for all $(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$, (HB)$_{\gamma,\beta}(\psi)$ cycles.

Expression of $\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$: We have $(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$

$$\begin{aligned}
&\overset{\text{(Lemma 8.3.4)}}{\iff} && \exists f \in \mathcal{F}_{\mu,L} \mid \forall t \in [\![0,K-1]\!], \nabla f(x_t^\circ) = \frac{(1+\beta)I_2 - R - \beta R^{-1}}{\gamma} x_t^\circ, \\
&\left(\bar{f}(x) \triangleq \frac{f(x) - \frac{\mu}{2}\|x\|^2}{L-\mu}\right) \overset{}{\iff} && \exists \bar{f} \in \mathcal{F}_{0,1} \mid \forall t \in [\![0,K-1]\!], \nabla \bar{f}(x_t^\circ) = \frac{(1+\beta-\mu\gamma)I_2 - R - \beta R^{-1}}{(L-\mu)\gamma} x_t^\circ, \\
&\overset{\text{(By definition of } M)}{\iff} && \exists \bar{f} \in \mathcal{F}_{0,1} \mid \forall t \in [\![0,K-1]\!], \nabla \bar{f}(x_t^\circ) = M x_t^\circ, \qquad (8.11) \\
&\overset{\text{(By properties of the Fenchel transform (Rockafellar, 1997, Theorem 23.5))}}{\iff} && \exists \bar{f}^* \in \mathcal{F}_{1,\infty} \mid \forall t \in [\![0,K-1]\!], \nabla \bar{f}^*(Mx_t^\circ) = x_t^\circ, \\
&\left(\hat{f}(x) = \bar{f}^*(x) - \frac{1}{2}\|x\|^2\right) \overset{}{\iff} && \exists \hat{f} \in \mathcal{F}_{0,\infty} \mid \forall t \in [\![0,K-1]\!], \nabla \hat{f}(Mx_t^\circ) = (I-M)x_t^\circ.
\end{aligned}$$

Applying the interpolation theorem (Taylor et al., 2017c, theorem 1), the latest assertion is equivalent to

$$\exists (\hat{f}_t)_{t\in[\![0,K-1]\!]} \mid \forall i \neq j \in [\![0,K-1]\!], \hat{f}_i \geqslant \hat{f}_j + \left\langle (I-M)x_j^\circ, M(x_i^\circ - x_j^\circ) \right\rangle.$$

Note the inner product $\left\langle (I-M)x_j^\circ, M(x_i^\circ - x_j^\circ)\right\rangle$ is also equal to $\left\langle (I-M)x_0^\circ, M(x_{i-j}^\circ - x_0^\circ)\right\rangle$. Hence we can conclude $(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$

$$\begin{aligned}
&\iff \exists (\hat{f}_t)_{t\in[\![0,K-1]\!]} \mid \forall i \neq j \in [\![0,K-1]\!], \hat{f}_i \geqslant \hat{f}_j + \left\langle (I-M)x_0^\circ, M(x_{i-j}^\circ - x_0^\circ)\right\rangle, \\
&\iff \exists (\hat{f}_t)_{t\in[\![0,K-1]\!]} \mid \forall j \in [\![0,K-1]\!], \forall \Delta \in [\![1,K-1]\!], \hat{f}_{j+\Delta} \geqslant \hat{f}_j + \left\langle (I-M)x_0^\circ, M(x_\Delta^\circ - x_0^\circ)\right\rangle.
\end{aligned}$$

Summing up the latest over j implies $\forall \Delta \in [\![1,K-1]\!], 0 \geqslant \left\langle (I-M)x_0^\circ, M(x_\Delta^\circ - x_0^\circ)\right\rangle$.

Reciprocally, this assertion implies the previous one with $\hat{f}_t = 0$ for all t. Hence[3],

$$\boxed{(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) \iff \forall \Delta \in [\![1, K-1]\!], \left\langle M^T(I-M)x_0^\circ, x_\Delta^\circ - x_0^\circ \right\rangle \leqslant 0.} \quad (8.12)$$

This can be written

$$\left(\frac{1+\beta-\gamma\mu-(1+\beta)\cos\theta_K}{(L-\mu)\gamma} - \left(\frac{1+\beta-\gamma\mu-(1+\beta)\cos\theta_K}{(L-\mu)\gamma}\right)^2 - \left(\frac{(1-\beta)\sin\theta_K}{(L-\mu)\gamma}\right)^2\right)(\cos\Delta\theta_K - 1) + \frac{(1-\beta)\sin\theta_K}{(L-\mu)\gamma}\sin\Delta\theta_K \leqslant 0,$$

or dividing by $1-\cos\Delta\theta_K$,

$$-\left(\frac{1+\beta-\gamma\mu-(1+\beta)\cos\theta_K}{(L-\mu)\gamma} - \left(\frac{1+\beta-\gamma\mu-(1+\beta)\cos\theta_K}{(L-\mu)\gamma}\right)^2 - \left(\frac{(1-\beta)\sin\theta_K}{(L-\mu)\gamma}\right)^2\right) + \frac{(1-\beta)\sin\theta_K}{(L-\mu)\gamma}\frac{\sin\Delta\theta_K}{1-\cos\Delta\theta_K} \leqslant 0.$$

This inequality must hold for any $\Delta \in [\![1, K-1]\!]$ and the LHS expression is maximized for $\Delta = 1$. We conclude

$$\boxed{(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) \iff \left\langle M^T(I-M)x_0^\circ, x_1^\circ - x_0^\circ \right\rangle \leqslant 0.}$$

Or equivalently, $(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ if and only if

$$\left(\frac{1+\beta-\gamma\mu-(1+\beta)\cos\theta_K}{(L-\mu)\gamma} - \left(\frac{1+\beta-\gamma\mu-(1+\beta)\cos\theta_K}{(L-\mu)\gamma}\right)^2 - \left(\frac{(1-\beta)\sin\theta_K}{(L-\mu)\gamma}\right)^2\right)(\cos\theta_K - 1) + \frac{(1-\beta)\sin\theta_K}{(L-\mu)\gamma}\sin\theta_K \leqslant 0.$$

After multiplying the above inequality by $\kappa\frac{((L-\mu)\gamma)^2}{1-\cos\theta_K}$ and rearranging the terms, we obtain equivalence with

$$(\mu\gamma)^2 - 2\left[\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)\right](\mu\gamma) + 2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K) \leqslant 0.$$

Cycle on ψ: Recall the function ψ is defined as

$$\psi : x \mapsto \frac{L}{2}\|x\|^2 - \frac{L-\mu}{2}d(x, \operatorname{convh}\{Mx_t, t \in [\![0, K-1]\!]\})^2.$$

We define $\bar{\psi}$ as

$$\bar{\psi} : x \mapsto \frac{\psi(x) - \frac{\mu}{2}\|x\|^2}{L-\mu} = \frac{1}{2}\|x\|^2 - \frac{1}{2}d(x, \operatorname{convh}\{Mx_t, t \in [\![0, K-1]\!]\})^2,$$

[3]Note that this result might be surprising (no function value appears) to the readers familiar with *cyclic monotonicity* (see, e.g., Rockafellar (1997)). In our case, function values naturally disappear as we can arbitrarily set them to zero, which is different than simply not taking function values into account (see, e.g., discussion in (Taylor et al., 2017c, Remark 1)).

and then we have

$$\nabla\bar{\psi}(x) = \mathrm{proj}_{\mathrm{convh}\{Mx_t, t\in[\![0,K-1]\!]\}}(x).$$

Using the same normalization as (8.11), we know that (HB)$_{\gamma,\beta}$ (f) cycles on $\bigcirc_K$ if and only if $\forall t \in [\![0, K-1]\!], \nabla\bar{\psi}(x_t) = Mx_t$, i.e.

$$\forall t \in [\![0, K-1]\!], \mathrm{proj}_{\mathrm{convh}\{Mx_t, t\in[\![0,K-1]\!]\}}(x_t) = Mx_t.$$

The projection on a convex set $\mathrm{proj}_{\mathrm{convh}\{Mx_t, t\in[\![0,K-1]\!]\}}(x_t) = Mx_t$ can be characterized by the following set of inequalities: $\{\langle x_t - Mx_t, Mx_s - Mx_t\rangle \leqslant 0, s \neq t\}$. We conclude with (8.12) that

$$\boxed{(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) \iff \text{(HB)}_{\gamma,\beta}\ (\psi) \text{ cycles on } \bigcirc_K}$$

Construction of ψ: Using (8.11), we search for a function $\bar{f} \in \mathcal{F}_{0,1} \mid \forall t \in [\![0, K-1]\!], \nabla\bar{f}(x_t^\circ) = Mx_t^\circ$. The functions $h : x \mapsto \max_{t\in[\![0,K-1]\!]} \langle Mx_t, x\rangle$ is a natural convex function verifying $\forall t \in [\![0, K-1]\!], \nabla\bar{f}(x_t^\circ) = Mx_t^\circ$. However, the latter is not smooth. We therefore compute M_h its Moreau envelope with smoothing parameter 1, defined as

$$\begin{aligned}
M_h(x) &\triangleq \min_y f(y) + \frac{1}{2}\|x-y\|^2 \\
&= \min_y \max_{t\in[\![0,K-1]\!]} \langle Mx_t, y\rangle + \frac{1}{2}\|x-y\|^2 \\
&= \min_y \max_{(\lambda_t)_{t\in[\![0,K-1]\!]}\geqslant 0 \mid \sum_{t=0}^{K-1}\lambda_t=1} \left\langle \sum_{t=0}^{K-1} \lambda_t Mx_t, y \right\rangle + \frac{1}{2}\|x-y\|^2 \\
&= \max_{(\lambda_t)_{t\in[\![0,K-1]\!]}\geqslant 0 \mid \sum_{t=0}^{K-1}\lambda_t=1} \left\langle \sum_{t=0}^{K-1} \lambda_t Mx_t, x \right\rangle - \frac{1}{2}\| \sum_{t=0}^{K-1} \lambda_t Mx_t\|^2 \\
&= \max_{z\in\mathrm{convh}\{Mx_t\}} \langle z, x\rangle - \frac{1}{2}\|z\|^2 \\
&= \frac{1}{2}\|x\|^2 - \frac{1}{2}d(x, \mathrm{convh}\,\{Mx_t\})^2.
\end{aligned}$$

This function is $\bar{\psi}$ and by renormalization we obtain ψ.

■

8.B.2 Analysis of $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$

In this section, we aim at proving that the roots-of-unity cycling region $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ can be written as $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}) = \{(\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}) \mid \gamma \geqslant \gamma_{\min}(\beta,\mu,L)\}$ for some values of $\gamma_{\min}(\beta,\mu,L)$ to be determined. Theorem 8.3.5 states that, for any $(\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$, $(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ if and only if

$$P_{\beta,K,\mu,L}(\gamma) \leqslant 0 \tag{8.13}$$

with

$$P_{\beta,K,\mu,L} : \gamma \mapsto (\mu\gamma)^2 - 2\left[\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)\right](\mu\gamma) + 2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K). \tag{8.14}$$

The polynomial $P_{\beta,K,\mu,L}$ has two roots for β larger than a value $\beta_-(K,\mu,L)$, we thus use the following notations.

Notation 8.B.1. *For any $\mu, L,\ K \geqslant 2$, we denote*

$$\beta_-(K,\mu,L) \triangleq \frac{\kappa\cos\theta_K{}^2 + (1-\kappa)^2\cos\theta_K - \kappa + (1-\kappa)(1-\cos\theta_K)\sqrt{2\kappa(1+\cos\theta_K)}}{1-2\kappa+\kappa^2\cos\theta_K{}^2},$$

For any $\beta \geqslant \beta_-(K,\mu,L)$, we denote

$$A_K(\beta,\mu,L) \triangleq [\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)]\,, \tag{8.15}$$

$$B_K(\beta,\mu,L) \triangleq \sqrt{[\beta-\cos\theta_K+\kappa(1-\beta\cos\theta_K)]^2 - 2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K)}, \tag{8.16}$$

Finally, we introduce the roots $(\gamma_-(\beta,K,\mu,L), \gamma_{\max}(\beta,K,\mu,L))$ of $P_{\beta,K,\mu,L}$:

$$\gamma_-(\beta,K,\mu,L) \triangleq \frac{A_K(\beta,\mu,L) - B_K(\beta,\mu,L)}{\mu},$$
$$\gamma_+(\beta,K,\mu,L) \triangleq \frac{A_K(\beta,\mu,L) + B_K(\beta,\mu,L)}{\mu}.$$

We underline that we can obtain an alternative expression of $\beta_-(K,\mu,L)$, that intuitively provides an approximation of $\beta_-(K,\mu,L)$ as $\kappa \to 0$

Lemma 8.B.2. *For any $K,\ \mu, L$, it holds that:*

$$\beta_-(K,\mu,L) - \cos\theta_{K+1} = \sqrt{\kappa}(1-\cos\theta_K)\frac{(1+\cos\theta_K)\sqrt{\kappa} + \sqrt{2(1+\cos\theta_K)}}{1+\kappa\cos\theta_K + \sqrt{2\kappa(1+\cos\theta_K)}}.$$

Proof. The proof consists in algebraic manipulations.

$$\begin{aligned}
\beta_-(K,\mu,L) - \cos\theta_K =& \frac{\kappa\cos\theta_K{}^2 + (1-\kappa)^2\cos\theta_K - \kappa + (1-\kappa)(1-\cos\theta_K)\sqrt{2\kappa(1+\cos\theta_K)}}{1-2\kappa+\kappa^2\cos\theta_K{}^2} \\
&- \frac{(1-2\kappa)\cos\theta_K + \kappa^2\cos\theta_K{}^3}{1-2\kappa+\kappa^2\cos\theta_K{}^2} \\
=& \frac{\kappa(\kappa\cos\theta_K - 1)(1-\cos^2\theta_K) + (1-\kappa)(1-\cos\theta_K)\sqrt{2\kappa(1+\cos\theta_K)}}{1-2\kappa+\kappa^2\cos\theta_K{}^2} \\
=& (1-\cos\theta_K)\sqrt{\kappa(1+\cos\theta_K)}\cdot\frac{\sqrt{2}(1-\kappa) + (\kappa\cos\theta_K - 1)\sqrt{\kappa(1+\cos\theta_K)}}{1-2\kappa+\kappa^2\cos\theta_K{}^2} \\
=& \sqrt{\kappa}(1-\cos\theta_K)\sqrt{1+\cos\theta_K} \\
&\times \frac{(1+\kappa\cos\theta_K - \sqrt{2\kappa(1+\cos\theta_K)})(\sqrt{2}+\sqrt{\kappa(1+\cos\theta_K)})}{(1+\kappa\cos\theta_K - \sqrt{2\kappa(1+\cos\theta_K)})(1+\kappa\cos\theta_K + \sqrt{2\kappa(1+\cos\theta_K)})} \\
=& \sqrt{\kappa}(1-\cos\theta_K)\frac{(1+\cos\theta_K)\sqrt{\kappa} + \sqrt{2(1+\cos\theta_K)}}{1+\kappa\cos\theta_K + \sqrt{2\kappa(1+\cos\theta_K)}}.
\end{aligned}$$

■

Using those notations, we can restate the condition (8.13) as follows.

Fact 8.B.3. *$P_{\beta,K,\mu,L}(\gamma) \leqslant 0$ if and only if $\beta \geqslant \beta_-(K,\mu,L)$ and*

$$\gamma_-(\beta,K,\mu,L) \leqslant \gamma \leqslant \gamma_+(\beta,K,\mu,L), \tag{8.17}$$

i.e.:

$$\begin{aligned}
\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}) = \Big\{&(\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}) \mid \\
&\exists K \geqslant 3 \text{ such that } \beta \geqslant \beta_-(K,\mu,L) \text{ and } \gamma_-(\beta,K,\mu,L) \leqslant \gamma \leqslant \gamma_+(\beta,K,\mu,L)\Big\}.
\end{aligned} \tag{8.18}$$

In words, for any $\beta \geqslant 0$, the set of all γ such that $(\gamma,\beta) \in \Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ is a union of intervals given by (8.18), non-necessarily connected. The next theorem states that this set of γ is actually a single interval as soon as κ is sufficiently small.

Theorem 8.B.4 (Analytical form of Roots-of-unity cycle region)**.** *If* $\kappa \leqslant \left(\frac{3-\sqrt{5}}{4}\right)^2$*, the roots-of-unity cycling region is:*

$$\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}) = \Big\{(\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}) \mid \exists K \geqslant 3 \text{ such that } \beta \geqslant \beta_-(K,\mu,L) \text{ and } \gamma \geqslant \gamma_-(\beta,K,\mu,L)\Big\}.$$

This theorem means that we can ignore, the condition $\gamma \leqslant \gamma_+(\beta,K,\mu,L)$ in the parametric description of $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ in (8.17). Note that $\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) \neq \{(\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}) \mid \beta \geqslant \beta_-(K,\mu,L) \text{ and } \gamma \geqslant \gamma_-(\beta,K,\mu,L)\}$: the theorem states that the union (over $K \geqslant 2$) of those sets can be written without an upper bound on γ, not each set individually. That is, when $(\gamma,\beta) \notin \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ because $\gamma \geqslant \gamma_+(\beta,K,\mu,L)$, $(\gamma,\beta) \in \Omega_{K'\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ for some $K' \leqslant K$.

Remark 8.B.5 ($K=2$)**.** *Plugging* $K=2$ *into* (8.17) *leads to* $\gamma \in \left[\frac{2(1+\beta)}{L}, \frac{2(1+\beta)}{\mu}\right]$ *whose intersection with* $\Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$ *is empty. Hence,* $\Omega_{2\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) = \emptyset$*, which explains why Theorem 8.B.4 does not consider cycles of length 2.*

In order to prove Theorem 8.B.4, we first state Lemma 8.B.6.

Lemma 8.B.6. *We assume* $\kappa \leqslant \left(\frac{3-\sqrt{5}}{4}\right)^2$*. For any* $K \geqslant 2$*, and any* $\beta \geqslant \beta_-(K+1,\mu,L)$*, we have*

$$\gamma_-(\beta,K,\mu,L) \leqslant \gamma_+(\beta,K+1,\mu,L).$$

First, we show that thanks to Lemma 8.B.6, we can establish Theorem 8.B.4.

Proof. [Theorem 8.B.4] We denote $\Omega_{\leqslant K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) = \bigcup_{3\leqslant \bar{K}\leqslant K} \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ the set of $(\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$ such that there exists $\bar{K} \leqslant K$ with $(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$. We prove by induction over K that

$$\Omega_{\leqslant K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) = \Big\{(\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}) \mid \exists \bar{K} \leqslant K \text{ such that } \beta \geqslant \beta_-(\bar{K},\mu,L) \text{ and } \gamma \geqslant \gamma_-(\beta,\bar{K},\mu,L)\Big\}. \quad (8.19)$$

Initialization ($K=3$)**:** From Lemma 8.B.6, $\gamma_+(\beta,3,\mu,L) \geqslant \gamma_-(\beta,2,\mu,L)$, and from Remark 8.B.5, $\gamma_-(\beta,2,\mu,L) = \frac{2(1+\beta)}{L}$. Thus

$$\begin{aligned}\Omega_{\leqslant 3\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) &= \Omega_{3\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) \\ &= \{(\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}) \mid \beta \geqslant \beta_-(3,\mu,L) \text{ and } \mu\gamma \geqslant \mu\gamma_-(\beta,3,\mu,L)\}\,.\end{aligned}$$

Indeed, the additional assumption that $\gamma \leqslant \gamma_+(\beta,3,\mu,L)$ is useless as $\gamma_+(\beta,3,\mu,L)$ is larger than $\frac{2(1+\beta)}{L}$, which corresponds to the right-hand side border of $\Omega_{\text{cv}}(\mathcal{F}_{\mu,L})$.

Induction: Let us assume that (8.19) holds for some K. We prove it still holds for $K+1$. Indeed, $(\gamma,\beta) \in \Omega_{\leqslant K+1\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ if and only if $(\gamma,\beta) \in \Omega_{\leqslant K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ or $(\gamma,\beta) \in \Omega_{K+1\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$, i.e. if, by our induction hypothesis $\gamma \geqslant \gamma_-(\beta,\bar{K},\mu,L)$ for some $\bar{K} \leqslant K$, or $\gamma_-(\beta,K+1,\mu,L) \leqslant \gamma \leqslant \gamma_+(\beta,K+1,\mu,L)$.

Then by Lemma 8.B.6, if $\gamma \geqslant \gamma_+(\beta,K+1,\mu,L)$, we have $\gamma \geqslant \gamma_-(\beta,K,\mu,L)$, thus $(\gamma,\beta) \in \Omega_{\leqslant K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$. Finally, $(\gamma,\beta) \in \Omega_{\leqslant K+1\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$ if and only if $\gamma \geqslant \gamma_-(\beta,\bar{K},\mu,L)$

for some $\bar{K} \leqslant K$, or $\gamma \geqslant \gamma_-(\beta, K+1, \mu, L)$, i.e. if and only if $\gamma \geqslant \gamma_-(\beta, \bar{K}, \mu, L)$ for some $\bar{K} \leqslant K+1$, which concludes the proof. ■

We now prove Lemma 8.B.6.

Proof. [Lemma 8.B.6] Since for any K, $\gamma_-(\beta, K, \mu, L)$ is defined as $\frac{A_K(\beta,\mu,L)-B_K(\beta,\mu,L)}{\mu}$, we show the equivalent formulation

$$A_K(\beta,\mu,L) - B_K(\beta,\mu,L) \leqslant A_{K+1}(\beta,\mu,L) + B_{K+1}(\beta,\mu,L). \tag{8.20}$$

First, we rely on the following sequence of implication, that shows that proving (8.21) hereafter is sufficient to establish (8.20).

$$\begin{aligned}
&(A_K(\beta,\mu,L) - A_{K+1}(\beta,\mu,L))^2 \leqslant B_K(\beta,\mu,L)^2 - B_{K+1}(\beta,\mu,L)^2 && (8.21)\\
&\overset{B_{K+1}(\beta,\mu,L)^2\geqslant 0}{\Longrightarrow} (A_K(\beta,\mu,L) - A_{K+1}(\beta,\mu,L))^2 \leqslant B_K(\beta,\mu,L)^2 && (8.22)\\
&\overset{B_K(\beta,\mu,L)\geqslant 0}{\Longrightarrow} A_K(\beta,\mu,L) - A_{K+1}(\beta,\mu,L) \leqslant B_K(\beta,\mu,L)\\
&\overset{\text{Reordering terms}}{\Longleftrightarrow} A_K(\beta,\mu,L) - B_K(\beta,\mu,L) \leqslant A_{K+1}(\beta,\mu,L)\\
&\overset{B_{K+1}(\beta,\mu,L)\geqslant 0}{\Longrightarrow} A_K(\beta,\mu,L) - B_K(\beta,\mu,L) \leqslant A_{K+1}(\beta,\mu,L) + B_{K+1}(\beta,\mu,L).
\end{aligned}$$

Equation (8.22) has the advantage of isolating $B_K(\beta,\mu,L)$ so that we get rid of the square-root appearing in its definition. Equation (8.21) has the additional advantage of getting rid of the terms that are independent of the cosines and of making a factor $\cos\theta_{K+1} - \cos\theta_K$ appear on both sides of the inequality. Indeed, first we have

$$\begin{aligned}
A_K(\beta,\mu,L) - A_{K+1}(\beta,\mu,L) &\overset{(8.15)}{=} [\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)]\\
&\quad - [\beta - \cos\theta_{K+1} + \kappa(1-\beta\cos\theta_{K+1})]\\
&= (1+\beta\kappa)(\cos\theta_{K+1} - \cos\theta_K)\\
(A_K(\beta,\mu,L) - A_{K+1}(\beta,\mu,L))^2 &= (1+\beta\kappa)^2(\cos\theta_{K+1} - \cos\theta_K)^2. \qquad (8.23)
\end{aligned}$$

Also, $B_K(\beta,\mu,L)^2$ can be simplified, expanding and reordering terms, as

$$\begin{aligned}
B_K(\beta,\mu,L)^2 &\overset{(8.16)}{=} [\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)]^2 - 2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K)\\
&= [\beta + \kappa - (1+\beta\kappa)\cos\theta_K]^2 - 2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K)\\
&= (\beta+\kappa)^2 - 2(\beta+\kappa)(1+\beta\kappa)\cos\theta_K + (1+\beta\kappa)^2\cos^2\theta_K\\
&\quad - 2\kappa(1+\beta^2-2\beta\cos\theta_K - (1+\beta^2)\cos\theta_K + 2\beta\cos^2\theta_K)\\
&= \Big[(\beta+\kappa)^2 - 2\kappa(1+\beta^2)\Big] - 2\Big[(\beta+\kappa)(1+\beta\kappa) - \kappa(1+\beta)^2\Big]\cos\theta_K\\
&\quad + \Big[(1+\beta\kappa)^2 - 4\beta\kappa\Big]\cos^2\theta_K\\
&= \underbrace{\Big[(\beta+\kappa)^2 - 2\kappa(1+\beta^2)\Big]}_{\text{independent of } K} - 2\beta(1-\kappa)^2\cos\theta_K + (1-\beta\kappa)^2\cos^2\theta_K.
\end{aligned}$$

While inequality (8.22) thus writes

$$(1+\beta\kappa)^2(\cos\theta_{K+1} - \cos\theta_K)^2 \leqslant \Big[(\beta+\kappa)^2 - 2\kappa(1+\beta^2)\Big] - 2\beta(1-\kappa)^2\cos\theta_K + (1-\beta\kappa)^2\cos^2\theta_K,$$

and is thus not easy to handle, $B_K(\beta,\mu,L)^2 - B_{K+1}(\beta,\mu,L)^2$ simplifies as

$$\begin{aligned}
B_K(\beta,\mu,L)^2 - B_{K+1}(\beta,\mu,L)^2 &= -2\beta(1-\kappa)^2(\cos\theta_K - \cos\theta_{K+1})\\
&\quad + (1-\beta\kappa)^2(\cos^2\theta_K - \cos^2\theta_{K+1}), \qquad (8.24)
\end{aligned}$$

and using (8.23) and (8.24), inequality (8.21) thus writes

$$\begin{aligned}(1+\beta\kappa)^2(\cos\theta_{K+1}-\cos\theta_K)^2 \leqslant & 2\beta(1-\kappa)^2(\cos\theta_{K+1}-\cos\theta_K)\\ & -(1-\beta\kappa)^2(\cos^2\theta_{K+1}-\cos^2\theta_K)\\ & =\left[2\beta(1-\kappa)^2-(1-\beta\kappa)^2(\cos\theta_{K+1}+\cos\theta_K)\right]\\ & \times(\cos\theta_{K+1}-\cos\theta_K).\end{aligned}$$

Dividing by the positive term $\cos\theta_{K+1}-\cos\theta_K$[4], we get that (8.21) is equivalent to

$$(1+\beta\kappa)^2(\cos\theta_{K+1}-\cos\theta_K)\leqslant\left[2\beta(1-\kappa)^2-(1-\beta\kappa)^2(\cos\theta_{K+1}+\cos\theta_K)\right].$$

By ordering terms, this is also equivalent to

$$(1+\beta\kappa)^2(\cos\theta_{K+1}-\cos\theta_K)+(1-\beta\kappa)^2(\cos\theta_{K+1}+\cos\theta_K)\leqslant 2\beta(1-\kappa)^2,$$

or again, dividing by $4\beta\kappa$,

$$\frac{1}{2}\left(\frac{1}{\beta\kappa}+\beta\kappa\right)\cos\theta_{K+1}\leqslant\frac{(1-\kappa)^2}{2\kappa}+\cos\theta_K. \tag{8.25}$$

Note that we need this last equation to hold for all $\beta\geqslant\beta_-(K+1,\mu,L)$. Moreover, while the RHS does not depend on β, the LHS increases or decreases with respect to β, according to the sign of $\cos\theta_{K+1}$. We thus have to distinguish several cases.

- When $K=2$, $\cos\theta_{K+1}=\cos\theta_3=-1/2<0$. Then the LHS increases and it is sufficient to verify (8.25) for $\beta=1$. We have: if $-\frac{1}{4}\left(\frac{1}{\kappa}+\kappa\right)\leqslant\frac{(1-\kappa)^2}{2\kappa}-1$, then (8.20) holds for $K=2$. A simple technical computation gives that this condition holds if and only if $\kappa\leqslant\frac{4-\sqrt{7}}{3}$.
- When $K=3$, $\cos\theta_{K+1}=\cos\theta_4=0$. Then the LHS does not depend on β neither. Equation (8.25) is independent of β and is written $0\leqslant\frac{(1-\kappa)^2}{2\kappa}-\frac{1}{2}$. We conclude that, if $\kappa\leqslant\frac{3-\sqrt{5}}{2}$, then (8.B.6) holds for $K=3$.
- When $K\geqslant 4$, $\cos\theta_{K+1}\geqslant\cos\theta_5>0$, hence the LHS decreases with respect to β. It is sufficient to verify (8.25) for $\beta=\beta_-(K+1,\mu,L)$, or even for any value smaller than $\beta_-(K+1,\mu,L)$ to prove the lemma.

We use the following lower bound on $\beta_-(K+1,\mu,L)$, obtained from Lemma 8.B.2

$$\begin{aligned}\beta_-(K+1,\mu,L)-\cos\theta_{K+1} &\overset{8.B.2}{=}\sqrt{\kappa}(1-\cos\theta_{K+1})\frac{(1+\cos\theta_{K+1})\sqrt{\kappa}+\sqrt{2(1+\cos\theta_{K+1})}}{1+\kappa\cos\theta_{K+1}+\sqrt{2\kappa(1+\cos\theta_{K+1})}}\\ &\geqslant\sqrt{\kappa}(1-\cos\theta_{K+1}).\\ &=\cos\theta_{K+1}(1+\sqrt{\kappa}\xi_{K+1}),\end{aligned}$$

with $\xi_{K+1}\triangleq\frac{1}{\cos\theta_{K+1}}-1$. Overall, for $K\geqslant 4$, (8.25) is valid for all $\beta\geqslant\beta_-(K+1,\mu,L)$ if it is valid at $\cos\theta_{K+1}(1+\sqrt{\kappa}\xi_{K+1})$. Plugging this value into (8.25) it is thus sufficient to prove that

$$\frac{1}{2}\left(\frac{1}{\kappa(1+\sqrt{\kappa}\xi_{K+1})}+\kappa\cos^2\theta_{K+1}(1+\sqrt{\kappa}\xi_{K+1})\right)\leqslant\frac{(1-\kappa)^2}{2\kappa}+\cos\theta_K.$$

[4]This simplification was the motivation to prove the relaxation (8.21) instead of (8.22). Interestingly, this relaxation is tight when $B_{K+1}(\beta,\mu,L)=0$, i.e. for $\beta=\beta_-(K+1,\mu,L)$. As we will see later, this value of β is the only one that actually matters (if $K\neq 3$).

Multiplying by 2κ and reordering terms, the latter is equivalent to

$$\frac{1}{1+\sqrt{\kappa}\xi_{K+1}} - (1-\kappa)^2 \leqslant 2\kappa\cos\theta_K - \kappa^2 \underbrace{\cos^2\theta_{K+1}}_{\leqslant \cos\theta_{K+1}} \underbrace{(1+\sqrt{\kappa}\xi_{K+1})}_{\leqslant 1+\xi_{K+1}}. \tag{8.26}$$

First, as for any $\tau \geqslant 0$, $\frac{1}{1+\tau} \leqslant 1-\tau+\tau^2$, the LHS of (8.26) is upper bounded by $1-\sqrt{\kappa}\xi_{K+1}+\kappa\xi_{K+1}^2-(1-\kappa)^2$. Second, the RHS of (8.26) is lower bounded by $2\kappa\cos\theta_K-\kappa^2$. It is thus sufficient to prove

$$1-\sqrt{\kappa}\xi_{K+1}+\kappa\xi_{K+1}^2-(1-2\kappa+\kappa^2) \leqslant 2\kappa\cos\theta_K-\kappa^2.$$

Simplifying and reordering terms, this is equivalent to

$$\sqrt{\kappa}\xi_{K+1} \geqslant \kappa(\xi_{K+1}^2+2-2\cos\theta_K),$$

or again

$$\frac{1}{\sqrt{\kappa}} \geqslant \left(\xi_{K+1}+\frac{2(1-\cos\theta_K)}{\xi_{K+1}}\right) = \left(\frac{1}{\cos\theta_{K+1}}-1+\frac{2\cos\theta_{K+1}(1-\cos\theta_K)}{(1-\cos\theta_{K+1})}\right).$$

Finally,

$$\begin{aligned}
\frac{1}{\cos\theta_{K+1}}-1+\frac{2\cos\theta_{K+1}(1-\cos\theta_K)}{(1-\cos\theta_{K+1})} &\overset{\text{Lemma 8.B.7}}{\leqslant} \frac{1}{\cos\theta_{K+1}}-1+2\times 1\times\frac{3}{2} = \frac{1}{\cos\theta_{K+1}}+2\\
&\overset{K\geqslant 4}{\leqslant} \frac{1}{\cos\theta_5}+2 = \frac{1}{\cos\frac{2\pi}{5}}+2\\
&= 3+\sqrt{5}.
\end{aligned}$$

Ultimately, $\kappa \leqslant \left(\frac{1}{3+\sqrt{5}}\right)^2 = \left(\frac{3-\sqrt{5}}{4}\right)^2$ is a sufficient condition for Equation (8.20) to hold for any $K \geqslant 4$. As a summary, we proved the following sufficient conditions so that (8.20) holds:

- For $K=2$, (8.20) holds as soon as $\kappa \leqslant \frac{4-\sqrt{7}}{3}$.
- For $K=3$, (8.20) holds as soon as $\kappa \leqslant \frac{3-\sqrt{5}}{2}$.
- For $K \geqslant 4$, (8.20) holds as soon as $\kappa \leqslant \left(\frac{3-\sqrt{5}}{4}\right)^2$.

Among those 3 values, $\left(\frac{3-\sqrt{5}}{4}\right)^2$ is the smallest. Hence, we conclude that (8.20) holds for any $K \geqslant 2$ if $\kappa \leqslant \left(\frac{3-\sqrt{5}}{4}\right)^2$.

■

8.B.3 Proof of Theorem 8.3.6

In order to prove Theorem 8.3.6, we first establish technical results in Subsection 8.B.3, then prove the result in Subsection 8.B.3

Technical lemmas

We start by proving the following technical lemmas.

Lemma 8.B.7. *For any integer* $K \geqslant 2$, $1 \leqslant \frac{1-\cos\theta_K}{1-\cos\theta_{K+1}} \leqslant \frac{3}{2}$.

Proof. First of all, let's note that

$$\frac{1-\cos\frac{2\pi}{K}}{1-\cos\frac{2\pi}{K+1}} = \frac{\sin^2\frac{\pi}{K}}{\sin^2\frac{\pi}{K+1}}.$$

Since $\sin$ is positive and increasing on $[0,\pi/2]$, $\sin^2\frac{\pi}{K} \geqslant \sin^2\frac{\pi}{K+1}$, hence for all $K \geqslant 2$, $1 \leqslant \frac{1-\cos\frac{2\pi}{K}}{1-\cos\frac{2\pi}{K+1}}$. Note moreover this bound is tight as $\frac{1-\cos\frac{2\pi}{K}}{1-\cos\frac{2\pi}{K+1}} \underset{K\to\infty}{\to} 1$. It remains to show that for all integer $K \geqslant 2$, $\frac{\sin\frac{\pi}{K}}{\sin\frac{\pi}{K+1}} \leqslant \sqrt{\frac{3}{2}}$. We study the four following cases:

- for $K=2$, $\frac{\sin\frac{\pi}{K}}{\sin\frac{\pi}{K+1}} = \frac{1}{\sqrt{3}/2} < \sqrt{\frac{3}{2}}$,
- for $K=3$, $\frac{\sin\frac{\pi}{K}}{\sin\frac{\pi}{K+1}} = \frac{\sqrt{3}/2}{\sqrt{2}/2} = \sqrt{\frac{3}{2}}$,
- for $K=4$, $\frac{\sin\frac{\pi}{K}}{\sin\frac{\pi}{K+1}} = \frac{\sqrt{2}/2}{\sqrt{10-2\sqrt{5}}/4} < \sqrt{\frac{3}{2}}$,
- for $K \geqslant 5$,

$$\begin{aligned}\frac{\sin\frac{\pi}{K}}{\sin\frac{\pi}{K+1}} &= \frac{\sin(\frac{\pi}{K+1}+\frac{\pi}{K(K+1)})}{\sin\frac{\pi}{K+1}} = \frac{\sin\frac{\pi}{K+1}\cos\frac{\pi}{K(K+1)}+\cos\frac{\pi}{K+1}\sin\frac{\pi}{K(K+1)}}{\sin\frac{\pi}{K+1}}\\ &= \cos\frac{\pi}{K(K+1)} + \operatorname{cotan}\frac{\pi}{\mathrm{K}+1}\sin\frac{\pi}{\mathrm{K}(\mathrm{K}+1)}\\ &\leqslant 1 + \frac{K+1}{\pi}\frac{\pi}{K(K+1)} = 1+\frac{1}{K} \leqslant \frac{6}{5} < \sqrt{\frac{3}{2}}.\end{aligned}$$

This shows that

$$\forall \text{ integer } K \geqslant 2,\ \frac{1-\cos\frac{2\pi}{K}}{1-\cos\frac{2\pi}{K+1}} \leqslant \frac{3}{2}.$$

Note furthermore that this bound is also tight as reached for $K=3$.

■

Next, we establish a second technical result.

Lemma 8.B.8. *For any* $\beta \in [0,1]$, *there exists* $K \geqslant 2$ *such that* $\frac{2}{3} \leqslant \frac{\beta-\cos\theta K}{1-\beta} \leqslant \frac{3}{2}$.

Proof. Let $\beta \in [0,1]$. And let $Z \geqslant 2$ a real number such that $\beta = \frac{1+\cos\frac{2\pi}{Z}}{2}$. We note that $\frac{1+\cos\frac{2\pi}{\lfloor Z\rfloor}}{2} \leqslant \beta < \frac{1+\cos\frac{2\pi}{\lfloor Z\rfloor+1}}{2}$. Besides, from Lemma 8.B.7, $1 \leqslant \frac{1-\cos\frac{2\pi}{\lfloor Z\rfloor}}{1-\cos\frac{2\pi}{\lfloor Z+1\rfloor}} \leqslant \frac{3}{2}$, which implies

$$1 \leqslant \frac{1-\cos\frac{2\pi}{\lfloor Z\rfloor}}{1-\cos\frac{2\pi}{Z}} \leqslant \frac{5}{4} \quad \text{or} \quad 1 \leqslant \frac{1-\cos\frac{2\pi}{Z}}{1-\cos\frac{2\pi}{\lfloor Z+1\rfloor}} \leqslant \frac{6}{5}.$$

In the first case, we define $K = \lfloor Z \rfloor$ while in the second case, we define $K = \lfloor Z + 1 \rfloor$. In any case, $\frac{5}{6} \leqslant \frac{1-\cos\frac{2\pi}{K}}{1-\cos\frac{2\pi}{Z}} \leqslant \frac{5}{4}$. Moreover, we have $\frac{\beta-\cos\frac{2\pi}{K}}{1-\beta} = -1 + 2\frac{1-\cos\frac{2\pi}{K}}{1-\cos\frac{2\pi}{Z}}$, and we conclude $\frac{2}{3} \leqslant \frac{\beta-\cos\frac{2\pi}{K}}{1-\beta} \leqslant \frac{3}{2}$. ■

Third, we establish a final technical result.

Lemma 8.B.9. *Assume that $\kappa \leqslant \frac{1}{16}$. For any $\beta \in [0,1]$, for any $K \geqslant 2$ such that $\frac{2}{3} \leqslant \frac{\beta-\cos\theta_K}{1-\beta} \leqslant \frac{3}{2}$, we have $\beta \geqslant \beta_-(K,\mu,L)$.*

Proof. The proof consists in proving that

$$\frac{1-\beta_-(K,\mu,L)}{1-\cos\theta_K} \geqslant \frac{3}{5}.$$

Then, since $\frac{\beta-\cos\theta_K}{1-\beta} \geqslant \frac{2}{3}$, we have $\frac{1-\cos\theta_K}{1-\beta} \geqslant \frac{5}{3}$, and finally $\frac{1-\beta}{1-\cos\theta_K} \leqslant \frac{3}{5} \leqslant \frac{1-\beta_-(K,\mu,L)}{1-\cos\theta_K}$, concluding that $\beta \geqslant \beta_-(K,\mu,L)$.

By Lemma 8.B.2, we have

$$\frac{\beta_-(K,\mu,L)-\cos\theta_K}{1-\cos\theta_K} = \frac{\kappa+\kappa\cos\theta_K+\sqrt{2\kappa(1+\cos\theta_K)}}{1+\kappa\cos\theta_K+\sqrt{2\kappa(1+\cos\theta_K)}}.$$

Applying the mapping $x \mapsto 1-x$ to the previous equation, we get

$$\frac{1-\beta_-(K,\mu,L)}{1-\cos\theta_K} = \frac{1-\kappa}{1+\kappa\cos\theta_K+\sqrt{2\kappa(1+\cos\theta_K)}}.$$

We now need to prove that whatever θ_K is, $\frac{1-\kappa}{1+\kappa\cos\theta_K+\sqrt{2\kappa(1+\cos\theta_K)}} \geqslant \frac{3}{5}$. We note that the LHS converges to 1 when κ goes to 0, hence we know that this inequality holds for κ small enough. We make the precise computation below.

$$\begin{aligned}
\frac{1-\kappa}{1+\kappa\cos\theta_K+\sqrt{2\kappa(1+\cos\theta_K)}} \geqslant \frac{3}{5} &\iff 3(1+\kappa\cos\theta_K+\sqrt{2\kappa(1+\cos\theta_K)}) \leqslant 5(1-\kappa) \\
&\iff 3\sqrt{2\kappa(1+\cos\theta_K)}+\kappa(5+3\cos\theta_K) \leqslant 2 \\
&\impliedby 6\sqrt{\kappa}+8\kappa \leqslant 2 \\
&\iff 1-3\sqrt{\kappa}-4\kappa \geqslant 0 \\
&\iff (1-4\sqrt{\kappa})(1+\sqrt{\kappa}) \geqslant 0 \\
&\iff \sqrt{\kappa} \leqslant \frac{1}{4}.
\end{aligned}$$

Hence, the desired result.

■

Proof of Theorem 8.3.6

Finally, we prove Theorem 8.3.6 :

Theorem 8.3.6. *There exists an absolute constant $C > 0$ (any $C > \frac{50}{3}$), such that for any $0 < \mu < L$, we have:*

$$(\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}))^c \cap \text{SLS}_{\mu,L}\left(\frac{1-C\kappa}{1+C\kappa}\right) = \emptyset. \tag{8.5}$$

Proof. First, we assume that $\kappa \leqslant \left(\frac{3-\sqrt{5}}{4}\right)^2$. Note that in the opposite case, $\sqrt{\kappa} \leqslant (3+\sqrt{5})\kappa \leqslant \frac{50}{3}\kappa$, hence the result would still hold.

Let $(\gamma,\beta) \in (\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}))^c = (\bigcup_{K=2}^{\infty} \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}))^c = \bigcap_{K=2}^{\infty}(\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}))^c$. By Theorem 8.B.4 (since $\kappa \leqslant \left(\frac{3-\sqrt{5}}{4}\right)^2$), for all $K \geqslant 2$, if $\beta \geqslant \beta_-(K,\mu,L)$ (see Notation 8.B.1),

$$\begin{aligned}
\mu\gamma &\leqslant \mu\gamma_-(\beta,K,\mu,L)\\
&= [\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)]\left(1 - \sqrt{1 - \frac{2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K)}{[\beta-\cos\theta_K+\kappa(1-\beta\cos\theta_K)]^2}}\right)\\
&\leqslant [\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)]\left(1 - \left(1 - \frac{2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K)}{[\beta-\cos\theta_K+\kappa(1-\beta\cos\theta_K)]^2}\right)\right)\\
&= \left(\frac{2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K)}{\beta-\cos\theta_K+\kappa(1-\beta\cos\theta_K)}\right)\\
&\leqslant \left(\frac{2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K)}{\beta-\cos\theta_K}\right).
\end{aligned}$$

Finally,

$$\mu\gamma \leqslant \min_{K\geqslant 2}\left(\frac{2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K)}{\beta-\cos\theta_K}\right).$$

We introduce $C_\beta \triangleq \frac{\beta-\cos\frac{2\pi}{K_\beta}}{1-\beta}$ with K chosen so that Lemma 8.B.8 holds. First we know by Lemma 8.B.9 (since $\kappa \leqslant \left(\frac{3-\sqrt{5}}{4}\right)^2 \leqslant \frac{1}{16}$) that $\beta \geqslant \beta_-(K_\beta,\mu,L)$ and then that the previous calculus is valid. Second, we have

$$\begin{aligned}
\mu\gamma &\leqslant \left(2\left(2\beta C_\beta + 1 + 3\beta + \tfrac{1+\beta}{C_\beta}\right)\right)\kappa(1-\beta)\\
&\leqslant 4\left(C_\beta + \frac{1}{C_\beta} + 2\right)\kappa(1-\beta)\\
&\leqslant 4\left(\frac{3}{2} + \frac{2}{3} + 2\right)\kappa(1-\beta)\\
&= \frac{50}{3}\kappa(1-\beta) \qquad (8.27)
\end{aligned}$$

which proves that $\mu\gamma \leqslant \frac{50}{3}\kappa(1-\beta)$. Set $C = 50/3$. By contradiction, we assume that $\rho < \frac{1-C\kappa}{1+C\kappa}$, and additionally $(\gamma,\beta) \in \text{SLS}_{\mu,L}(\rho)$, From Lemma 8.2.4 we know

$$\begin{aligned}
\frac{\frac{1-\kappa}{1+\kappa} - \rho}{\frac{1}{\rho} - \frac{1-\kappa}{1+\kappa}} \leqslant \beta \leqslant \rho^2\\
\mu\gamma \geqslant (1-\rho)(1-\tfrac{\beta}{\rho}).
\end{aligned}$$

And from (8.27), we know that

$$\mu\gamma \leqslant C\kappa(1-\beta).$$

Combining those inequalities, we get $(1-\rho)(1-\frac{\beta}{\rho}) \leqslant C\kappa(1-\beta)$. Rearanging the terms leads to

$$\left(\frac{1-\rho}{\rho} - C\kappa\right)\beta \geqslant 1 - \rho - C\kappa.$$

Besides, $\frac{1-\rho}{\rho} - C\kappa \geqslant 0$ (since $\rho < \frac{1-C\kappa}{1+C\kappa} \leqslant \frac{1}{1+C\kappa}$) and moreover $\beta \in \left[\frac{\frac{1-\kappa}{1+\kappa}-\rho}{\frac{1}{\rho}-\frac{1-\kappa}{1+\kappa}}, \rho^2\right]$, thus $\beta < \rho^2$ we get:

$$\left(\frac{1-\rho}{\rho} - C\kappa\right)\rho^2 \geqslant 1 - \rho - C\kappa.$$

Equivalently $(1-\rho)^2 \leqslant C\kappa(1-\rho^2)$, that is $\rho \geqslant \frac{1-C\kappa}{1+C\kappa}$. Which is in contradiction with our initial assumption $\rho < \frac{1-C\kappa}{1+C\kappa}$. As a conclusion,

$$\rho_{\mathrm{HB}_{\gamma,\beta}} \leqslant \rho \Longrightarrow \rho \geqslant \tfrac{1-C\kappa}{1+C\kappa}.$$

■

8.C Auxiliary proofs from Section 8.4: Proof of Lemma 8.4.12

Lemma 8.4.12. *A matrix $\bar{G}$ is symmetric and circulant such that $\bar{G}\mathbf{1}_K = 0$, if and only is there exist non-negative $\nu_1, \dots, \nu_{\lfloor K/2 \rfloor}$ such that $\bar{G} = \sum_{\ell=1}^{\lfloor K/2 \rfloor} \nu_\ell H_\ell$, with $H_\ell \triangleq \left(\cos\left(\frac{2\pi\ell}{K}|i-j|\right)\right)_{i,j}$.*

Proof. Lemma 8.4.12 The minimal polynomial of J_K is $X^K - 1$, a split polynomial with simple roots in $\mathbb{C}$. Therefore, J_K diagonalizes on $\mathbb{C}$ with eigenvalues $\omega^\ell = e^{2i\pi\ell/K}$, $\ell \in [\![0, K-1]\!]$, and G diagonalizes in the same basis with eigenvalues $\nu_\ell \triangleq \sum_{j=0}^{K-1} c_j \omega^{j\ell}$. The sequence $(\nu_\ell)_{\ell\in[\![0,K-1]\!]}$ is the discrete Fourier transform of $(c_j)_{j\in[\![0,K-1]\!]}$. Therefore, $(c_j)_{j\in[\![0,K-1]\!]}$ is the inverse discrete Fourier transform of $(\nu_\ell)_{\ell\in[\![0,K-1]\!]}$:

$$\forall j \in [\![0, K-1]\!], c_j = \frac{1}{K}\sum_{\ell=0}^{K-1} \nu_\ell \omega^{-j\ell}. \tag{8.28}$$

Moreover,

$$\nu_{K-\ell} = \sum_{j=0}^{K-1} c_j \omega^{j(K-\ell)} = \sum_{j=0}^{K-1} c_j \omega^{-j\ell} = \bar{\nu_\ell},$$

and by symmetry of G, $\bar{\nu_\ell} = \nu_\ell$. Hence, the eigenvalues almost all have an even multiplicity. This excludes ν_0 and $\nu_{K/2}$ if K is even. We can therefore reorder and group terms in (8.28) as

$$\begin{aligned}\forall j \in [\![0, K-1]\!], c_j &= \frac{1}{K}\left[\nu_0 + \sum_{\ell=1}^{\lfloor\frac{K-1}{2}\rfloor} \nu_\ell(\omega^{-j\ell} + \omega^{j\ell}) + \nu_{\frac{K}{2}}\omega^{-j\frac{K}{2}}\right] \\ &\qquad \text{(where the last term drops if } K = 1 \pmod 2\text{)} \\ &= \frac{1}{K}\left[\nu_0 + \sum_{\ell=1}^{\lfloor\frac{K-1}{2}\rfloor} 2\nu_\ell \cos\left(\frac{2\pi jl}{K}\right) + \nu_{\frac{K}{2}}(-1)^j\right]\end{aligned}$$

Finally, G is symmetric positive semi-definite and circulant if and only if there exists

non-negative $(\nu_l)_{l\in[\![0,\lfloor\frac{K}{2}\rfloor]\!]}$ such that

$$G = \sum_{j=0}^{K-1} \frac{1}{K}\left[\nu_0 + \sum_{\ell=1}^{\lfloor\frac{K-1}{2}\rfloor} 2\nu_\ell \cos\left(\frac{2\pi jl}{K}\right) + \nu_{\frac{K}{2}}(-1)^j\right] J_K^j$$
$$= \frac{\nu_0}{K}\underbrace{\sum_{j=0}^{K-1} J_K^j}_{\text{Matrix full of 1s}} + \frac{\nu_{\frac{K}{2}}}{K}\underbrace{\sum_{j=0}^{K-1}(-1)^j J_K^j}_{\text{Checkerboard } \left((-1)^{|i-j|}\right)_{i,j}} + \sum_{\ell=1}^{\lfloor\frac{K-1}{2}\rfloor}\frac{2\nu_\ell}{K}\underbrace{\sum_{j=0}^{K-1}\cos\left(\frac{2\pi jl}{K}\right)J_K^j}_{\left(\cos\left(\frac{2\pi\ell}{K}|i-j|\right)\right)_{i,j}}.$$

The condition $G\mathbf{1}_K = 0$ implies $\nu_0 = 0$. ■

8.D Auxiliary proofs from Section 8.5

8.D.1 Proof of Theorem 8.5.3

Theorem 8.5.3. *Consider $0 < \mu < L$, $K \geqslant 2$, the roots-of-unity cycle $(x_k^\circ)_{k\in[\![0,K-1]\!]}$, and $(\gamma,\beta) \in \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$. Let $(z_t)_{t\in\mathbb{N}}$ be the sequence generated by running (HB)$_{\gamma_t,\beta_t}$, with time varying parameters γ_t, β_t, initialized at $(z_0, z_1) = (x_0^\circ + \delta_0, x_1^\circ + \delta_1)$, with perturbed gradients $\hat{g}_t(z) = \nabla\psi^K_{\gamma,\beta,\mu,L}(z) + \delta_{g_t}$. There exist κ_P and $\rho_D < 1$ (made explicit in the proof) such that if the following three conditions hold,*

1. $\sqrt{\|\delta_0\|^2 + \|\delta_1\|^2} \leqslant \kappa_P r_{\max}$,
2. *for all $t \in \mathbb{N}$,* $\left(\frac{4}{\gamma} + \mu\kappa_P r_{\max}\right)|\delta_{\gamma_t}| + (2 + 2\kappa_P r_{\max})|\delta_{\beta_t}| \leqslant \frac{1}{2}(1-\rho_D)\kappa_P r_{\max}$,
3. *for all $t \in \mathbb{N}$,* $\frac{4}{L}\|\delta_{g_t}\| \leqslant \frac{1}{2}(1-\rho_D)\kappa_P r_{\max}$,

then, $(z_t)_{t\geqslant 0}$ keeps cycling in a neighborhood of $\bigcirc_K$: $\|z_t - x^\circ_{t \pmod K}\| \leqslant r_{\max}$ (and thus $z_t \in \mathcal{V}_{t \pmod K}$).

Proof. We introduce matrices P and D verifying $PDP^{-1} = \begin{pmatrix}(1+\beta)\mathrm{I}_2 - \mu\gamma\mathrm{I}_2 & -\beta\mathrm{I}_2\\ \mathrm{I}_2 & 0\end{pmatrix}$, and such that the operator norm ρ_D of the matrix D, $\rho_D = \|D\|_{\text{op}}$ is smaller than 1, and set $\kappa_P = \frac{1}{\|P\|_{\text{op}}\|P^{-1}\|_{\text{op}}} \leqslant 1$. The existence of such matrices is guaranteed as $(\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$, and we discuss the choice of this reduction in Subsection 8.D.2.

We prove by induction that $\forall t \geqslant 1, \left\|P^{-1}\begin{pmatrix}\delta_t\\ \delta_{t-1}\end{pmatrix}\right\| \leqslant \frac{r_{\max}}{\|P\|_{\text{op}}}$. which implies that $\forall t \geqslant 1, \left\|\begin{pmatrix}\delta_t\\ \delta_{t-1}\end{pmatrix}\right\| \leqslant r_{\max}$, thus the result of the theorem, i.e., $z_t \in B(x^\circ_{t \pmod K}, r_{\max}) \subseteq \mathcal{V}_{t \pmod K}$.

In the proof, we extend again (x_k°) to $k \in \mathbb{N}$ by periodicity, i.e., $(x_k^\circ) = (x^\circ_{k \pmod K})$.

Initialization: $\left\|\begin{pmatrix}\delta_1\\ \delta_0\end{pmatrix}\right\| = \sqrt{\|\delta_0\|^2 + \|\delta_1\|^2} \leqslant \kappa_P r_{\max} = \frac{r_{\max}}{\|P\|_{\text{op}}\|P^{-1}\|_{\text{op}}}$ implies $\left\|P^{-1}\begin{pmatrix}\delta_1\\ \delta_0\end{pmatrix}\right\| \leqslant \frac{r_{\max}}{\|P\|_{\text{op}}}$.

Induction: By induction hypothesis $\left\|P^{-1}\begin{pmatrix}\delta_t\\ \delta_{t-1}\end{pmatrix}\right\| \leqslant \frac{r_{\max}}{\|P\|_{\mathrm{op}}}$, thus $z_t \in \mathcal{V}_{t \pmod K}$. We now write (HB)'s $(t+1)^{\text{th}}$ step:

$$z_{t+1} = z_t - \gamma_t \hat{g}_t(x_t^\circ + \delta_t) + \beta_t(z_t - z_{t-1}) = z_t - \gamma_t(\nabla\psi(x_t^\circ + \delta_t) + \delta_{g_t}) + \beta_t(z_t - z_{t-1}).$$

Using that for all $t \geqslant 0$, $z_{t+1} = x_{t+1}^\circ + \delta_{t+1}$.

$$x_{t+1}^\circ + \delta_{t+1} = x_t^\circ + \delta_t + (\beta + \delta_{\beta_t})(x_t^\circ + \delta_t - x_{t-1}^\circ - \delta_{t-1}) - (\gamma + \delta_{\gamma_t})\nabla\psi(x_t^\circ + \delta_t) - \gamma_t\delta_{g_t}.$$

And as $(\mathrm{HB})_{\gamma,\beta}(\psi)$ cycles on $\bigcirc_K$, we have that $x_{t+1}^\circ = x_t^\circ + \beta(x_t^\circ - x_{t-1}^\circ) - \gamma\nabla\psi(x_t^\circ)$, thus

$$\delta_{t+1} = \delta_t + \delta_{\beta_t}(x_t^\circ - x_{t-1}^\circ) + (\beta + \delta_{\beta_t})(\delta_t - \delta_{t-1}) - (\delta_{\gamma_t})\nabla\psi(x_t^\circ) - (\gamma + \delta_{\gamma_t})\mu\delta_t - \gamma_t\delta_{g_t},$$
$$\delta_{t+1} = \delta_t + \beta(\delta_t - \delta_{t-1}) - \gamma\mu\delta_t + \delta_{\beta_t}(x_t^\circ - x_{t-1}^\circ + \delta_t - \delta_{t-1}) - \delta_{\gamma_t}(\nabla\psi(x_t^\circ) + \mu\delta_t) - \gamma_t\delta_{g_t}.$$

This can be written in an augmented space as

$$\begin{pmatrix}\delta_{t+1}\\ \delta_t\end{pmatrix} = \begin{pmatrix}1+\beta-\mu\gamma & -\beta\\ 1 & 0\end{pmatrix}\begin{pmatrix}\delta_t\\ \delta_{t-1}\end{pmatrix} + \begin{pmatrix}\delta_{\beta_t}(x_t^\circ - x_{t-1}^\circ + \delta_t - \delta_{t-1}) - \delta_{\gamma_t}(\nabla\psi(x_t^\circ) + \mu\delta_t) - \gamma_t\delta_{g_t}\\ 0\end{pmatrix}.$$

The second and third assumptions corresponds on the perturbations enable to write that

$$\begin{aligned}
&\left\|\begin{pmatrix}\delta_{\beta_t}(x_t^\circ - x_{t-1}^\circ + \delta_t - \delta_{t-1}) - \delta_{\gamma_t}(\nabla\psi(x_t^\circ) + \mu\delta_t - \gamma_t\delta_{g_t})\\ 0\end{pmatrix}\right\|\\
&\quad= \|\delta_{\beta_t}(x_t^\circ - x_{t-1}^\circ + \delta_t - \delta_{t-1}) - \delta_{\gamma_t}(\nabla\psi(x_t^\circ) + \mu\delta_t) - \gamma_t\delta_{g_t}\|\\
&\quad\leqslant |\delta_{\beta_t}|\|x_t^\circ - x_{t-1}^\circ + \delta_t - \delta_{t-1}\| + |\delta_{\gamma_t}|\|\nabla\psi(x_t^\circ) + \mu\delta_t\| + \gamma_t\|\delta_{g_t}\|\\
&\quad= \left(\frac{\sqrt{(1+\beta)^2(1-\cos(\theta_K))^2 + (1-\beta)^2\sin(\theta_K)^2}}{\gamma} + \mu\kappa_P r_{\max}\right)|\delta_{\gamma_t}|\\
&\qquad+ \left(\sqrt{(1-\cos(\theta_K))^2 + \sin(\theta_K)^2} + 2\kappa_P r_{\max}\right)|\delta_{\beta_t}| + \frac{4}{L}\|\delta_{g_t}\|\\
&\quad\leqslant \left(\frac{4}{\gamma} + \mu\kappa_P r_{\max}\right)|\delta_{\gamma_t}| + (2 + 2\kappa_P r_{\max})\,|\delta_{\beta_t}| + \frac{4}{L}\|\delta_{g_t}\|\\
&\quad\leqslant \frac{1}{2}(1-\rho_D)\kappa_P r_{\max} + \frac{1}{2}(1-\rho_D)\kappa_P r_{\max} = (1-\rho_D)\kappa_P r_{\max}.
\end{aligned}$$

We now have

$$\begin{aligned}
\begin{pmatrix}\delta_{t+1}\\ \delta_t\end{pmatrix} &= \begin{pmatrix}1+\beta-\mu\gamma & -\beta\\ 1 & 0\end{pmatrix}\begin{pmatrix}\delta_t\\ \delta_{t-1}\end{pmatrix} + \begin{pmatrix}\delta_{\beta_t}(x_t^\circ - x_{t-1}^\circ + \delta_t - \delta_{t-1}) - \delta_{\gamma_t}(\nabla\psi(x_t^\circ) + \mu\delta_t)\\ 0\end{pmatrix}\\
&= PDP^{-1}\begin{pmatrix}\delta_t\\ \delta_{t-1}\end{pmatrix} + \begin{pmatrix}\delta_{\beta_t}(x_t^\circ - x_{t-1}^\circ + \delta_t - \delta_{t-1}) - \delta_{\gamma_t}(\nabla\psi(x_t^\circ) + \mu\delta_t)\\ 0\end{pmatrix}\\
P^{-1}\begin{pmatrix}\delta_{t+1}\\ \delta_t\end{pmatrix} &= DP^{-1}\begin{pmatrix}\delta_t\\ \delta_{t-1}\end{pmatrix} + P^{-1}\begin{pmatrix}\delta_{\beta_t}(x_t^\circ - x_{t-1}^\circ + \delta_t - \delta_{t-1}) - \delta_{\gamma_t}(\nabla\psi(x_t^\circ) + \mu\delta_t)\\ 0\end{pmatrix}\\
\left\|P^{-1}\begin{pmatrix}\delta_{t+1}\\ \delta_t\end{pmatrix}\right\| &\leqslant \|D\|\left\|P^{-1}\begin{pmatrix}\delta_t\\ \delta_{t-1}\end{pmatrix}\right\| + \left\|P^{-1}\begin{pmatrix}\delta_{\beta_t}(x_t^\circ - x_{t-1}^\circ + \delta_t - \delta_{t-1}) - \delta_{\gamma_t}(\nabla\psi(x_t^\circ) + \mu\delta_t)\\ 0\end{pmatrix}\right\|\\
&\leqslant \rho_D\left\|P^{-1}\begin{pmatrix}\delta_t\\ \delta_{t-1}\end{pmatrix}\right\| + \|P^{-1}\|_{\mathrm{op}}(1-\rho_D)\kappa_P r_{\max}\\
&\overset{\text{by induction}}{\leqslant} \rho_D\frac{r_{\max}}{\|P\|_{\mathrm{op}}} + (1-\rho_D)\frac{r_{\max}}{\|P\|_{\mathrm{op}}}\\
&= \frac{r_{\max}}{\|P\|_{\mathrm{op}}},
\end{aligned}$$

thereby reaching the desired claim. ■

8.D.2 Discussion about the reduction made in the proof of Theorem 8.5.3

In Theorem 8.5.3, we decompose the matrix $\begin{pmatrix} 1+\beta-\mu\gamma & -\beta \\ 1 & 0 \end{pmatrix}$ into the form PDP^{-1} with $\|D\| < 1$. In this section, we discuss a possible way to do it depending on the region in which (γ, β) lies (see Proposition 8.2.1). In the 3 possible cases, we provide a decomposition PDP^{-1} and $\rho_D = \|D\| < 1$. To compute $\kappa_P = \frac{1}{\|P\|_{\text{op}}\|P^{-1}\|_{\text{op}}}$, we can use the fact that κ_P^2 is the ratio of the 2 eigenvalues of the matrix P^TP:

$$\kappa_P = \left(\frac{\frac{\mathrm{Tr}(P^TP)}{2} - \sqrt{\left(\frac{\mathrm{Tr}(P^TP)}{2}\right)^2 - \mathrm{Det}(P^TP)}}{\frac{\mathrm{Tr}(P^TP)}{2} + \sqrt{\left(\frac{\mathrm{Tr}(P^TP)}{2}\right)^2 - \mathrm{Det}(P^TP)}} \right)^{1/2} = \left(\frac{\left(\frac{\mathrm{Tr}(P^TP)}{2} - \sqrt{\left(\frac{\mathrm{Tr}(P^TP)}{2}\right)^2 - \mathrm{Det}(P^TP)}\right)^2}{\mathrm{Det}(P^TP)} \right)^{1/2}$$

$$= \frac{\frac{\mathrm{Tr}(P^TP)}{2} - \sqrt{\left(\frac{\mathrm{Tr}(P^TP)}{2}\right)^2 - \mathrm{Det}(P)^2}}{|\mathrm{Det}(P)|} = \frac{\mathrm{Tr}(P^TP)}{2|\mathrm{Det}(P)|} - \sqrt{\left(\frac{\mathrm{Tr}(P^TP)}{2|\mathrm{Det}(P)|}\right)^2 - 1}.$$

Lazy region ($\gamma < \frac{(1-\sqrt{\beta})^2}{\mu}$). In this region,

$$\begin{pmatrix} 1+\beta-\mu\gamma & -\beta \\ 1 & 0 \end{pmatrix} \sim D = \begin{pmatrix} \frac{1+\beta-\mu\gamma}{2} + \sqrt{\left(\frac{1+\beta-\mu\gamma}{2}\right)^2 - \beta} & 0 \\ 0 & \frac{1+\beta-\mu\gamma}{2} - \sqrt{\left(\frac{1+\beta-\mu\gamma}{2}\right)^2 - \beta} \end{pmatrix}$$

with the transition matrix $P = \begin{pmatrix} \frac{1+\beta-\mu\gamma}{2} + \sqrt{\left(\frac{1+\beta-\mu\gamma}{2}\right)^2 - \beta} & \frac{1+\beta-\mu\gamma}{2} - \sqrt{\left(\frac{1+\beta-\mu\gamma}{2}\right)^2 - \beta} \\ 1 & 1 \end{pmatrix}$.

We then obtain $\rho_D = \|D\|_{\text{op}} = \frac{1+\beta-\mu\gamma}{2} + \sqrt{\left(\frac{1+\beta-\mu\gamma}{2}\right)^2 - \beta} < 1$.

Robust region ($\gamma > \frac{(1-\sqrt{\beta})^2}{\mu}$). In this region,

$$\begin{pmatrix} 1+\beta-\mu\gamma & -\beta \\ 1 & 0 \end{pmatrix} \sim D = \begin{pmatrix} \frac{1+\beta-\mu\gamma}{2} + i\sqrt{\beta - \left(\frac{1+\beta-\mu\gamma}{2}\right)^2} & 0 \\ 0 & \frac{1+\beta-\mu\gamma}{2} - i\sqrt{\beta - \left(\frac{1+\beta-\mu\gamma}{2}\right)^2} \end{pmatrix}$$

with the transition matrix $P = \begin{pmatrix} \frac{1+\beta-\mu\gamma}{2} + i\sqrt{\beta - \left(\frac{1+\beta-\mu\gamma}{2}\right)^2} & \frac{1+\beta-\mu\gamma}{2} - i\sqrt{\beta - \left(\frac{1+\beta-\mu\gamma}{2}\right)^2} \\ 1 & 1 \end{pmatrix}$.

We then obtain $\rho_D = \|D\|_{\text{op}} = \sqrt{\beta} < 1$.

Boundary ($\gamma = \frac{(1-\sqrt{\beta})^2}{\mu}$). On the boundary between the lazy and the robust regions, $\begin{pmatrix} 1+\beta-\mu\gamma & -\beta \\ 1 & 0 \end{pmatrix}$ is not diagonalisable.

However, we can write $\begin{pmatrix} 1+\beta-\mu\gamma & -\beta \\ 1 & 0 \end{pmatrix} \sim D = \sqrt{\beta}\begin{pmatrix} 1 & \varepsilon \\ 0 & 1 \end{pmatrix}$, with any $\varepsilon > 0$ using the transition matrix $P = \begin{pmatrix} \sqrt{\beta} & \frac{\varepsilon\sqrt{\beta}}{1+\beta} \\ 1 & -\frac{\beta\varepsilon}{1+\beta} \end{pmatrix}$. We then obtain $\rho_D = \|D\| = \sqrt{\beta}\sqrt{1 + \frac{\varepsilon^2}{2} + \sqrt{\left(1+\frac{\varepsilon^2}{2}\right)^2 - 1}} = \sqrt{\beta}\left(\frac{\varepsilon}{2} + \sqrt{1+\frac{\varepsilon^2}{4}}\right)$. Note $\rho_D < 1$ if and only if $\varepsilon < \frac{1-\beta}{\sqrt{\beta}}$.

8.E Auxiliary proofs from Section 8.6

Definition 8.E.1 (Mollifier u_ε, $\varepsilon > 0$)**.** *We define, for any* $\varepsilon > 0$.

$$u(x) \triangleq \frac{1}{Z} e^{-\frac{1}{1-\|x\|^2}} \mathbf{1}_{\|x\|<1}, \qquad \text{with } Z \triangleq \int_{\|x\|\leqslant 1} e^{-\frac{1}{1-\|x\|^2}} \mathrm{dx}, \quad u_\varepsilon(x) \triangleq \frac{1}{\varepsilon^2} u\left(\frac{1}{\varepsilon} x\right).$$

We first recall some classical properties of u_ε, see for example (Section 4.4 of Brézis, 2011, on *mollifiers*).

Lemma 8.E.2. *$\forall \varepsilon > 0, u_\varepsilon$ is the probability density function (pdf) of an L^∞ and centered random variable.*

Lemma 8.E.3 (C^∞ with compact support)**.** *For any $\varepsilon > 0$, $u_\varepsilon \in C^\infty$ and its support is $B(0, \varepsilon)$.*

Lemma 8.E.4 (Bounded derivatives)**.** *For any $\varepsilon > 0$ and any $r \geqslant 0$,*

$$M_\varepsilon^{(r)} \triangleq \sup_{x\in\mathbb{R}^2} \|\nabla^r u_\varepsilon(x)\| = \sup_{x\in B(0,\varepsilon)} \|\nabla^r u_\varepsilon(x)\| < \infty.$$

Let $\psi \in \mathcal{F}_{\mu,L}$. We consider $\varphi_\varepsilon = \psi * u_\varepsilon$. We recall the following properties of φ_ε.

Lemma 8.E.5 (Higher-order derivatives are bounded)**.** *Let $\psi \in \mathcal{F}_{\mu,L}$ and $\varphi_\varepsilon = \psi * u_\varepsilon$. Then for any $\varepsilon > 0$,*

1. *$\varphi_\varepsilon \in C^\infty$*
2. *for any $r \geqslant 2$ $\|\nabla^r(\varphi_\varepsilon)\|$ is bounded.*
3. *$\varphi_\varepsilon \in \mathcal{F}_{\mu,L}$.*

The proof of point 1 is standard, the one of point 2 uses that $\forall x, \|\nabla^2\psi(x)\| \leqslant L$ and the properties of the convolution, and finally, the proof of point 3 relies on the fact that $\forall\, x, \mu \leqslant \|\nabla^2\psi(x)\| \leqslant L$ and that u_ε is a probability density function.

8.F A summary of convergence rates on $\mathcal{F}_{\mu,L}$ and $\mathcal{Q}_{\mu,L}$

For completeness and reference purposes, we summarize the convergence rates of the algorithms under consideration of this work, on the functional classes $\mathcal{F}_{\mu,L}$ and $\mathcal{Q}_{\mu,L}$. Section 8.F provides the algorithms and their respective worst-case converge rates in the sense of Definition 8.1.3 expressed uniformly over κ, and approximately as $\kappa \to 0$.

Table 8.3: Asymptotic convergence rates (on $\|x_t - x_\star\|$) for standard algorithms. Optimal convergence rates (lower and upper complexity bounds) are highlighted in boxes.

Algorithm	Known convergence rate		Approximate rate as $\kappa \to 0$	
	on $\mathcal{F}_{\mu,L}$	on $\mathcal{Q}_{\mu,L}$	on $\mathcal{F}_{\mu,L}$	on $\mathcal{Q}_{\mu,L}$
GD($\gamma = 1/L$)	$1-\kappa$ (see Nesterov (2003))	$1-\kappa$ (see Nesterov (2003))	$1-\kappa$	$1-\kappa$
GD($\gamma = 2/(L+\mu)$)	$\frac{1-\kappa}{1+\kappa}$ (see Nesterov (2003))	$\frac{1-\kappa}{1+\kappa}$ (see Nesterov (2003))	$1-2\kappa$	$1-2\kappa$
Chebyshev's method	?	$\boxed{\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}}$ (see Nemirovskii (1994))	?	$\boxed{1-2\sqrt{\kappa}}$
HB($\gamma^\star(\mathcal{Q}_{\mu,L}), \beta^\star(\mathcal{Q}_{\mu,L})$)	none (cycles) (see Lessard et al. (2016))	$\boxed{\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}}$ (see Nemirovskii (1994))	none (cycles)	$\boxed{1-2\sqrt{\kappa}}$
HB($\gamma^\star(\mathcal{F}_{\mu,L}), \beta^\star(\mathcal{F}_{\mu,L})$)	$1-\Theta(\kappa)$ (Corollary 8.3.7)	$1-\Theta(\kappa)$ (Theorem 8.3.6)	$1-\Theta(\kappa)$	$1-\Theta(\kappa)$
NAG ($\gamma = 1/L, \beta = \frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}$)	$(1-\sqrt{\kappa})^{1/2}$ (see Nesterov (2003))	$1-\sqrt{\kappa}$ (see Hagedorn and Jarre (2023))	$1-\frac{1}{2}\sqrt{\kappa}$	$1-\sqrt{\kappa}$
Information-theoretic exact method	$\boxed{1-\sqrt{\kappa}}$ (see (Taylor and Drori, 2022))	$1-\sqrt{\kappa}$ (see (Taylor and Drori, 2022))	$\boxed{1-\sqrt{\kappa}}$	$1-\sqrt{\kappa}$
Triple momentum method	$\boxed{1-\sqrt{\kappa}}$ (see (Van Scoy et al., 2017))	$1-\sqrt{\kappa}$ (see (Van Scoy et al., 2017))	$\boxed{1-\sqrt{\kappa}}$	$1-\sqrt{\kappa}$
Lower complexity bounds	$\boxed{1-\sqrt{\kappa}}$ (see Drori and Taylor (2022))	$\boxed{\frac{1-\sqrt{\kappa}}{1+\sqrt{\kappa}}}$ (see Nemirovskii (1994))	$\boxed{1-\sqrt{\kappa}}$	$\boxed{1-2\sqrt{\kappa}}$

Part III

Conclusion

9

Summary

In the ever-evolving landscape of optimization theory, which many applications such as machine learning rely on, the thesis, "Constructive Approaches to Worst-Case Complexity Analyses of Gradient Methods for Convex Optimization: Contributions, New Insights, and New Results," stands as a testament to the indispensable role of the uniformization and automation of the approaches.

Amidst the myriad optimization approaches, first-order optimization methods exhibit a good balance between efficacy and computational efficiency of recent emerging problems. At its core, this thesis is a quest to not merely find solution to optimization problems but to accelerate their discoveries by constructively contribute to the theoretical foundations of first-order optimization. The contributions of this thesis take diverse forms, from comprehending the insights that existing approaches bring to the field and developing new approaches, to using them to solve specific open problems. This is facilitated through the development of the Python package PEPIT, designed to support further investigations.

The journey undertaken within these pages begins by analyzing the known link between quadratic optimization and polynomials. We developed equioscillation theorems and used them to design the worst-case optimal method under an empirically observed assumption. This result closed the gap between theory and practice, explaining the empirical performance of the corresponding strategy.

One of the primary contributions lies in the synthesis of many insights brought by the *Performance Estimation Framework* about proof structures. By developing the supporting Python package PEPIT, the thesis not only advances the theoretical discourse but also provides practitioners with tangible tools for the constructive exploration of optimization proofs. The accompanying documentation and the tutorial serve as a beacon, guiding researchers on how to derive natural proofs in optimization.

The thesis also applies these methodologies to a large class of non necessarily smooth functions to derive several algorithms and tight worst-case guarantees.

Finally, while existing approach guide the design of proofs of convergence, the thesis introduces a new constructive approach to disprove convergence of first-order methods. Moreover, this new approach can easily be embedded in the Python package PEPIT. This not only broadens the scope of understanding first-order optimization but also enriches the theoretical arsenal available to researchers and practitioners. The journey culminates in a critical examination of the Heavy-ball method, challenging the conventional wisdom by using the theoretical Performance Estimation framework to disprove its acceleration over the class of smooth and strongly convex functions.

In summary, rather than exclusively addressing particular open problems, this thesis introduces and examines general tools and concrete elements that underscore the significance and effectiveness of employing uniform approaches in the exploration of optimization methods.

10

A few open directions

Contents

10.1 Analysis of bilinear games via polynomials.

The equivalence theorem between algorithms and families of polynomials can be adapted to games.

Indeed, assume we seek to converge to the saddle point of the bilinear function of (x, y)

$$(x - x_\star)^T A(y - y_\star), \tag{10.1}$$

that is we want to find (x, y) such that $A(y - y_\star) = 0$ and $A^T(x - x_\star) = 0$.

A well-known fact is that the simultaneous Gradient descent ascent method does not converge well in this setting. Indeed, the latter consists of the following updates

$$x_{t+1} = x_t - \gamma A(y_t - y_\star) \tag{10.2}$$
$$y_{t+1} = y_t + \gamma A^T(x_t - x_\star) \tag{10.3}$$

for some positive γ, and we can therefore verify

$$\|A(y_{t+1} - y_\star)\|^2 + \|A^T(x_{t+1} - x_\star)\|^2 \tag{10.4}$$
$$= \|A(y_t - y_\star - \gamma A^T(x_t - x_\star))\|^2 + \|A^T(x_t - x_\star + \gamma A(y_t - y_\star))\|^2 \tag{10.5}$$
$$= \|A(y_t - y_\star)\|^2 - 2\gamma \cancel{\left\langle A(y_t - y_\star), AA^T(x_t - x_\star) \right\rangle} + \|AA^T(x_t - x_\star)\|^2 \tag{10.6}$$
$$+ \|A^T(x_t - x_\star)\|^2 + 2\gamma \cancel{\left\langle A^T(x_t - x_\star), A^T A(y_t - y_\star) \right\rangle} + \|A^T A(y_t - y_\star)\|^2 \tag{10.7}$$
$$\geqslant \|A(y_t - y_\star)\|^2 + \|A^T(x_t - x_\star)\|^2. \tag{10.8}$$

To overcome this issue, methods like *extragradient* (Korpelevich, 1976; Gidel et al., 2018) and *negative-momentum* (Gidel et al., 2019) have been created. Extragradient has recently been studied through the prism of polynomials theory (Kim et al., 2022). However, to the best of my knowledge, there does not exist an equivalent to the conjugate gradient method for this type of problem. We want to extend the reasoning we used in Chapter 2 to this particular problem.

Let us consider a generic form of algorithms as

$$x_t = x_0 - \sum_{s=0}^{t-1} \gamma_{t,s}^{(x)} A(y_s - y_\star) \tag{10.9}$$
$$y_t = y_0 + \sum_{s=0}^{t-1} \gamma_{t,s}^{(y)} A^T(x_t - x_\star) \tag{10.10}$$

We can define 4 polynomials $P_t^{(x,x)}$, $P_t^{(x,y)}$, $P_t^{(y,x)}$ and $P_t^{(y,y)}$ such that for all $t \geqslant 0$,

$$x_t - x_\star = P_t^{(x,x)}(AA^T)(x_0 - x_\star) + AP_t^{(x,y)}(A^T A)(y_0 - y_\star), \tag{10.11}$$
$$y_t - y_\star = A^T P_t^{(y,x)}(AA^T)(x_0 - x_\star) + P_t^{(y,y)}(A^T A)(y_0 - y_\star). \tag{10.12}$$

This way, we could try to minimize a given metric in a greedy way using properties of orthogonal polynomials.

Challenge. The difficulty here lies in the fact the 4 polynomials are not independent of each other, therefore the degree of freedom is limited. Considering the interlaced polynomials $P_t^{(x,x)}(X^2) + XP^{(x,y)}(X^2)$ and $P_t^{(y,y)}(X^2) + XP^{(y,x)}(X^2)$ seems to be the key to solve this issue.

10.2 Non-quadratic PEP constraints.

An essential ingredient for PEPs to be transformed into an SDP is the homogeneity of the different formulas. In particular, the fact that the interpolation constraints of the studied class of functions write linearly in function values and quadratically in points and gradients. For example, the classes of smooth, smooth convex, quadratically bounded, and quadratically bounded convex functions verify the respective inequalities for all i, j:

$$\begin{aligned}
f_i - f_j &\leqslant \langle g_j, x_i - x_j\rangle + \frac{L}{2}\|x_i - x_j\|^2,\\
f_i - f_j &\geqslant \langle g_j, x_i - x_j\rangle + \frac{1}{2L}\|g_i - g_j\|^2,\\
f_i - f_\star &\leqslant \frac{L}{2}\|x_i - x_\star\|^2,\\
\begin{cases} f_\star - f_i &\geqslant \langle g_i, x_\star - x_i\rangle + \frac{1}{2L}\|g_i\|^2,\\ f_i - f_j &\geqslant \langle g_j, x_i - x_j\rangle\,. \end{cases}
\end{aligned}$$

All the LHS are linear expressions of the function values f_i and f_j, while all the RHS are quadratic expressions of the points x_i and x_j and the gradients g_i and g_j. After introducing the Gram matrix of all x and g, all the inequalities become *linear*.

Note however that this excludes classes such as Holder continuous functions verifying

$$f_i - f_j \leqslant \langle g_j, x_i - x_j\rangle + \frac{L}{1+p}\|x_i - x_j\|^{1+p} \tag{10.13}$$

for some $p \in (0, 1)$. This also excludes the corresponding generalization of $\mathrm{QG}^+(L)$:

$$f_i - f_\star \leqslant \frac{L}{1+p}\|x_i - x_\star\|^{1+p}. \tag{10.14}$$

Indeed, even after the SDP lifting step, the inequality contains some entries of the Gram matrix elevated to the power $\frac{1+p}{2}$, which makes the problem non-linear when $p \neq 1$.

However, some tricks may sometimes exist to study classes of functions that seem to be out of the scope of the PEP framework (see e.g., Dragomir (2021); Dragomir et al. (2021); Dragomir and Nesterov (2023)). An promising trick consists in replacing some scalar inequalities with linear matrix inequalities (LMI). For the sake of simplicity, let's define $E_L \triangleq f_i - f_j - \langle g_j, x_i - x_j\rangle$ or $E_L \triangleq f_i - f_\star$ and $E_R \triangleq \left(\frac{L}{1+p}\right)^{1/(1+p)}\|x_i - x_\star\|^2$. E_L and E_R both are linear expressions of the function values and the entries of the Gram matrix introduced during the SDP lifting step. The inequalities (10.13) and (10.14) write as $E_L \leqslant E_R^{(1+p)/2}$. As mentioned above, in some cases, introducing an LMI helps overcome this issue. The simplest example corresponds to $p = 0$. In this case, we aim at linearizing $E_L \leqslant E_R^{1/2}$. Note that the latter is equivalent to $E_R - E_L^2 \geqslant 0$, or equivalently to $\begin{pmatrix} E_R & E_L \\ E_L & 1 \end{pmatrix} \succcurlyeq 0$. Note this trick of transforming a quadratic expression into an LMI has been used in (De Klerk et al., 2017) to study the Gradient descent method with exact line-search. This way, we can afford to authorize the power $1/2$. Subsequently, by iterating this trick, we can also authorize the power $3/4$ and all the $q/2^n$ as well, where q and n are positive integers with $q/2^n \in (0, 1)$.

Structure of the proof of worst-case guarantee. Each time we introduce a 2×2 LMI of the form $\begin{pmatrix} A & C \\ C & B \end{pmatrix} \succcurlyeq 0$, we obtain the optimal primal value $P = \begin{pmatrix} P_A & P_C \\ P_C & P_B \end{pmatrix} \succcurlyeq 0$, and the

optimal dual value $D = \begin{pmatrix} D_A & D_C \\ D_C & D_B \end{pmatrix} \succcurlyeq 0$. Writing a proof under the form (Generic proof), the above LMI is involved linearly using only $\left\langle \begin{pmatrix} A & C \\ C & B \end{pmatrix}, \begin{pmatrix} D_A & D_C \\ D_C & D_B \end{pmatrix} \right\rangle \geqslant 0$. Note the KKT condition gives $\left\langle \begin{pmatrix} P_A & P_C \\ P_C & P_B \end{pmatrix}, \begin{pmatrix} D_A & D_C \\ D_C & D_B \end{pmatrix} \right\rangle = 0$. Then, except when $P = 0$ (which cannot happen in many examples like above when $P_B = 1$), the matrix B is not full rank. Therefore, we know that $D_C^2 = D_A D_B$. Finally, $\left\langle \begin{pmatrix} P_A & P_C \\ P_C & P_B \end{pmatrix}, \begin{pmatrix} D_A & D_C \\ D_C & D_B \end{pmatrix} \right\rangle = AD_A + BD_B \pm 2C\sqrt{D_A D_B}$. As a conclusion, when we have access to the 2×2 LMI that we can use in the proof, we actually only use one scalar inequality of the form $\pm C \leqslant \frac{1}{2}(XA + B/X)$ where $X = \sqrt{D_A/D_B}$.

Example. Considering the non-linear inequality $E_L \leqslant E_R^{1/2}$ as above, we can introduce the LMI $\begin{pmatrix} E_R & E_L \\ E_L & 1 \end{pmatrix} \succcurlyeq 0$ to obtain an SDP, easily solvable numerically. On the other hand, to reconstruct the proof, it is important to understand beforehand the kind of structure we are seeking. Here, the proof will only use some inequality $E_L \leqslant \frac{1}{2}(XE_R + 1/X)$ for some positive scalar X, which could be obtained by using Young inequality.

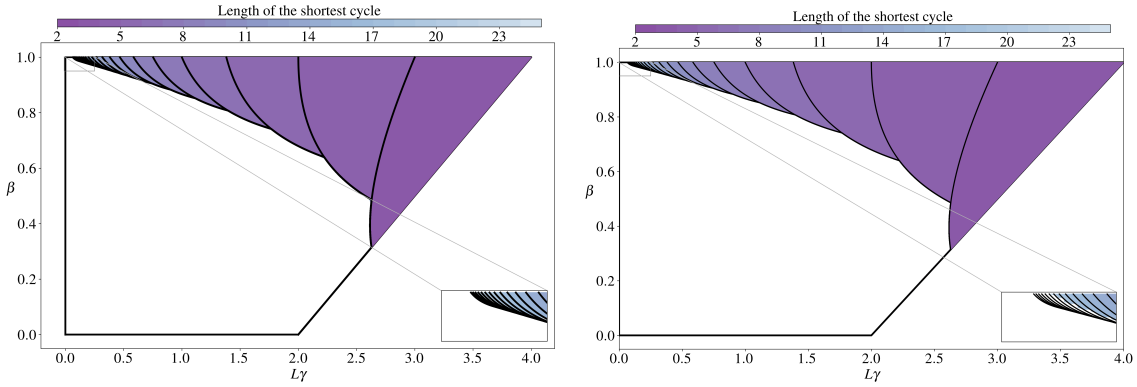
Generalization. Instead of considering the inequality $E_L \leqslant E_R^{1/2}$, we can now look at inequalities of the form $E_L \leqslant H(E_R)$ where H is a function. A generalization of the Young inequality applied to the function $-H$ states $\forall Z, Y,\ ZY \leqslant (-H)(Y) + (-H)^*(Z)$. With $Y := E_R$ and $Z := -X$, we have for any X: $E_L \leqslant H(E_R) \leqslant XE_R + (-H)^*(-X)$, allowing to only deal with affine combinations of the usual PEP terms. In particular, this allows us to derive certificates that hold on Holder continuous convex functions, rediscovering the universal methods described in Nesterov (2015), but also to generalize this methodology to other classes. In particular, in Chapter 6, we derive guarantees of first-order methods on $\mathrm{QG}^+(L)$ convex functions, and this can be generalized to other types of bounds on f such as (10.14).

Challenge. Understand when this method is guaranteed to deliver optimal guarantees and when this is not the case. In particular, when H is concave, the Young inequalities applied on $-H$ lead to upper bounding H by all its tangents, which perfectly characterize H. On the other hand, when H is not concave, this method applies on H as it applies on $-(-H)^{**} \geqslant H$, that is this method does not fully characterize H. But is it possible to do better?

10.3 HB on $\mathcal{F}_{\mu,L}$

In Chapter 8, we found the analytic expression of the region $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ of parameters for which HB admits a cycle on $\mathcal{F}_{\mu,L}$, excluding the possibility of an acceleration. Moreover, we proved that, for each parameter of this region, we can find a cycle in a two-dimensional space. Therefore, some questions remains open:

1. Can we find cycles in dimension one for any parameter on which HB admits a cycle?
2. What is the analytical expression of $\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L})$?



(a) Border of $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ in black lines. $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$ in shades of purples.

(b) Border of $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ in black lines. $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})(D=1)$ in shades of purples.

Figure 10.1: Comparison of cycle regions in different dimensions.

3. What is the optimal convergence rate of HB on $\mathcal{F}_{\mu,L}$?

Those questions are discussed in the 3 following sections.

10.3.1 Does HB accelerate in dimension 1?

Here we seek the set of parameters (γ, β) such that $\text{HB}_{\gamma,\beta}$ cycles on a one-dimensional function.

First note that, with the notation of Chapter 8,

$$\left.\begin{matrix}\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})(D=1)\\ \Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})\end{matrix}\right\} \subseteq \Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})(D=2) \subseteq \Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L}). \tag{10.15}$$

Conjecture 8.4.15 states

$$\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}) = \Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})(D=2) = \Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L}). \tag{10.16}$$

In the following, we conjecture

$$\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})(D=1) = \Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})(D=2) = \Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L}), \tag{10.17}$$

i.e. $\text{HB}_{\gamma,\beta}$ cycles on a function (without dimension constraint) if and only if it cycles in dimension 1. This would imply that $\text{HB}_{\gamma,\beta}$ cannot accelerate, even in dimension 1.

Here is how those conjectures are supported:

- First, Theorem 8.3.5 provides analytical formulas for $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$.
- Second, Theorem 8.4.13 provides an efficient way to numerically obtain $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$ by solving an LP. This way, we can numerically compare $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ and $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})$, as shown in Figure 10.1a.
- Finally, in the following, we provide a way to numerically obtain $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})(D=1)$, and we also compare $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ and $\Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})(D=1)$ in Figure 10.1b.

Finding 1D cycles. Let $K \geqslant 3$. Let $(x_0, \cdots, x_{K-1}) \in \mathbb{R}^K$ a cycle. We note $X = (x_0, \cdots, x_{K-1})^\top \in \mathbb{R}^K$. $\text{HB}_{\gamma,\beta}$ cycles if and only if there exists a function with gradients $G = (g_0, \cdots, g_{K-1}) \in \mathbb{R}^K$ verifying $G = \frac{[(1+\beta)I - J - \beta J^{-1}]}{\gamma} X$.

In dimension 1, verifying that the underlying function belongs to $\mathcal{F}_{\mu,L}$ is equivalent to verifying $\mu \leqslant \frac{g^{(i+1)} - g^{(i)}}{x^{(i+1)} - x^{(i)}} \leqslant L$ where the exponents sort X, i.e. there exists a permutation σ such that $X = \sigma(x^{(0)}, \cdots, x^{(K-1)})^\top$ with $x^{(0)} \leqslant \cdots \leqslant x^{(K-1)}$, and we define $(g^{(0)}, \cdots, g^{(K-1)})^T$ with $G = \sigma(g^{(0)}, \cdots, g^{(K-1)})^T$. Note that σ is defined as the permutation that sorts X, and it sorts also G if and only if f is convex.

Therefore, defining $\delta_k \triangleq x^{(k)} - x^{(k-1)} \geqslant 0$ for any $k \in [\![1, K-1]\!]$, we have

$$X = \sigma \begin{pmatrix} X^{(0)} \\ \vdots \\ X^{(K-1)} \end{pmatrix} = \sigma S \begin{pmatrix} \delta_1 \\ \vdots \\ \delta_{K-1} \end{pmatrix}, \qquad \text{with } S \triangleq \begin{pmatrix} 0 & 0 & \dots & 0 \\ 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ 1 & 1 & \dots & 1 \end{pmatrix} \in \mathbb{R}^{K \times (K-1)}.$$

Note that we simplified w.l.o.g choosing $X^{(0)} = 0$, which reduces the dimension of the problem by 1. We define the desired gradients as

$$G = \frac{[(1+\beta)I - J - \beta J^{-1}]}{\gamma} X = \frac{[(1+\beta)I - J - \beta J^{-1}]}{\gamma} \sigma S \begin{pmatrix} \delta_1 \\ \vdots \\ \delta_{K-1}. \end{pmatrix},$$

and

$$\begin{pmatrix} g^{(1)} - g^{(0)} \\ \vdots \\ g^{(K-1)} - g^{(K-2)} \end{pmatrix} = D\sigma^{-1} G, \qquad \text{with } D \triangleq \begin{pmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & -1 & 1 & 0 \\ 0 & 0 & \dots & 0 & -1 & 1 \end{pmatrix} \in \mathbb{R}^{(K-1) \times K}.$$

Finally,

$$\begin{pmatrix} g^{(1)} - g^{(0)} \\ \vdots \\ g^{(K-1)} - g^{(K-2)} \end{pmatrix} = D\sigma^{-1} \frac{[(1+\beta)I - J - \beta J^{-1}]}{\gamma} \sigma S \begin{pmatrix} \delta_1 \\ \vdots \\ \delta_{K-1}. \end{pmatrix}.$$

And we know that $f \in \mathcal{F}_{\mu,L}$ if and only if

$$\begin{pmatrix} g^{(1)} - g^{(0)} \\ \vdots \\ g^{(K-1)} - g^{(K-2)} \end{pmatrix} \leqslant L \begin{pmatrix} \delta_1 \\ \vdots \\ \delta_{K-1}. \end{pmatrix},$$

$$\begin{pmatrix} g^{(1)} - g^{(0)} \\ \vdots \\ g^{(K-1)} - g^{(K-2)} \end{pmatrix} \geqslant \mu \begin{pmatrix} \delta_1 \\ \vdots \\ \delta_{K-1}. \end{pmatrix},$$

i.e. if and only if

$$D\sigma^{-1}[(1+\beta-L\gamma)I-J-\beta J^{-1}]\sigma S\begin{pmatrix}\delta_1\\ \vdots\\ \delta_{K-1}.\end{pmatrix}\leqslant\vec{0},$$
$$D\sigma^{-1}[(1+\beta-\mu\gamma)I-J-\beta J^{-1}]\sigma S\begin{pmatrix}\delta_1\\ \vdots\\ \delta_{K-1}.\end{pmatrix}\geqslant\vec{0}.$$

The cycle is parametrized by $\vec{\delta}$. We need $\vec{\delta}\geqslant\vec{0}$ and $\vec{\delta}\neq\vec{0}$. Indeed $\vec{\delta}=\vec{0}$ leads to a single point cycle and we know it corresponds to the optimizer being a fixed point.

Finally, we conclude that the existence of a cycle is equivalent to the existence of a permutation σ and a vector $\vec{\delta}$ such that

$$\begin{aligned}D\sigma^{-1}[(1+\beta-L\gamma)I-J-\beta J^{-1}]\sigma S\vec{\delta}&\leqslant\vec{0},\\ D\sigma^{-1}[(1+\beta-\mu\gamma)I-J-\beta J^{-1}]\sigma S\vec{\delta}&\geqslant\vec{0},\\ \vec{\delta}&\geqslant\vec{0},\\ \vec{1}^{\top}\vec{\delta}&\geqslant 1.\end{aligned}$$

Note that for each permutation σ, it consists in an LP. But to find all the K-cycles, we need to solve $(K-1)!$ LPs. (Number of permutations divided by number of cycling permutations).

Remark 10.3.1 (Correct permutation)**.** *Figure 10.1b had first been drawn for $K\leqslant 6$ due to the increasing amount of problems to solve. Then, identifying a pattern, we conjectured one good permutation (corresponding to actual cycles) could be $(x_0,x_1,x_2,\cdots,x_{K-3},x_{K-2},x_{K-1})=(x^{(1)},x^{(3)},x^{(5)},\cdots,x^{(4)},x^{(2)},x^{(0)})$, illustrated in Figure 10.2. This enabled drawing Figure 10.1b until $K=25$. Note however, that the regions are imprecise at the end.*

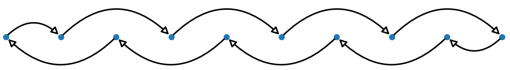


Figure 10.2: Seemingly right permutation.

Challenge. Assuming γ,β verifying the above LP, can we prove it verifies the LP provided in Theorem 8.4.13?

10.3.2 On the relationship between $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})^c$ and $\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L})$

On the one hand, we know the analytical form of $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ as

$$\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L}) = \bigcup_{K\geqslant 3} \Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) \tag{10.18}$$

with

$$\begin{aligned}\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L}) = \Big\{ & (\gamma,\beta) \in \Omega_{\text{cv}}(\mathcal{Q}_{\mu,L}) \mid \\ & (\mu\gamma)^2 - 2\left[\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)\right](\mu\gamma) \\ & + 2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K) \leqslant 0.\Big\},\end{aligned}$$

and we know that

$$\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L}) \subseteq \Omega_{\text{cv}}(\mathcal{F}_{\mu,L}) \subseteq \Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})^c \subseteq \Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})^c. \tag{10.19}$$

Figure 10.1a brings some evidence to Conjecture 8.4.15 according to which

$$\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L}) \subseteq \Omega_{\text{cv}}(\mathcal{F}_{\mu,L}) \subseteq \Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})^c = \Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})^c. \tag{10.20}$$

Moreover, Figure 10.3 tends to show that $\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L}) \neq \Omega_{\text{Cycle}}(\mathcal{F}_{\mu,L})^c = \Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})^c$. Indeed, some white regions appear between $\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L})$ and $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$.

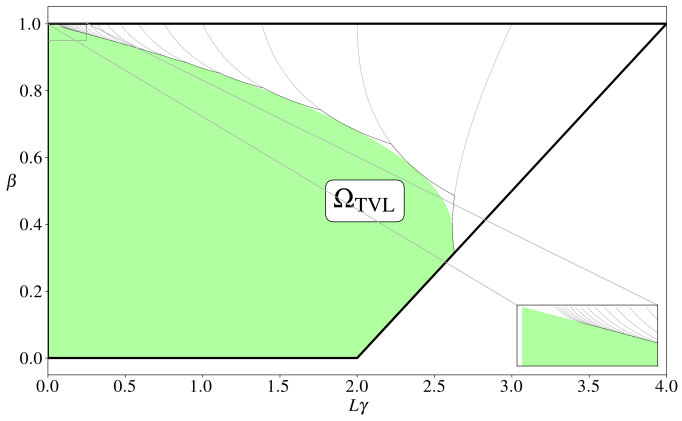


Figure 10.3: Area of Lyapunov in green. Cycles's borders in black.

Therefore, a question naturally arises:

Can we analytically characterize $\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L})$?

First, we start from the expression of $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$. According to Theorem 3.5, $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ can be defined by the inequality

$$\min_{\substack{K\geqslant 2\\ K \text{ integer}}} \left\{(\mu\gamma)^2 - 2\left[\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)\right](\mu\gamma) + 2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K)\right\} \leqslant 0. \tag{10.21}$$

Figure 10.3 not only shows that $\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L})$ and $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ do not fill $\Omega_{\text{cv}}(\mathcal{Q}_{\mu,L})$, but also that $\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L})$ frontier looks smooth while $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ do not due to

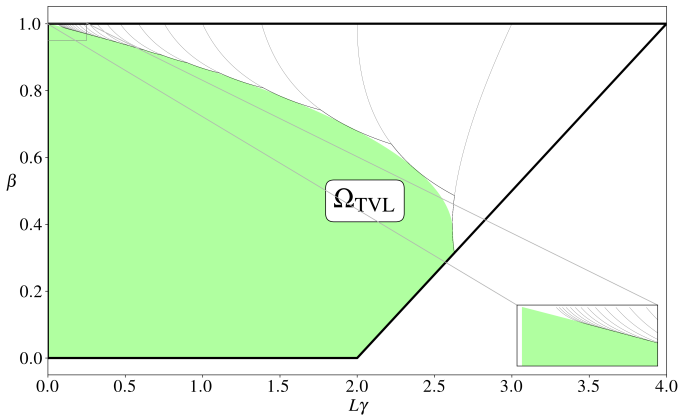


Figure 10.4: Area of Lyapunov in green. Cycles's borders of length multiples of 1/4 in black.

the countable union of smooth sets. Moreover, $\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L})$ seems tangent to each $\Omega_{K\text{-o-Cycle}}(\mathcal{F}_{\mu,L})$. A natural thought is to "smooth" $\Omega_{\text{o-Cycle}}(\mathcal{F}_{\mu,L})$ by considering non-integer K. Figure 10.4 empirically shows that the hull of all those newly created regions fits exactly $\Omega_{\text{Taylor}}(\mathcal{F}_{\mu,L})$.

We then consider the relaxed equation

$$\min_{\substack{K\geqslant 3\\ K\,real}} \left\{(\mu\gamma)^2 - 2\left[\beta - \cos\theta_K + \kappa(1-\beta\cos\theta_K)\right](\mu\gamma) + 2\kappa(1-\cos\theta_K)(1+\beta^2-2\beta\cos\theta_K)\right\} \leqslant 0. \tag{10.22}$$

We can show that the latter is equivalent to

$$\frac{1-\beta}{(1-\beta\kappa)^2}\left[(1+3\beta(1-\kappa)-\kappa\beta^2) + \sqrt{4\beta(2-\kappa-\kappa\beta)(1+\beta-2\beta\kappa)}\right] \leqslant L\gamma \leqslant \frac{1+\beta^2+4\beta}{1+\beta\kappa}. \tag{10.23}$$

This is represented on Figure 10.5

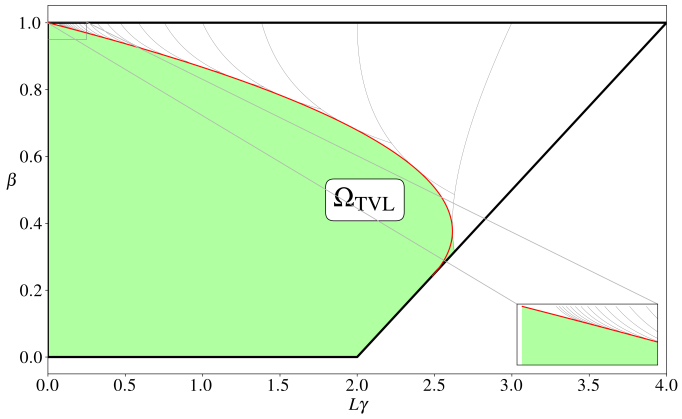


Figure 10.5: Area of Lyapunov in green. Cycle borders in black, and cycles hull border in red.

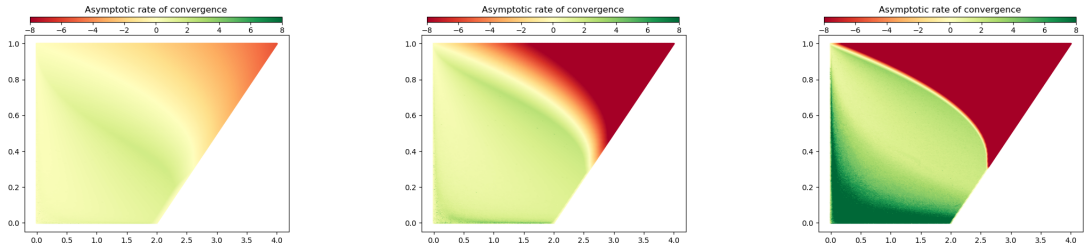
Challenge. How could we interpret those fractionary cycles?

10.3.3 Optimal convergence rate

Finally, we would like to obtain the optimal convergence rate of HB over $\mathcal{F}_{\mu,L}$. On the one hand, we know that HB is at least as fast as GD since one possible tuning of HB is $\beta = 0$. Then, HB's optimal rate is at most $\frac{1-\kappa}{1+\kappa}$. On the other hand, excluding cycle regions, Theorem 8.3.6 and Corollary 8.3.7 show that the rate of HB cannot be better than $\frac{1-50/3\kappa}{1+50/3\kappa}$. We conclude that the optimal ρ verifies

$$\frac{1-50\kappa/3}{1+50\kappa/3} \leqslant \rho^\star \leqslant \frac{1-\kappa}{1+\kappa}. \tag{10.24}$$

We parametrize ρ^* as $\frac{1-c\kappa}{1+c\kappa}$ where c is a number in $[1, 50/3]$ and look for the largest c corresponding to a convergence rate HB can reach. We numerically compute it in Figure 10.6.



(a) $\kappa = 0.05$ (b) $\kappa = 0.01$ (c) $\kappa = 0.001$

Figure 10.6: Heat map of the rate expressed with c.

The optimal rates and settings are summarized in Table 10.1.

κ	Optimal c	Optimal β	Optimal γ
0.05	1.87	0.37	1.9
0.01	2.58	0.65	1.57
0.001	3.76	0.839	1.01

Table 10.1: Optimal setting for HB on $\mathcal{F}_{\mu,L}$

The optimal c seems to increase when κ decreases. What is the limit?

Challenge. Finding the analytical expressions of the optimal tuning and rate.

10.4 An interesting class between $\mathcal{F}_{\mu,L}$ and $\mathcal{Q}_{\mu,L}$?

On the one hand, a well-known result about HB is its accelerated convergence rate over $\mathcal{Q}_{\mu,L}$. On the other hand, in Chapter 8, we prove the non-acceleration of HB on $\mathcal{F}_{\mu,L}$. A natural question therefore is "Can we find a class in between $\mathcal{Q}_{\mu,L}$ and $\mathcal{F}_{\mu,L}$ on which HB accelerates?" In simple words, "Can we *extend* the acceleration result of HB beyond the quadratics?"

As detailed in Chapter 5, a way to extend a result from a class to a larger one is to look at the proof and discard the unused constraints. However, in this case, we know a proof over $\mathcal{Q}_{\mu,L}$, based on polynomials or matrix manipulations, not combination of inequalities. The first step is therefore to find a proof involving interpolation constraints of the class $\mathcal{Q}_{\mu,L}$.

A class constraints-based proof. By chance, they have been recently discovered by Bousselmi et al. (2023). Using them, and applying the Lyapunov-search approach of (Taylor et al., 2018a), we found the following result:

Theorem 10.4.1. *Let x_t be a sequence of iterates generated by* (HB)$_{\gamma,\beta}$ *on a given function f. Set $g_t \triangleq \nabla f(x_t)$. Let V_t defined as $V_t \triangleq \|x_t\|^2 - \langle x_{t-1}, x_{t+1}\rangle$. We verify that:*

1. *If $\langle x_t, g_{t+1}\rangle \leqslant \langle x_{t+1}, g_t\rangle$ (e.g., if f is quadratic), then $V_{t+1} \leqslant \beta V_t$;*
2. *If $f \in \mathcal{F}_{\mu,L}$, then $V_t \geqslant \frac{1}{\beta}\left(\frac{\gamma}{4}\left(\frac{2(1+\beta)}{h} - \gamma\right) - \frac{(1-\beta)^2}{4h^2}\right)\|g_t\|^2$, with $h = \mu$ if $\gamma < \frac{L+\mu}{2L\mu}\frac{(1-\beta)^2}{1+\beta}$ and $h = L$ if $\gamma > \frac{L+\mu}{2L\mu}\frac{(1-\beta)^2}{1+\beta}$.*

The first result shows that V_t decreases exponentially fast, while the second one ensures V_t is lower bounded. Moreover, the lower bound is non-negative in the robust region, and only in the robust region. And it is positive in the interior of the robust region. Finally, this proves that (HB)$_{\gamma,\beta}$ verifies $\|g_t\|^2 = \mathcal{O}(\beta^t)$ in the interior of the robust region.

Proof.

1. We start by proving the first assertion. From
$$\langle x_t, \gamma g_{t+1}\rangle \leqslant \langle x_{t+1}, \gamma g_t\rangle,$$
we use the update equation of Heavy-ball and obtain
$$\langle x_t, x_{t+1} - x_{t+2} + \beta(x_{t+1} - x_t)\rangle \leqslant \langle x_{t+1}, x_t - x_{t+1} + \beta(x_t - x_{t-1})\rangle.$$
A simple reordering of the terms leads to
$$\langle x_t, x_{t+1} - x_{t+2}\rangle - \langle x_{t+1}, x_t - x_{t+1}\rangle \leqslant \beta\langle x_{t+1}, x_t - x_{t-1}\rangle - \beta\langle x_t, x_{t+1} - x_t\rangle,$$
that is $V_{t+1} \leqslant \beta V_t$.

2. We now prove the second assertion.
$$\begin{aligned}
\beta V_t =& \beta\left[\|x_t\|^2 - \langle x_{t-1}, x_{t+1}\rangle\right]\\
=& \beta\left[\|x_t\|^2 - \langle x_{t-1}, x_t + \beta(x_t - x_{t-1}) - \gamma g_t\rangle\right]\\
=& \left(\frac{2}{L-\mu}\left|\frac{\gamma}{4}(1+\beta) - \frac{L+\mu}{2L\mu}\frac{(1-\beta)^2}{4}\right| + \frac{(1-\beta)^2}{4L\mu}\right)\left[(L+\mu)\langle x_t, g_t\rangle - \|g_t\|^2 - L\mu\|x_t\|^2\right]\\
&+ \left(\frac{\gamma}{4}\left(\frac{2(1+\beta)}{h} - \gamma\right) - \frac{(1-\beta)^2}{4h^2}\right)\|g_t\|^2\\
&+ \frac{2L\mu}{L-\mu}\left|\frac{\gamma}{4}(1+\beta) - \frac{L+\mu}{2L\mu}\frac{(1-\beta)^2}{4}\right|\|x_t - \frac{1}{h}g_t\|^2\\
&+ \frac{1}{4}\|\gamma g_t - (1+\beta)x_t + 2\beta x_{t-1}\|^2\\
\geqslant& \left(\frac{\gamma}{4}\left(\frac{2(1+\beta)}{h} - \gamma\right) - \frac{(1-\beta)^2}{4h^2}\right)\|g_t\|^2,
\end{aligned}$$

with $h = \mu$ if $\gamma < \frac{L+\mu}{2L\mu}\frac{(1-\beta)^2}{1+\beta}$ and $h = L$ if $\gamma > \frac{L+\mu}{2L\mu}\frac{(1-\beta)^2}{1+\beta}$.

■

Challenge. We know the quadratic assumption is only used through the inequality $\langle x_t, g_{t+1}\rangle \leqslant \langle x_{t+1}, g_t\rangle$. Can we relax the quadratic assumption, replacing it by a strictly weaker assumption whose expression does not depend on the iterates?

10.5 An adaptive strategy for HB on $\mathcal{F}_{\mu,L}$?

While in Chapter 8 we prove the non-acceleration of HB with constant parameters on $\mathcal{F}_{\mu,L}$, the question is still open for adaptive strategies. Barré et al. (2020) already proved that HB with Polyak step-sizes converges with a linear rate $\left(1 - \frac{\mu}{L}\right)^{3/4}$, which is faster than GD but still slower than accelerating methods like ITEM (Taylor and Drori, 2022). Another type of adaptive strategy is the backtracking line-search. As explained in (Park and Ryu, 2021), the backtracking line-search strategy consists in forcing the inequalities used in the proof to hold.

Challenge. Based on the previous analysis made of HB on non-quadratic objectives, can we enforce $\langle x_t, \gamma g_{t+1}\rangle \leqslant \langle x_{t+1}, \gamma g_t\rangle$ to hold just by modifying the parameters in an online manner?

10.6 Distributed learning

Obtaining a worst-case guarantee in first-order optimization is a very challenging task. The constructive approaches that we discussed constitute essential tools towards this goal. While we focus on very classical algorithms in this thesis, the presented tools can be applied to more complex algorithms such as stochastic methods (see Taylor and Bach (2019); Hu et al. (2021)) and inexact gradient-based methods (see De Klerk et al. (2020); Gannot (2021)). In particular, recent extensions of PEP/IQCs include distributed and decentralized optimization (Sundararajan et al., 2019, 2020; Colla and Hendrickx, 2021). In this setting, we need to deal with both a local gradient update and a communication step. Moreover, when the communication bandwidth is limited (see e.g. Alistarh et al. (2017); Wu et al. (2018); Mishchenko et al. (2019); Horváth et al. (2019); Li et al. (2020); Horváth and Richtárik (2020)), or when privacy is desired (see e.g. Arora et al. (2022); Bassily et al. (2021)), the communicated states need to be corrupted by either a compression step or a high level of noise. The compression step is often very challenging to tackle in optimization as it biases the estimates of the gradients. To improve over classical SGD skim with inexact gradients, methods like *Error Feedback* have been proposed (see Seide et al. (2014); Stich and Karimireddy (2020); Karimireddy et al. (2019); Richtárik et al. (2021)).

Challenges. Comparing those methods' performance is a challenging task as, while there exist some upper bounds, there is no known tight bound. Finding such bounds is therefore challenging. The PEP approach described in Chapter 5 together with our software presented in Chapter 4 will be of great help in this endeavor. Finally, applying the SSEP techniques (see Drori and Taylor (2020)) on the dual of the PEP could help in designing a performing method while looking for a Lyapunov functional for this method.

Bibliography

H. Abbaszadehpeivasti, E. de Klerk, and M. Zamani. The exact worst-case convergence rate of the gradient method with fixed step lengths for L-smooth functions. *Optimization Letters*, 2021.

H. Abbaszadehpeivasti, E. de Klerk, and M. Zamani. Conditions for linear convergence of the gradient method for non-convex optimization. *Optimization Letters*, 17(5):1105–1125, 2023.

A. Agarwal, S. N. Negahban, and M. J. Wainwright. Fast global convergence of gradient methods for high-dimensional statistical recovery. *Annals of statistics*, 40(5):2452–2482, 2012.

N. Agarwal, S. Goel, and C. Zhang. Acceleration via fractal learning rate schedules. *arXiv preprint arXiv:2103.01338*, 2021.

D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic. QSGD: Communication-Efficient SGD via Gradient Quantization and Encoding. *Advances in Neural Information Processing Systems (NIPS)*, 30:1709–1720, 2017.

R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 32, 2019.

M. Anitescu. Degenerate nonlinear programming with a quadratic growth condition. *SIAM Journal on Optimization*, 10(4):1116–1135, 2000.

Y. Arjevani, S. Shalev-Shwartz, and O. Shamir. On lower and upper bounds in smooth and strongly convex optimization. *The Journal of Machine Learning Research*, 17(1): 4303–4353, 2016.

L. Armijo. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 1966.

R. Arora, R. Bassily, C. Guzmán, M. Menart, and E. Ullah. Differentially private generalized linear models revisited. *Advances in Neural Information Processing Systems (NeurIPS)*, 35: 22505–22517, 2022.

F. Bach. Self-concordant analysis for logistic regression. *Electronic Journal of Statistics*, 4: 384–414, 2010.

N. Bansal and A. Gupta. Potential-function proofs for gradient methods. *Theory of Computing*, 15(1):1–32, 2019.

M. Barré. *Worst-case analysis of efficient first-order methods*. PhD thesis, Université Paris sciences et lettres, 2021.

M. Barré, A. Taylor, and A. d'Aspremont. Complexity guarantees for polyak steps with momentum. In *Conference on Learning Theory*, 452–478. PMLR, 2020.

M. Barré, A. B. Taylor, and F. Bach. Principled analyses and design of first-order methods with inexact proximal operators. *Math. Programming*, 201(1):185–230, 2023.

J. Barzilai and J. M. Borwein. Two-point step size gradient methods. *IMA journal of numerical analysis*, 8(1):141–148, 1988.

R. Bassily, C. Guzmán, and M. Menart. Differentially private stochastic optimization: New results in convex and non-convex settings. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:9317–9329, 2021.

H. H. Bauschke, J. Bolte, and M. Teboulle. A descent lemma beyond Lipschitz gradient continuity: first-order methods revisited and applications. *Mathematics of Operations Research*, 42(2):330–348, 2017.

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl. Algorithms for hyper-parameter optimization. *Advances in Neural Information Processing Systems (NIPS)*, 24, 2011.

R. Berthier, F. Bach, and P. Gaillard. Accelerated gossip in networks of given dimension using Jacobi polynomial iterations. *SIAM Journal on Mathematics of Data Science*, 2(1): 24–47, 2020.

D. P. Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 1997.

J. Bolte, T. P. Nguyen, J. Peypouquet, and B. W. Suter. From error bounds to the complexity of first-order descent methods for convex functions. *Math. Programming*, 165(2):471–507, 2017.

J. F. Bonnans and A. Ioffe. Second-order sufficiency and quadratic growth for nonisolated minima. *Mathematics of Operations Research*, 20(4):801–817, 1995.

J.-F. Bonnans, J.-C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal. *Numerical optimization: theoretical and practical aspects*. Springer Science & Business Media, 2006.

K.-H. Borgwardt. *Untersuchungen zur Asymptotik der mittleren Schrittzahl von Simplexverfahren in der linearen Optimierung*. PhD thesis, Universitat Kaiserslautern, 1977.

K. H. Borgwardt. *The simplex method: a probabilistic analysis*, Number 1 in Algorithms and Combinatorics. Springer-Verlag, 1980.

L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2007.

N. Bousselmi, J. M. Hendrickx, and F. Glineur. Interpolation conditions for linear operators and applications to performance estimation problems. *arXiv:2302.08781*, 2023.

S. Boyd, L. Xiao, and A. Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2003.

H. Brézis. *Functional analysis, Sobolev spaces and partial differential equations*, 2. Springer, 2011.

S. Bubeck. Convex optimization: Algorithms and complexity. *Found. and Trends in Machine Learning*, 8(3-4):231–357, 2015.

A. Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 1847.

A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of mathematical imaging and vision*, 40:120–145, 2011.

A. Chambolle and T. Pock. An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319, 2016.

P. L. Chebyshev. *Théorie des mécanismes connus sous le nom de parallélogrammes*. Imprimerie de l'Académie impériale des sciences, 1853.

N. H. Chieu, N. T. Q. Trang, and H. A. Tuan. Quadratic growth and strong metric subregularity of the subdifferential for a class of non-prox-regular functions, 2021.

S. Colla and J. M. Hendrickx. Automated worst-case performance analysis of decentralized gradient descent. In *Proceedings of the 60th Conference on Decision and Control (CDC)*, 2021.

P. L. Combettes and J.-C. Pesquet. Proximal splitting methods in signal processing. *Fixed-point algorithms for inverse problems in science and engineering*, 185–212, 2011.

R. Couillet and F. Benaych-Georges. Kernel spectral clustering of large dimensional data. *Electronic Journal of Statistics*, 2016.

Y. Cui, C. Ding, and X. Zhao. Quadratic growth conditions for convex matrix optimization problems associated with spectral functions, 2017.

L. Cunha, G. Gidel, F. Pedregosa, D. Scieur, and C. Paquette. Only tails matter: Average-case universality and robustness in the convex regime. In *International Conference on Machine Learning (ICML)*, 2022.

S. Cyrus, B. Hu, B. Van Scoy, and L. Lessard. A robust accelerated optimization algorithm for strongly convex functions. In *2018 Annual American Control Conference (ACC)*, 1376–1381. IEEE, 2018.

S. Das Gupta, B. P. Van Parys, and E. K. Ryu. Branch-and-bound performance estimation programming: a unified methodology for constructing optimal optimization methods. *Math. Programming*, 1–73, 2023.

D. Davis and W. Yin. A three-operator splitting scheme and its optimization applications. *Set-valued and variational analysis*, 25:829–858, 2017.

E. De Klerk, F. Glineur, and A. B. Taylor. On the worst-case complexity of the gradient method with exact line search for smooth strongly convex functions. *Optimization Letters*, 11(7):1185–1199, 2017.

E. De Klerk, F. Glineur, and A. B. Taylor. Worst-case convergence analysis of inexact gradient and newton methods through semidefinite programming performance estimation. *SIAM Journal on Optimization*, 30(3):2053–2082, 2020.

A. Defazio. A simple practical accelerated method for finite sums. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *The Journal of Machine Learning Research (JMLR)*, 17(1):2909–2913, 2016.

P. Dobson, J. M. Sanz-Serna, and K. Zygalakis. On the connections between optimization algorithms, lyapunov functions, and differential equations: theory and insights. *arXiv preprint arXiv:2305.08658*, 2023.

R. D'Orazio, N. Loizou, I. Laradji, and I. Mitliagkas. Stochastic mirror descent: Convergence analysis and adaptive variants via the mirror stochastic Polyak stepsize. *Transactions on Machine Learning Research (TMLR)*, 2021.

R.-A. Dragomir. *Bregman Gradient Methods for Relatively-Smooth Optimization*. PhD thesis, UT1 Capitole, 2021.

R.-A. Dragomir and Y. Nesterov. Convex quartic problems: homogenized gradient method and preconditioning. *arXiv preprint arXiv:2306.17683*, 2023.

R.-A. Dragomir, A. B. Taylor, A. d'Aspremont, and J. Bolte. Optimal complexity and certification of bregman first-order methods. *Math. Programming*, 1–43, 2021.

Y. Drori. *Contributions to the Complexity Analysis of Optimization Algorithms*. PhD thesis, Tel-Aviv University, 2014.

Y. Drori. The exact information-based complexity of smooth convex minimization. *Journal of Complexity*, 39:1–16, 2017.

Y. Drori and A. Taylor. On the oracle complexity of smooth strongly convex minimization. *Journal of Complexity*, 68:101590, 2022.

Y. Drori and A. B. Taylor. Efficient first-order methods for convex minimization: a constructive approach. *Math. Programming*, 184(1):183–220, 2020.

Y. Drori and M. Teboulle. Performance of first-order methods for smooth convex minimization: a novel approach. *Math. Programming*, 145(1):451–482, 2014.

Y. Drori and M. Teboulle. An optimal variant of kelley's cutting-plane method. *Math. Programming*, 160(1-2):321–351, 2016.

D. Drusvyatskiy and A. D. Ioffe. Quadratic growth and critical point stability of semi-algebraic functions. *Math. Programming*, 153(2):635–653, 2015.

D. Drusvyatskiy and A. S. Lewis. Error bounds, quadratic growth, and linear convergence of proximal methods. *Mathematics of Operations Research*, 43(3):919–948, 2018.

P. Dvurechensky, S. Shtern, and M. Staudigl. First-order methods for convex optimization. *EURO Journal on Computational Optimization*, 9, 2021.

A. d'Aspremont, D. Scieur, and A. Taylor. Acceleration methods. *Foundations and Trends® in Optimization*, 5(1-2):1–245, 2021.

M. Fazel, H. Hindi, and S. P. Boyd. Log-det heuristic for matrix rank minimization with applications to hankel and euclidean distance matrices. In *American Control Conference (ACC).*, 3, 2156–2162. IEEE, 2003.

M. Fazlyab, A. Ribeiro, M. Morari, and V. M. Preciado. Analysis of optimization algorithms via integral quadratic constraints: Nonstrongly convex problems. *SIAM Journal on Optimization*, 28(3):2654–2689, 2018.

D. Ferbach, B. Goujaud, G. Gidel, and A. Dieuleveut. Proving linear mode connectivity of neural networks via optimal transport. *arXiv preprint arXiv:2310.19103*, 2023.

B. Fischer. *Polynomial based iteration methods for symmetric linear systems*. SIAM, 2011.

D. A. Flanders and G. Shortley. Numerical determination of fundamental modes. *Journal of Applied Physics*, 21(12):1326–1332, 1950.

O. Gannot. A frequency-domain analysis of inexact gradient methods. *Math. Programming*, 1–42, 2021.

E. Ghadimi, H. R. Feyzmahdavian, and M. Johansson. Global convergence of the heavy-ball method for convex optimization. In *European control conference (ECC)*, 310–315. IEEE, 2015.

B. Ghorbani, S. Krishnan, and Y. Xiao. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning (ICML)*, 2019.

G. Gidel, H. Berard, G. Vignoud, P. Vincent, and S. Lacoste-Julien. A variational inequality perspective on generative adversarial networks. In *International Conference on Learning Representations*, 2018.

G. Gidel, R. A. Hemmat, M. Pezeshki, R. Le Priol, G. Huang, S. Lacoste-Julien, and I. Mitliagkas. Negative momentum for improved game dynamics. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 1802–1811. PMLR, 2019.

G. Goh. Why Momentum Really Works, 2017.

G. H. Golub and R. S. Varga. Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods. *Numerische Mathematik*, 3(1):157–168, 1961.

P. Gong and J. Ye. Linear convergence of variance-reduced stochastic gradient without strong convexity. *arXiv:1406.1102*, 2014.

E. Gorbunov, N. Loizou, and G. Gidel. Extragradient method: $O(1/k)$ last-iterate convergence for monotone variational inequalities and connections with cocoercivity. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 366–402, 2022.

B. Goujaud and F. Pedregosa. Cyclical step-sizes. http://fa.bianp.net/blog/2022/cyclical/, 2022.

B. Goujaud, E. W. Tramel, P. Courtiol, M. Zaslavskiy, and G. Wainrib. Robust detection of covariate-treatment interactions in clinical trials. *arXiv preprint arXiv:1712.08211*, 2017.

B. Goujaud, C. Moucer, F. Glineur, J. Hendrickx, A. Taylor, and A. Dieuleveut. PEPit: computer-assisted worst-case analyses of first-order optimization methods in Python. *arXiv preprint arXiv:2201.04040*, 2022a.

B. Goujaud, D. Scieur, A. Dieuleveut, A. B. Taylor, and F. Pedregosa. Super-acceleration with cyclical step-sizes. In *International Conference on Artificial Intelligence and Statistics*, 3028–3065. PMLR, 2022b.

B. Goujaud, A. Taylor, and A. Dieuleveut. Optimal first-order methods for convex functions with a quadratic upper bound. *arXiv preprint arXiv:2205.15033*, 2022c.

B. Goujaud, A. Taylor, and A. Dieuleveut. Quadratic minimization: from conjugate gradient to an adaptive heavy-ball method with Polyak step-sizes. *arXiv preprint arXiv:2210.06367*, 2022d.

B. Goujaud, A. Dieuleveut, and A. Taylor. Counter-examples in first-order optimization: a constructive approach. *IEEE Control Systems Letters*, 2023a. (See arXiv 2303 10503 for complete version with appendices).

B. Goujaud, A. Dieuleveut, and A. Taylor. On fundamental proof structures in first-order optimization. *arXiv preprint arXiv:2310.02015*, 2023b.

B. Goujaud, A. Taylor, and A. Dieuleveut. Provable non-accelerations of the heavy-ball method. *arXiv preprint arXiv:2307.11291*, 2023c.

R. M. Gower, M. Blondel, N. Gazagnadou, and F. Pedregosa. Cutting some slack for SGD with Adaptive Polyak Stepsizes. *arXiv preprint arXiv:2202.12328*, 2022.

D. Granziol, X. Wan, S. Albanie, and S. Roberts. Explaining the Adaptive Generalisation Gap. *arXiv preprint arXiv:2011.08181*, 2020.

R. M. Gray. Toeplitz and circulant matrices: A review. *Foundations and Trends® in Communications and Information Theory*, 2(3):155–239, 2006. Original publication 1971.

G. Gu and J. Yang. Tight sublinear convergence rate of the proximal point algorithm for maximal monotone inclusion problems. *SIAM Journal on Optimization*, 30(3):1905–1921, 2020.

C. Guille-Escuret, B. Goujaud, M. Girotti, and I. Mitliagkas. A study of condition numbers for first-order optimization. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 1261–1269, 2021.

C. Guille-Escuret, B. Goujaud, A. Ibrahim, and I. Mitliagkas. Gradient descent is optimal under lower restricted secant inequality and upper error bound. 35, 24893–24904, 2022.

C. Gupta, S. Balakrishnan, and A. Ramdas. Path length bounds for gradient descent and flow. *The Journal of Machine Learning Research*, 22(1):3154–3216, 2021.

S. D. Gupta, R. M. Freund, X. A. Sun, and A. Taylor. Nonlinear conjugate gradient methods: worst-case convergence rates via computer-assisted analyses. *arXiv preprint arXiv:2301.01530*, 2023.

M. Hagedorn and F. Jarre. Iteration complexity of fixed-step methods by Nesterov and Polyak for convex quadratic functions. *Journal of Optimization Theory and Applications*, 1–19, 2023.

W. W. Hager and H. Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

F. Hanzely, P. Richtarik, and L. Xiao. Accelerated bregman proximal gradient methods for relatively smooth convex optimization. *Computational Optimization and Applications*, 79 (2):405–440, 2021.

M. Hardt, T. Ma, and B. Recht. Gradient descent learns linear dynamical systems. *Journal of Machine Learning Research*, 19, 2018.

E. Hazan and S. Kakade. Revisiting the Polyak step size. *arXiv preprint arXiv:1905.00313*, 2019.

E. Hazan, K. Levy, and S. Shalev-Shwartz. Beyond convexity: Stochastic quasi-convex optimization. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NIPS)*, 1594–1602. Curran Associates, Inc., 2015.

S. Horváth and P. Richtárik. A Better Alternative to Error Feedback for Communication-Efficient Distributed Learning. *International Conference on Learning Representations (ICLR))*, June 2020. arXiv: 2006.11077.

S. Horváth, D. Kovalev, K. Mishchenko, S. Stich, and P. Richtárik. Stochastic Distributed Learning with Gradient Quantization and Variance Reduction. *Optimization Methods and Software*, Apr. 2019. arXiv: 1904.05115.

B. Hu, S. Wright, and L. Lessard. Dissipativity theory for accelerating stochastic variance reduction: A unified analysis of svrg and katyusha using semidefinite programs. In *International Conference on Machine Learning (ICML)*, 2018.

B. Hu, P. Seiler, and L. Lessard. Analysis of biased stochastic gradient descent using sequential semidefinite programs. *Math. Programming*, 187:383–408, 2021.

A. Ioffe. On sensitivity analysis of nonlinear programs in banach spaces: the approach via composite unconstrained optimization. *SIAM Journal on Optimization*, 4(1):1–43, 1994.

A. Iouditski and Y. Nesterov. Primal-dual subgradient methods for minimizing uniformly convex functions. *arXiv preprint arXiv:1401.1792*, 2014.

U. Jang, S. D. Gupta, and E. K. Ryu. Computer-assisted design of accelerated composite optimization methods: Optista. *arXiv preprint arXiv:2305.15704*, 2023.

I. M. Johnstone. On the distribution of the largest eigenvalue in principal components analysis. *Annals of statistics*, 2001.

R. E. Kalman and J. E. Bertram. Control System Analysis and Design Via the "Second Method" of Lyapunov: I—Continuous-Time Systems. *Journal of Basic Engineering*, 82(2): 371–393, 06 1960a.

R. E. Kalman and J. E. Bertram. Control System Analysis and Design Via the "Second Method" of Lyapunov: II—Discrete-time systems. *Journal of Basic Engineering*, 82(2): 394–400, 06 1960b.

S. P. Karimireddy, Q. Rebjock, S. Stich, and M. Jaggi. Error feedback fixes signsgd and other gradient compression schemes. In *International Conference on Machine Learning (ICML)*, 3252–3261. PMLR, 2019.

D. Kim. Accelerated proximal point method for maximally monotone operators. *Math. Programming*, 1–31, 2021.

D. Kim and J. A. Fessler. Optimized first-order methods for smooth convex minimization. *Math. programming*, 159(1):81–107, 2016.

D. Kim and J. A. Fessler. Optimizing the efficiency of first-order methods for decreasing the gradient of smooth convex functions. *Journal of Optimization Theory and Applications*, 188(1):192–219, 2021.

J. L. Kim, G. Gidel, A. Kyrillidis, and F. Pedregosa. Extragradient with positive momentum is optimal for games with cross-shaped jacobian spectrum. *arXiv preprint arXiv:2211.04659*, 2022.

G. M. Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 12:747–756, 1976.

M. Krenn, L. Buffoni, B. Coutinho, S. Eppel, J. G. Foster, A. Gritsevskiy, H. Lee, Y. Lu, J. P. Moutinho, N. Sanjabi, et al. Predicting the future of ai with ai: High-quality link prediction in an exponentially growing knowledge network. *arXiv preprint arXiv:2210.00881*, 2022.

K. Kurdyka. On gradients of functions definable in o-minimal structures. *Annales de l'institut Fourier*, 48:769–783, 1998.

D. Lambert, J.-P. Crouzeix, V. H. Nguyen, and J.-J. Strodiot. Finite convex integration. *Journal of Convex Analysis*, 11(1):131–146, 2004.

C. Lanczos. Solution of systems of linear equations by. *Journal of research of the National Bureau of Standards*, 49(1):33, 1952.

Y. Le Cun, C. Cortes, and C. Burges. MNIST handwritten digit database. *ATT Labs [Online]*, 2010.

L. Lessard. The analysis of optimization algorithms: A dissipativity approach. *IEEE Control Systems Magazine*, 42(3):58–72, 2022.

L. Lessard, B. Recht, and A. Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

Z. Li, D. Kovalev, X. Qian, and P. Richtarik. Acceleration for Compressed Gradient Descent in Distributed and Federated Optimization. In *International Conference on Machine Learning*, 5895–5904. PMLR, Nov. 2020. ISSN: 2640-3498.

F. Lieder. On the convergence rate of the Halpern-iteration. *Optimization Letters*, 15(2): 405–418, 2021.

J. Liu and S. J. Wright. Asynchronous stochastic coordinate descent: Parallelism and convergence properties. *SIAM Journal on Optimization*, 25(1):351—376, 2015.

N. Loizou, S. Vaswani, I. H. Laradji, and S. Lacoste-Julien. Stochastic Polyak step-size for SGD: An adaptive learning rate for fast convergence. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 1306–1314. PMLR, 2021.

I. Loshchilov and F. Hutter. SGDR: stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

H. Lu, R. M. Freund, and Y. Nesterov. Relatively-smooth convex optimization by first-order methods, and applications. *SIAM Journal on Optimization*, 28(1):333–354, 2018.

Z.-Q. Luo and P. Tseng. Error bounds and convergence analysis of feasible descent methods: a general approach. *Annals of Operations Research*, 46(1):157–178, 1993.

A. M. Lyapunov and A. Fuller. The general problem of the stability of motion. *International journal of control*, 55(3):531–534, 1992. Original text in Russian, 1892.

Y. Malitsky and K. Mishchenko. Adaptive gradient descent without descent. In *International Conference on Machine Learning (ICML)*, 6702–6712. PMLR, 2020.

U. Marteau-Ferey, D. Ostrovskii, F. Bach, and A. Rudi. Beyond least-squares: Fast rates for regularized empirical risk minimization through self-concordance. In *Conference on learning theory*, 2294–2340. PMLR, 2019.

K. Mishchenko, E. Gorbunov, M. Takáč, and P. Richtárik. Distributed Learning with Compressed Gradient Differences. *arXiv:1901.09269 [cs, math, stat]*, June 2019. arXiv: 1901.09269.

R. D. Monteiro and B. F. Svaiter. An accelerated hybrid proximal extragradient method for convex optimization and its implications to second-order methods. *SIAM Journal on Optimization*, 23(2):1092–1125, 2013.

A. MOSEK. *MOSEK Optimizer API for C 9.3.6*, 2019.

G. Narkiss and M. Zibulevsky. *Sequential subspace optimization method for large-scale unconstrained problems*. Technion-IIT, Department of Electrical Engineering, 2005.

I. Necoara, Y. Nesterov, and F. Glineur. Linear convergence of first order methods for non-strongly convex optimization. *Math. Programming*, 175(1):69–107, 2019.

A. S. Nemirovskii. Orth-method for smooth convex optimization. *Engineering Cybernetics*, 20(2):937–947, 1982.

A. S. Nemirovskii. Information-based complexity of linear operator equations. *Journal of Complexity*, 8(2):153–175, 1992.

A. S. Nemirovskii. Information-based complexity of convex programming. *Lecture notes,* http://www2.isye.gatech.edu/~nemirovs/Lec_EMCO.pdf, 1994.

A. S. Nemirovskii and Y. Nesterov. Optimal methods of smooth convex minimization. *USSR Computational Mathematics and Mathematical Physics*, 25(2):21–30, 1985.

A. S. Nemirovskii and D. B. Yudin. Problem complexity and method efficiency in optimization. *Willey-Interscience, New York*, 1983a.

A. S. Nemirovskii and D. B. Yudin. Information-based complexity of mathematical programming. *Izvestia AN SSSR, Ser. Tekhnicheskaya Kibernetika (the journal is translated to English as Engineering Cybernetics. Soviet J. Computer & Systems Sci.)*, 1, 1983b.

Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

Y. Nesterov. *Introductory Lectures on Convex Optimization*. Springer, 2003.

Y. Nesterov. Gradient methods for minimizing composite functions. *Math. programming*, 140(1):125–161, 2013.

Y. Nesterov. Universal gradient methods for convex optimization problems. *Math. Programming*, 152(1-2):381–404, 2015.

J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 1999.

S. Oymak. Super-convergence with an unstable learning rate. *arXiv preprint arXiv:2102.10734*, 2021.

B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.

V. Papyan. The full spectrum of deepnet hessians at scale: Dynamics with SGD training and sample size. *arXiv preprint arXiv:1811.07062*, 2018.

V. Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. In *International Conference on Machine Learning (ICML)*, 2019.

C. Park and E. K. Ryu. Optimal first-order algorithms as a function of inequalities. *arXiv:2110.11035*, 2021.

J. Park and E. K. Ryu. Exact optimal accelerated complexity for fixed-point iterations. In *International Conference on Machine Learning (ICML)*, 17420–17457. PMLR, 2022.

P. Patrinos, L. Stella, and A. Bemporad. Douglas-Rachford splitting: Complexity estimates and accelerated variants. In *Proceedings of the 53rd Conference on Decision and Control (CDC)*, 2014.

F. Pedregosa. On the Link Between Optimization and Polynomials, Part 1, 2020.

F. Pedregosa. On the Link Between Optimization and Polynomials, Part 3, 2021a.

F. Pedregosa. A hitchhiker's guide to momentum. http://fa.bianp.net/blog/2021/hitchhiker/, 2021b.

F. Pedregosa and D. Scieur. Acceleration through spectral density estimation. In *International Conference on Machine Learning (ICML)*, 2020.

W. Peng, H. Zhang, X. Zhang, and L. Cheng. Global complexity analysis of inexact successive quadratic approximation methods for regularized optimization under mild assumptions. *Journal of Global Optimization*, 78(1):69–89, 2020.

J. Pennington and P. Worah. Nonlinear random matrix theory for deep learning. In *Advances on Neural Information Processing Systems (NIPS)*, 2017.

B. T. Polyak. Gradient methods for the minimisation of functionals. *USSR Computational Mathematics and Mathematical Physics*, 3(4):864 – 878, 1963.

B. T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR computational mathematics and mathematical physics*, 4(5):1–17, 1964.

B. T. Polyak. *Introduction to optimization*. Optimization Software New York, 1987.

B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

P. Richtárik, I. Sokolov, and I. Fatkhullin. Ef21: A new, simpler, theoretically better, and practically faster error feedback. *Advances in Neural Information Processing Systems (NeurIPS)*, 34:4384–4396, 2021.

H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, 400–407, 1951.

R. T. Rockafellar. Monotone operators and the proximal point algorithm. *SIAM journal on control and optimization*, 14(5):877–898, 1976.

R. T. Rockafellar. *Convex analysis*, 11. Princeton university press, 1997.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

D. Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

H. Rutishauser. Theory of gradient methods. In *Refined iterative methods for computation of the solution and the eigenvalues of self-adjoint boundary value problems*. Springer, 1959.

E. K. Ryu, A. B. Taylor, C. Bergeling, and P. Giselsson. Operator splitting performance estimation: Tight contraction factors and optimal parameter selection. *SIAM Journal on Optimization*, 30(3):2251–2271, 2020.

L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou. Empirical analysis of the Hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

D. Scieur. *Acceleration in optimization*. PhD thesis, Université Paris sciences et lettres, 2018.

D. Scieur and F. Pedregosa. Universal Asymptotic Optimality of Polyak Momentum. In *International Conference on Machine Learning (ICML)*, 2020.

F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth annual conference of the international speech communication association*, 2014.

M. Slater. Lagrange multipliers revisited: a contribution to nonlinear programming, 1950.

L. N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2017.

S. U. Stich and S. P. Karimireddy. The error-feedback framework: Better rates for sgd with delayed gradients and compressed updates. *The Journal of Machine Learning Research*, 21(1):9613–9648, 2020.

A. Sundararajan, B. Van Scoy, and L. Lessard. A canonical form for first-order distributed optimization algorithms. In *2019 American Control Conference (ACC)*, 4075–4080. IEEE, 2019.

A. Sundararajan, B. Van Scoy, and L. Lessard. Analysis and design of first-order distributed optimization algorithms over time-varying graphs. *IEEE Transactions on Control of Network Systems*, 7(4):1597–1608, 2020.

I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning (ICML)*, 2013.

A. Taylor and F. Bach. Stochastic first-order methods: non-asymptotic and computer-aided analyses via potential functions. In *Proceedings of the 32nd Conference on Learning Theory (COLT)*, 2019.

A. Taylor and Y. Drori. An optimal gradient method for smooth strongly convex minimization. *Math. Programming*, 199(1-2):557–594, 2022.

A. Taylor, B. Van Scoy, and L. Lessard. Lyapunov functions for first-order methods: Tight automated convergence guarantees. In *International Conference on Machine Learning (ICML)*, 2018a.

A. B. Taylor. Computer-aided analyses in optimization, 2020.

A. B. Taylor, J. M. Hendrickx, and F. Glineur. Exact worst-case performance of first-order methods for composite convex optimization. *SIAM Journal on Optimization*, 27(3): 1283–1313, 2017a.

A. B. Taylor, J. M. Hendrickx, and F. Glineur. Performance estimation toolbox (PESTO): automated worst-case analysis of first-order optimization methods. In *56th Annual Conference on Decision and Control (CDC)*, 1278–1283, 2017b.

A. B. Taylor, J. M. Hendrickx, and F. Glineur. Smooth strongly convex interpolation and exact worst-case performance of first-order methods. *Math. Programming*, 161(1-2): 307–345, 2017c.

A. B. Taylor, J. M. Hendrickx, and F. Glineur. Exact worst-case convergence rates of the proximal gradient method for composite convex minimization. *Journal of Optimization Theory and Applications*, 178:455–476, 2018b.

M. Upadhyaya, S. Banert, A. B. Taylor, and P. Giselsson. Automated tight lyapunov analysis for first-order methods. *arXiv preprint arXiv:2302.06713*, 2023.

B. Van Scoy, R. A. Freeman, and K. M. Lynch. The fastest known globally convergent first-order method for minimizing strongly convex functions. *IEEE Control Systems Letters*, 2(1):49–54, 2017.

L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996.

J.-K. Wang, C.-H. Lin, A. Wibisono, and B. Hu. Provable acceleration of heavy ball beyond quadratics for a class of Oolyak-Lojasiewicz functions when the non-convexity is averaged-out. In *International Conference on Machine Learning (ICML)*, 2022.

J. Wu, W. Huang, J. Huang, and T. Zhang. Error Compensated Quantized SGD and its Applications to Large-scale Distributed Optimization. In *International Conference on Machine Learning*, 5325–5333. PMLR, July 2018. ISSN: 2640-3498.

D. Young. On richardson's method for solving linear systems with positive definite matrices. *Journal of Mathematics and Physics*, 32(1-4):243–255, 1953.

H. Zhang. Linear convergence of the proximal incremental aggregated gradient method under quadratic growth condition, 2017a.

H. Zhang. The restricted strong convexity revisited: analysis of equivalence to error bound and quadratic growth. *Optimization Letters*, 11(4):817–833, 2017b.

H. Zhang and W. Yin. Gradient methods for convex minimization: better rates under weaker conditions. Cam report, UCLA, 2013.

ECOLE
DOCTORALE
DE MATHEMATIQUES
HADAMARD

Titre : A propos des approches constructives de la théorie des algorithmes d'optimisation du premier ordre

Mots clés : Optimisation, gradient, performance

Résumé : À l'heure actuelle, caractérisée par une croissance sans précédent des données disponibles et des capacités computationnelles, le domaine de l'apprentissage automatique, et plus particulièrement de l'apprentissage profond, a connu une évolution exceptionnelle. Les algorithmes d'apprentissage automatique reposent largement sur des techniques d'optimisation pour ajuster leurs paramètres et améliorer leurs prédictions. Parmi les différentes approches d'optimisation, les méthodes du premier ordre ont émergé comme des fondements incontournables, démontrant un équilibre notable entre rapidité et précision. Il est aujourd'hui crucial de développer une théorie solide de l'optimisation du premier ordre pour en exploiter pleinement le potentiel. Ces fondements théoriques approfondissent notre compréhension des algorithmes d'optimisation actuels et ouvrent la voie à la création d'algorithmes innovants. L'efficacité démontrée du concept de momentum dans l'accélération significative de la convergence de problèmes réels témoigne de l'importance de la théorie de l'optimisation. Cette théorie a permis la formulation du momentum, transformant des intuitions théoriques en un outil d'optimisation pratique, largement adopté.
Cette thèse vise à poursuivre et accélérer les efforts visant à développer une base théorique solide de l'optimisation du premier ordre. Nous avons présenté plusieurs résultats en exploitant les structures générales des certificats de preuves. (i) Le lien entre l'optimisation quadratique et la théorie des polynômes a été utilisé pour expliquer des phénomènes observés empiriquement. (ii) Un package Python a été mis en place pour faciliter l'utilisation du framework d'*estimation de performance*. (iii) Un tutoriel détaillé expliquant la dérivation de preuves naturelles en optimisation basée sur ce cadre a été rédigé. (iv) En utilisant notre package Python, nous avons appliqué cette méthodologie pour dériver une théorie complète de l'optimisation du premier ordre sur une vaste classe de fonctions. (v) Le framework théorique d'*estimation de performance* a été étendu pour réfuter la convergence d'une famille spécifique de méthodes, démontrant finalement la non-accélération de la célèbre méthode "Heavy-ball" sur la classe des fonctions lisses et fortement convexes.

Title : On constructive approaches to the theory of first-order optimization methods.

Keywords : Optimization, gradient, performance

Abstract : In the current era marked by an unprecedented surge in available data and computational prowess, the field of machine learning, and more specifically deep learning, has witnessed an extraordinary evolution. Machine learning algorithms heavily rely on optimization techniques to tune their parameters and enhance predictive accuracy. Among the myriad of optimization approaches, the first-order optimization methods have emerged as cornerstones, demonstrating a remarkable balance between efficacy and computational efficiency. Crucially, the development of strong optimization theory is pivotal in unraveling the full potential of first-order optimization. Theoretical underpinnings not only deepen our understanding of optimization landscapes but also pave the way for the design of novel algorithms. The momentum-based algorithms have proven their effectiveness by significantly accelerating training procedures. The conceptual foundation provided by optimization theory has enabled the formulation of momentum, turning theoretical insights into a powerful and widely adopted practical optimization tool.
The role of this thesis is to pursue and accelerate the effort to develop a strong theoretical foundation of first-order optimization. We proved various results, exploiting the general structures of the certificate proofs. (i) We used the link between quadratic optimization and polynomial theory to explain empirically observed phenomena. (ii) We implemented a Python package to support the *Performance estimation* framework. (iii) We wrote a tutorial to explain how to derive natural proofs in optimization based on this framework. (iv) We applied this methodology, with the help of our Python package, to derive a complete first-order optimization theory on a very large class of functions. (v) We complemented the theoretical *Performance estimation* framework to disprove the convergence of a specific family of methods and applied it to the famous Heavy-ball method to provably disprove an acceleration over the class of smooth and strongly convex functions.

Institut Polytechnique de Paris
91120 Palaiseau, France