Distributed Optimization and Control using Operator Splitting Methods

THÈSE N^{O} 8623 (2018)

PRÉSENTÉE LE 22 JUIN 2018

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE D'AUTOMATIQUE 3

PROGRAMME DOCTORAL EN GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Georgios STATHOPOULOS

acceptée sur proposition du jury:

Prof. G. Ferrari Trecate, président du jury
Prof. C. N. Jones, directeur de thèse
Prof. P. Patrinos, rapporteur
Prof. J. Lygeros, rapporteur
Prof. M. Jaggi, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2018

Acknowledgements

My greatest thanks goes to my advisor, Colin Jones, for mainly three reasons. First, for the freedom he has given to me in terms of choosing my research directions. Second, for being available for long discussions in front of a whiteboard and for actually engaging in the problem-solving process. Colin has the ability to not only propose an answer to a given question, but also to put together a start-to-end approach involving the questions that will be raised in the process and their potential answers. I was usually left with filling in the missing pieces, which were in most cases, and surprisingly, 'fillable'. Third, for all the provisions that I enjoyed during my PhD, be it traveling to any conference I would feel like attending, or pursuing side-projects that would interest me. His research style has, unarguably, been a decisive influence on mine, and I have, hopefully, adopted a tiny bit of his practical 'let's do something useful with it' attitude.

I would like to thank my committee members, Prof. Panos Patrinos, Prof. John Lygeros, Prof. Martin Jaggi and Prof. Giancarlo Ferrari Trecate for undertaking the laborious task of reading the dissertation, and for providing me with useful feedback as well as insightful suggestions for continuation of my work. I would also like to thank them for making the exam a pleasant and interesting experience.

I am indebted to Dr. Claus Danielson and Dr. Daniel Burns for hosting me at MERL during the summer of 2016. I really enjoyed the experience of working in such a lax and, at the same time, intriguing environment, which contributed to making my stay in Boston memorable and fun.

Luck plays a significant role in every life experience, and I consider myself really lucky to be in such a great working environment, the LA. The lab is a colorful mosaic of characters, all very different and exceptionally interesting. Everybody contributed in creating a warm and friendly atmosphere, fostering both fruitful collaborations and a great deal of extracurricular activities. More specifically, many thanks go to the first generation that warmly welcomed me to LA, Milan, Jean-Hubert, Andrea, and Ye, as well as the rest of the gang that joined roughly at the same time as I did, namely Tomasz, Faran, Altug; it was great having you all around. In addition, a big thanks goes to the whole LA; thanks Sanket, Francisco, Martand, Tafarel, Predrag, René, Luca, Kristoph, Harsh, Ehsan, Yanni, Truong, Ivan, Diogo, Michele, Peter, Zlatko, Sean, Niketh, Basile, Mahdieh, Melanie, Timm and the newcomers Pulkit and Mustafa. I am happy that I had the chance to share with each and every one of you views on both academic and non-academic topics. I am also thankful to the professors and the non-perishable members of LA for creating and maintaining such a family-like atmosphere through several initiatives; Dominique, Roland, Ali, Christophe, Philippe, and Sandra. A special thanks to the people who made my life easier on several occasions; Francis, Norbert, Christophe, Francine, Ruth, Eva, Margot, and Nicole.

Life outside the lab wouldn't be fun without friends to share it with; thank you Taso, Sofia,

Emre, Iris, Lorenzo, Vasili (a.k.a. Skylos) and the whole greek gang that's still lingering in Switzerland. Thanks to the EPFL migrants with whom we coincided in Boston and had the chance to revive the Swiss wine-tasting atmosphere in several occasions: Elda, Guillaume, and Xiaowen. A huge thanks goes to my lifelong friends Dimitris, Kostas, Yannis, and Giorgos for patiently listening to my uninteresting academic-life stories, for sharing a room during the summer vacations, and for regulating me whenever it was getting too much or too little.

It would be no exaggeration to say that my decision to come to Lausanne and pursue a PhD here was thanks to Dorina and Vicky, the closest thing to a 'family' that I have in Switzerland. Thank you for hosting me during the first period and for constantly being there providing both material and immaterial help. Special thanks to Vaggelis, whose arrival extended the family by two more members, and who reminded me of my adolescence on numerous occasions. My deepest thanks go to Dorina, for always caring to understand me, to find out what it is that I really want and make it a common goal, and for being supportive beyond what is expected, many times at a personal cost. She has also developed the ability to listen to my ongoing monologues on topics that she could not care less about, sometimes giving me the impression that what I say is interesting.

Last but not least, I owe my most profound gratitude to my parents and brother for their unconditional love and support. They have always made it possible for me to pursue my academic goals, at, literally, any cost. More importantly, the values and principles they have taught me have shaped who I am; and I am thankful for that.

Abstract

The significant progress that has been made in recent years both in hardware implementations and in numerical computing has rendered real-time optimization-based control a viable option when it comes to advanced industrial applications. At the same time, the field of big data has emerged, seeking solutions to problems that classical optimization algorithms are incapable of providing. Though for different reasons, both application areas triggered interest in revisiting the family of optimization algorithms commonly known as *decomposition schemes* or *operator splitting methods*. This lately revived interest in these methods can be mainly attributed to two characteristics: Computationally low per-iteration cost along with small memory footprint when it comes to embedded applications, and their capacity to deal with problems of vast scales via decomposition when it comes to machine learning-related applications.

In this thesis, we design decomposition methods that tackle both small-scale centralized control problems and larger-scale multi-agent distributed control problems. In addition to the classical objective of devising faster methods, we also delve into less usual aspects of operator splitting schemes, which are nonetheless critical for control. In the centralized case, we propose an algorithm that uses decomposition in order to exactly solve a classical optimal control problem that could otherwise be solved only approximately. In the multi-agent framework, we propose two algorithms, one that achieves faster convergence and a second that reduces communication requirements.

In the first part, we look into the infinite-horizon *constrained linear quadratic regulator* (CLQR) problem. The solution to the classical unconstrained version of the problem (LQR) is known to exist in closed form, and to come with stability guarantees for the closed-loop system. The addition of constraints to the LQR formulation renders the problem unsolvable in closed form and challenging due to the infinite number of optimization variables. We propose to use decomposition in order to take advantage of the closed-form solution of the LQR, consequently splitting the CLQR into two subproblems, an infinite-dimensional least squares problem, and a simple projection operation of an infinite sequence to the non-positive orthant, both of which are solved in closed form by means of the accelerated *projected gradient method* (ProjGM). The solution enjoys all the theoretical benefits of using an infinite-horizon cost. Numerical simulations demonstrate the good practical performance of the scheme when compared to regular MPC.

The remaining two parts of the thesis deal with distributed optimization algorithms applied in settings where heterogeneous agents collaborate under the presence of a global coordinator to achieve some objective. Our motivation for studying this setting stems from smart grid applications, where energy resources cooperate in order to deliver some ancillary service to the network operator based on some contract. Distributed optimization research has been primarily motivated by machine learning applications involving

large datasets and parallel multicore architectures. The nature of our problems of interest, however, dictates the adoption of a different angle in the analysis of distributed optimization algorithms. More specifically, the problems' scale we are interested in is generally medium to large, but not huge, and there is a focus on real-time rather than offline execution. Instead of processors who perform read/write operations on a shared memory we have (physical) agents governed by dynamics who often solve local optimization problems, the details of which should remain private, and who might be running on limited resources. Several of these characteristics pose new challenges and ask for novel solutions.

We initially focus on speeding-up the solution process in such multi-agent settings. This is achieved by enabling more agents to update in a unit of time. We assume that the agents' local optimization problems differ from one another in terms of their solve time. This characteristic motivates asynchronous optimization since it is more beneficial that the agents update in arbitrary sequences and with arbitrary frequencies instead of synchronizing their updates based on the slowest of them. We propose an asynchronous version of the forward-backward iteration that does not assume any probability distribution for the activation of the agents. We prove linear convergence of an accelerated version of the method under some structural assumptions on the involved functions. Thanks to the general nature of the forward-backward iteration, the scheme results in several novel asynchronous and accelerated versions of commonly used algorithms. We demonstrate the superiority of the scheme in comparison to other asynchronous algorithms in a simulated setting, where we employ our proposed version of the *proximal gradient method* (PGM) to solve a distributed load-sharing problem, where a population of smartly-actuated buildings is asked to track a power signal generated by the network operator. The asynchronous forward-backward iteration results in faster convergence at the expense of increased communication.

In the third part of the thesis, we reverse the trade-off and favor computation over communication. We consider the same distributed multi-agent framework with one global coordinator. In the majority of these schemes, the local subproblems that need to be solved are cast as proximal minimization problems. By studying the properties of the proximal operator, we develop a protocol where the coordinator can 'guess' the solution to an agent's local optimization problem and spare some communication rounds. The proposed scheme enables the coordinator to construct a convex set within which the agents' optimizers reside, and to iteratively refine the set every time that an agent is queried. The approach is tested on a load-sharing problem similar to the one described in the previous paragraph while employing the *alternating direction method of multipliers* (ADMM) to solve the problem.

In summary, we study in this thesis the potential of operator splitting methods when applied to constrained optimal control problems. Based on these methods, we design algorithms that successfully compete with existing solution approaches in terms of speed, and others that examine different strengths of these methods in distributed optimization settings. We hope that the research efforts in this thesis will benefit modern optimization-based control applications such as distributed control of smart energy networks, but will also inspire both alternative uses of decomposition methods

and further algorithmic developments.

Keywords: convex optimization, splitting methods, decomposition, distributed optimization, multi-agent systems, proximal operator, Moreau envelope, model predictive control, linear quadratic regulator, asynchronous optimization, smart grid, load sharing, energy networks

Résumé

Les progrès significatifs réalisés ces dernières années, tant au niveau de l'implementation matérielle et du calcul numérique, ont fait du contrôle basé sur l'optimisation en temps réel une option viable pour des applications industrielles avancées. Dans le même temps, le domaine du big data a émergé, cherchant des solutions à des problèmes que les algorithmes d'optimisation classiques sont incapables de fournir. Bien que pour des raisons différentes, ces deux domaines d'application ont poussé la communauté à revisiter la famille des algorithmes d'optimisation communément appelés schémas de décomposition ou méthodes de séparation des opérateurs. Cet intérêt récemment ravivé pour ces méthodes peut être attribué principalement à deux caractéristiques : un coût par itération très bas et une faible empreinte mémoire pour les applications embarquées, et leur capacité à traiter des problèmes de grande échelle via la décomposition lorsqu'il s'agit d'applications liées à l'apprentissage artificiel.

Dans cette thèse, nous concevons des méthodes de décomposition qui s'attaquent à la fois aux problèmes de contrôle centralisé à petite échelle et aux problèmes de contrôle distribué multi-agents à plus grande échelle. En plus de l'objectif classique de concevoir des méthodes plus rapides, nous nous penchons également sur des aspects moins habituels des schémas de decompoisition des opérateurs, qui sont néanmoins critiques pour le contrôle. Dans le cas centralisé, nous proposons un algorithme qui utilise la décomposition pour résoudre exactement un problème de contrôle optimal classique qui, autrement, ne pourrait être résolu qu'approximativement. Dans le cadre des problemes multi-agent, nous proposons deux algorithmes, l'un qui permet une convergence plus rapide et l'autre qui réduit les besoins de communication.

Dans la première partie, nous nous penchons sur le problème du régulateur quadratique linéaire à horizon infini (CLQR). La solution à la version classique sans contrainte du problème (LQR) est connue pour exister sous forme fermée, et offrir des garanties de stabilité pour le système en boucle fermée. L'ajout de contraintes à la formulation LQR rend le problème insoluble analytiquement et difficile à résoudre en raison du nombre infini de variables d'optimisation. Nous proposons d'utiliser la décomposition pour profiter de la solution analytique du LQR, en divisant par conséquent le CLQR en deux sous-problèmes, un problème de moindres carrés de dimensions infinie, et une simple opération de projection d'une séquence infinie sur l'orthant non-positif, les deux étant résolus analytiquement au moyen de la méthode du gradient projeté accéléré. La solution bénéficie de tous les avantages théoriques de l'utilisation d'un coût à horizon infini. Les simulations numériques démontrent la bonne performance pratique du système par rapport au MPC classique.

Les deux autres parties de la thèse traitent des algorithmes d'optimisation distribués appliqués dans des contextes où des agents hétérogènes collaborent sous la présence d'un coordinateur global pour atteindre un objectif. Notre motivation pour étudier ce paramètre provient des applications de réseau intelligent, où les ressources énergétiques coopèrent afin de fournir un service auxiliaire à l'opérateur de réseau sur la base d'un

contrat. La recherche en optimisation distribuée a été principalement motivée par des applications d'apprentissage automatique impliquant de grands ensembles de données et des architectures multi-cœurs parallèles. La nature de nos problèmes d'intérêt, cependant, dicte l'adoption d'un angle différent dans l'analyse des algorithmes d'optimisation distribués. Plus précisément, l'échelle des problèmes qui nous intéresse est généralement moyenne à grande, mais pas énorme, et l'accent est mis sur l'exécution en temps réel plutôt que sur l'exécution hors ligne. Au lieu de processeurs qui effectuent des opérations de lecture/écriture sur une mémoire partagée, nous avons des agents (physiques) régis par des dynamiques qui résolvent souvent des problèmes d'optimisation locale, dont les détails sont censés rester privés, et qui peuvent fonctionner avec des ressources limitées. Plusieurs de ces caractéristiques posent de nouveaux défis et demandent des solutions novatrices.

Dans un premier temps, nous étudions comment accélérer le processus de solution dans de tels environnements multi-agents. Ceci est réalisé en permettant à un plus grand nombre d'agents de se mettre à jour par unité de temps. Nous supposons que les problèmes d'optimisation locale des agents diffèrent les uns des autres en termes de temps de résolution. Cette caractéristique motive l'optimisation asynchrone puisqu'il est plus avantageux que les agents se mettent à jour dans des séquences arbitraires et avec des fréquences arbitraires au lieu de synchroniser leurs mises à jour en fonction de la plus lente d'entre elles. Nous proposons une version asynchrone de l'itération forward/backward qui ne requiert aucune distribution de probabilité pour l'activation des agents. Nous prouvons la convergence linéaire d'une version accélérée de la méthode sous certaines hypothèses structurelles sur les fonctions impliquées. Grâce à la nature générale de l'itération forward/backward, le schéma aboutit à plusieurs nouvelles versions asynchrones et accélérées d'algorithmes couramment utilisés. Nous démontrons la supériorité du schéma par rapport à d'autres algorithmes asynchrones dans un environnement simulé, où nous employons notre version proposée de la méthode du gradient proximal pour résoudre un problème de répartition de charge, où une population de bâtiments intelligemment actionnés doit suivre un signal de puissance généré par l'opérateur du réseau. L'itération asynchrone forward/backward entraîne une convergence plus rapide au détriment d'une communication accrue.

Dans la troisième partie de la thèse, nous inversons le compromis et favorisons le calcul par rapport à la communication. Nous considérons le même cadre multi-agents distribué avec un seul coordinateur global. Dans la majorité de ces schémas, les sous-problèmes locaux qui doivent être résolus sont présentés comme des problèmes de minimisation proximale. En étudiant les propriétés de l'opérateur proximal, nous développons un protocole où le coordinateur peut 'deviner' la solution au problème d'optimisation locale d'un agent et éviter certaines étapes de communication. Le schéma proposé permet au coordinateur de construire un ensemble convexe dans lequel résident les optimiseurs des agents et d'affiner de façon itérative l'ensemble à chaque fois qu'un agent est interrogé. L'approche est testée sur un problème de répartition de charge similaire à celui décrit dans le paragraphe précédent tout en utilisant la méthode des multiplicateurs à direction alternée pour résoudre le problème.

En résumé, nous étudions dans cette thèse le potentiel des méthodes de division des

opérateurs lorsqu'elles sont appliquées à des problèmes de contrôle optimal contraint. Sur la base de ces méthodes, nous concevons des algorithmes qui concurrencent avec succès les solutions existantes en termes de vitesse, et d'autres qui examinent les différentes forces de ces méthodes dans des situations d'optimisation distribués. Nous espérons que les efforts de recherche de cette thèse profiteront aux applications modernes de contrôle basées sur l'optimisation, comme le contrôle distribué des réseaux d'énergie intelligents, mais inspireront aussi bien des utilisations alternatives des méthodes de décomposition que de nouveaux développements algorithmiques.

Mots Clés : optimisation convexe, méthodes de fractionnement, décomposition, optimisation distribuée, systèmes multi-agents, opérateur proximal, enveloppe de Moreau, contrôle prédictif de modèle, régulateur quadratique linéaire, optimisation asynchrone, réseau intelligent, partage de charge, réseaux d'énergie.

Contents

1 Introduction **1**
- 1.1 Outline and Contribution . . . 3
- 1.2 Publications . . . 3

2 Preliminaries **7**
- 2.1 Convex Optimization . . . 8
  - 2.1.1 Function properties . . . 8
  - 2.1.2 Fenchel Duality . . . 8
  - 2.1.3 ϵ-subgradient . . . 9
- 2.2 Proximal Splitting Methods . . . 10
  - 2.2.1 Proximal Point Algorithm (PPA) . . . 10
  - 2.2.2 Moreau Envelope . . . 11
  - 2.2.3 Proximal Gradient Method (PGM) . . . 11
  - 2.2.4 Dual Proximal Gradient Method and Alternating Minimization Algorithm . 12
  - 2.2.5 Douglas-Rachford Splitting and Alternating Direction Method of Multipliers 13
  - 2.2.6 Convergence Rates and Inertial Acceleration . . . 14
- 2.3 Monotone Operators . . . 17
  - 2.3.1 Basics on Monotone Operators . . . 17
  - 2.3.2 The Krasnosel'skiĭ-Mann Iteration . . . 19
- 2.4 Model Predictive Control . . . 19
  - 2.4.1 Invariance and Lyapunov Stability . . . 20
  - 2.4.2 Linear Quadratic Regulator . . . 20
  - 2.4.3 Linear Model Predictive Control . . . 21
- 2.5 Distributed Ancillary Service Provision with Controllable Buildings . . . 22

3 Infinite-Horizon Constrained Linear Quadratic Regulator **27**
- 3.1 Introduction . . . 27
- 3.2 Related work . . . 28
- 3.3 Problem statement . . . 29
- 3.4 Dualization . . . 30
- 3.5 Solution using Accelerated Dual Proximal Gradient Method . . . 31
- 3.6 Convergence results . . . 35
- 3.7 Computational aspects and warm-starting . . . 39

3.7.1 Stepsize selection . . . 39
3.7.2 Complexity . . . 40
3.7.3 Warm-starting . . . 41
3.8 Examples . . . 41
3.8.1 Toy system . . . 41
3.8.2 Quadcopter system . . . 45
3.8.3 Timings . . . 45
3.9 Conclusion . . . 49
3.10 Appendices . . . 50
3.10.1 Required Operator Theory . . . 50
3.10.2 Boundedness of several operators . . . 51
3.10.3 Backtracking stepsize rule . . . 53

4 Inertial Parallel and Asynchronous Forward-Backward Iteration for Distributed Convex Optimization 56
4.1 Introduction . . . 56
4.2 Related work . . . 57
4.3 Problem description . . . 58
4.3.1 Asynchronous updates . . . 58
4.3.2 An asynchronous inertial forward-backward iteration . . . 58
4.3.3 Main contribution . . . 62
4.4 Convergence proof . . . 62
4.4.1 Express delayed variables as additive error . . . 63
4.4.2 Isolate the error . . . 64
4.4.3 Bound the error recursively . . . 64
4.4.4 Bound the error with respect to the maximum distance from the set of fixed points . . . 65
4.4.5 Condition for convergence . . . 65
4.5 Connection to other methods . . . 66
4.5.1 Gradient descent . . . 66
4.5.2 Proximal gradient . . . 66
4.5.3 Other methods . . . 66
4.6 Application: Distribution network real-time dispatch . . . 67
4.6.1 Modeling the agents . . . 67
4.6.2 Modeling the dispatch problem . . . 68
4.6.3 Simulation setup . . . 69
4.7 Conclusion . . . 71
4.8 Appendices . . . 72
4.8.1 Proof of Lemma 13 . . . 72
4.8.2 Proof of Lemma 14 . . . 72
4.8.3 Proof of Lemma 15 . . . 74
4.8.4 Proof of Lemma 16 . . . 75
4.8.5 Proof of Thoerem 7 . . . 77
4.8.6 Cocoercivity and quasi-strong monotonicity of S . . . 78

5 Estimating the Proximal Operator **80**
5.1 Introduction . . . 80
5.2 Related work . . . 81
5.3 Problem description . . . 82
5.4 Estimating the solution to a proximal minimization problem . . . 83
5.5 Communication . . . 88
5.5.1 Non worst-case candidates . . . 91
5.5.2 The algorithmic schemes . . . 91
5.6 Convergence . . . 93
5.6.1 Convergence of ADMM . . . 94
5.6.2 Convergence of randomized coordinate descent PGM . . . 95
5.7 Applications . . . 96
5.7.1 Distributed Load Sharing . . . 96
5.7.2 Optimal Price Adjustment . . . 99
5.7.3 Randomized Coordinate Load Sharing . . . 102
5.8 Conclusion . . . 105
5.9 Appendices . . . 105
5.9.1 Proof of Proposition 2 . . . 105
5.9.2 Proof of Proposition 5 . . . 106
5.9.3 Proof of Theorem 8 . . . 106
5.9.4 Proof of Proposition 6 . . . 107
5.9.5 Proof of Proposition 7 . . . 108
5.9.6 Proof of Theorem 10 . . . 109

6 Extensions and Conclusions **114**
6.1 Infinite horizon control . . . 114
6.2 Asynchronous optimization . . . 115
6.3 Learning a proximal problem . . . 117

A Appendix **120**
A.1 Derivation of the Alternating Direction Method of Multipliers from Douglas-Rachford Splitting . . . 120

List of Figures

2.1 The Moreau envelope of a linear function constrained in [-3,1]. The proximal evaluations at several randomly generated points gradually form the envelope. Note that the minimum values coincide. 11

2.2 Sketch of a network of buildings with a battery energy storage system (agents) and a global system operator (coordinator). The agents communicate their individual consumption vectors $(p_i^{\text{cb}}, p^{\text{bess}})$ and the coordinator uses them in order to form an incentive signal (e) that will be in turn communicated to them. 23

3.1 Histogram of $T^\infty = \max_k\{T^k\}$ for 750 initial conditions of the 2 state system sampled from a normal distribution around $(-3, 0.3)$ with covariance matrix $\text{diag}(4, 0.4)$. 42

3.2 Reachable sets for several horizon lengths and the LQ terminal set. The computation was done using the MPT3 toolbox [HKJM13]. It is apparent that a short horizon length reduces significantly the feasible region of the problem. 44

3.3 Comparison of MPC with finite horizon length, for several horizons, with CLQR. The horizontal axis corresponds to the optimal horizon length T^* per initial condition. As T^* increases, the corresponding states are sampled further from the origin. MPC with terminal set is depicted with the solid lines, while without terminal set with dashed lines. CLQR is performed both with the weight sequence, denoted as T_w^∞ CLQR, and without the weights, denoted as T^∞ CLQR. 44

3.4 Evolution of the ratios $\frac{T_{\min}}{T^*}$, $\frac{T_w^\infty}{T^*}$ and $\frac{T^\infty}{T^*}$ for the sampled initial conditions. 45

3.5 Histogram of $T^\infty = \max_k\{T^k\}$ for 272 initial conditions sampled with a Hit-And-Run algorithm. 46

3.6 Box plots for the average number of iterations of the warm-starting policy, given 0.5% and 1% uniform perturbations of the initial state are depicted. The horizontal line inside the box corresponds to the median, the edges of the box are the 25^{th} and 75^{th} percentiles while the horizontal lines outside the boxes correspond to the most extreme data points not considered outliers. Finally, the colored dots correspond to mean values. Warm-starting improves the performance of the suggested method in both cases, in all the depicted statistical measures. 46

3.7 Solve times of the CLQR solver versus CPLEX, MOSEK and QPgen for the toy system, sampled from 750 initial conditions. In **(a)**, the CLQR generally requires less time than solving a series of finite horizon optimal control problems, leading to the same optimal solution. Only the solver QPgen clearly outperforms our proposed approach, which makes use of the same, in principle, splitting method, enhanced by several add-on's that enable speedup. In **(b)** the three solvers are faster. The large variance of the CLQR approach in comparison to the others is a typical characteristic of first-order methods, and can be significantly reduced with appropriately conditioning of the problem. 48
3.8 Iteration and factorization counts for the toy system. It can be seen in **(a)** that more than 80% of the problems needed less than 400 iterations in order to converge, while the maximum number observed is around 5000 iterations. The mean value of factorizations in **(b)** is seven while no problem instance needed more than 25 factorizations. 49

4.1 Buffer $\mathcal{W}$ is filled by the Write thread and emptied by the Compute thread. Similarly, $\mathcal{R}$ is filled by the Compute thread and emptied by the Read thread. The scheme executes continuously and asynchronously, both at the coordinator level (concurrent threads) and at the interface between the agents and the coordinator. 61
4.2 An update is about to occur at $k+1$. From Assumption 5, the observed agent will update again no later than τ time epochs after z^i was communicated to the coordinator. Consequently, y_{write} cannot be further than 2τ from the next update, while $y_{\text{write}}^{\text{prev}}$ cannot be further than 3τ. 63
4.3 Distance from optimizer VS wall-clock time. 70
4.4 The reference signal to be tracked is depicted in red. The contribution of the BESS is colored in dark blue, that of the medium scale buildings in lighter blue and that of the two large buildings in green. The contribution of the small buildings is colored in pink, but is hardly visible due to their small capacity and despite their large population. 71

5.1 The epigraph of the envelope function $f^\gamma(z)$ and that of the quadratic upper bounding function $\overline{f}^\gamma(z; z_1)$ are illustrated. For any point centered at v that lies ϵ below the epigraph of $\overline{f}^\gamma(z; z_1)$, the ϵ-subdifferential set is depicted as the normal cone of the set constructed by the tangent hyperplanes of epi $(\overline{f}^\gamma(z; z_1))$, depicted with the grey lines, at $(v, \overline{f}^\gamma(v; z_1) - \epsilon)$. The actual gradient $\nabla f^\gamma(v)$ is contained in all $\partial_\epsilon \overline{f}^\gamma(v; z_1)$ for $\epsilon \geq \epsilon^*(v; z_1)$. As ϵ is getting smaller, the subdifferential sets shrink, and converge to $\partial_{\epsilon^*(v;z_1)} \overline{f}^\gamma(v; z_1)$, depicted by the normal cone to the set illustrated by the blue lines in the right figure. 84
5.2 Assuming two query points z_1 and z_2, $\nabla f^\gamma(v)$ can be located in the intersection of the two normal cones, which is sketched in purple color. In addition, the function $\overline{f}^\gamma(z \mid \mathcal{J}) = \text{conv}\{\overline{f}^\gamma(z; z_1), \overline{f}^\gamma(z; z_2)\}$ is drawn in deep red color. It is evident that the normal cone corresponding to the intersection of the two sets (left) is identical to the set $\partial_{\epsilon^*(v|\mathcal{J})} \overline{f}^\gamma(v \mid \mathcal{J})$ on the right. 87
5.3 Performance in terms of communication savings. 100
5.4 Tracking of a reference by a mix of small (pink) and medium scale (blue) buildings. 101

5.5 Performance in terms of communication savings for the price adjustment (dispatch) problem. 103
5.6 Performance in terms of communication savings for the randomized coordinate descent implementation of the load sharing problem. 104

List of Tables

3.1 Closed loop comparison between CLQR and MPC 49

4.1 Micro-grid case study overview 68
4.2 Accuracy reached within $Ts = 40$sec. 70
4.3 Average number of updates per agent within simulation time. 70

5.1 Micro-grid case study overview 98

Chapter 1

Introduction

The impressive developments that have occurred in recent years both in hardware implementations and in numerical computing have resulted in rapidly falling prices in sensor and actuator technologies. As a consequence, there is a massive deployment of sensing and communication hardware along with embedded computer platforms in a range of devices, a phenomenon that brought about a twofold change. First, once prohibitively expensive technologies are now available as consumer goods, as for example powerful smartphone devices and drones. Second, well-established industrial paradigms are questioned. For example, in modern electricity grids, the deployment of sensors and actuators along with increased computational capacity transforms traditionally passive loads into active producer-consumer (prosumer) entities, which are capable of engaging into decision making and of negotiating the provision of services to the network operator.

It is fair to say that mathematical optimization has been playing an active role for decades in enhancing the decision-making capacity of systems like the ones mentioned above. For example, the mathematical optimization framework has given rise to *model predictive control* (MPC), a control paradigm where one chooses the best possible control action from a set of candidate choices based on some performance criterion while guaranteeing safety of operations. MPC has been successfully applied to the process industry and has nowadays become a control standard. However, and in contrast to process industry applications, the current challenges push the limits of optimization-based solutions in terms of two specifications: increased computational speed and ability to deal with problems in a distributed way. The exploration of novel optimization algorithms that are capable of satisfying these specifications triggered interest in revisiting the family of optimization algorithms commonly known as *decomposition schemes* or *operator splitting methods*. These algorithms have proven to be promising candidates thanks to two characteristics: Computationally low per-iteration cost along with small memory footprint for time-critical (control) applications that run on limited resources, and their capacity to deal with problems that need to be solved in a distributed way due to size, physical limitations or privacy requirements.

The speed requirement has triggered considerable interest in the control community, and as a result many efficient high-speed solvers have been developed for both linear and nonlinear control, based on either *first-order methods* (FiOrdOs [Ull11], QPgen [GB],[GB15], DuQuad [NP15]), *interior point (IP) methods* (FORCES [DZZMJ12], CVXGEN [MB12]) and *active set methods* (QPOASES [FBD08]). The application of decomposition methods in control problems started mak-

ing its appearance more recently [OSB13], [GTSJ15], [GB], [GB15], [PB14b], and has demonstrated promising performance outcomes. Moreover, the ability of the latter methods to break the problem into simpler subproblems implies that there might be more to benefit from than only computational savings. Knowing that linear MPC is only an approximation to an infinite-dimensional constrained optimal control problem, one question that naturally arises is whether decomposition can be useful in helping to tackle the original problem while staying on par with MPC solvers in terms of speed and efficiency. The practical advantages of solving the original problem instead of the approximation are not negligible either since the optimal control law comes with stability guarantees and for a larger set of initial states. The resulting scheme could thus be a competitor to linear MPC.

The low per-iteration cost that motivates the use of operator splitting methods in embedded MPC platforms is yet another reason that these algorithms are preferred for large-scale applications. On the downside, however, these methods suffer from sublinear convergence rates, achieving initially quick progress towards some optimal point that subsequently levels off. Several ways to remedy this behavior have been proposed, including alternative metric selections [CV14; RL15; GB; GB15], relaxation strategies and inertial acceleration [AA01]. When it comes to large-scale machine learning-related applications, however, the inherent parallelization potential of operator splitting schemes can result in significant computational speedups, compensating for the often poor convergence rates. With that in mind, a considerable amount of research has been produced in asynchronous implementations, *i.e.,* in problems when, *e.g.,* the decision vector is stored in the shared memory space of a multicore computer and can be accessed and altered by the cores in an intermittent manner [LW15; PXYY16]. For example, a fitting problem might be so large that the features are assigned to different cores. Each core trains a block of features using a local model, and the blocks are updated by the cores in a continuous and asynchronous way. A reasonable thought is, therefore, to combine acceleration strategies which improve the theoretical convergence rate with asynchronous implementations and study the performance of the resulting schemes. The development of such schemes need not necessarily be beneficial only for multicore architectures, but rather for any setting that constitutes an inhomogeneous mixture of agents, the computations of which do not occur at a common rate.

Sometimes the need for solving a problem in a distributed manner does not stem from speedup requirements or size restrictions, but rather from physical limitations or privacy requirements. Take, for example, a population of physical agents that operate under a global coordinator in order to collaboratively solve a problem. In a convex setting, the local subproblems that need to be solved are often cast as proximal minimization problems, a term used to describe optimization problems that are regularized by the addition of a properly scaled quadratic term in their objective [PB14a]. The agents' data must remain private and not be shared with the coordinator, thus communication between the two parties looks unavoidable. However, the special structure of proximal minimization problems allows the coordinator to gradually construct a model of the agent, which is iteratively refined every time that a communication round takes place. This observation has several implications and can be used in different ways. One possible use is communication reduction in multi-agent setups when communication is costly. At the same time, such a communication protocol can raise privacy issues since the coordinator is able to gradually learn the value functions of the network agents. Studying such scenaria can, therefore, be valuable.

1.1 Outline and Contribution

This dissertation is organized into five main chapters. Chapter 2 introduces some preliminary material that is necessary for the main chapters of the thesis. We start by giving some basic definitions from convex optimization and proceed with a presentation of the proximal operator, its relation to several operator splitting algorithms and its interpretation as an envelope function. We also briefly discuss the convergence properties of first-order methods and existing techniques for accelerating them. We subsequently give an introduction to monotone operators and represent the mentioned algorithms in this framework, *i.e.,* as fixed-point iterations. The next section gives a short introduction to the basics of linear MPC, while the chapter concludes with a mention to electrical distribution grids and the services that can be provided by aggregations of smart buildings, which is the demonstration platform for the methods developed in this thesis.

In Chapter 3 we propose a method that solves the infinite-horizon constrained LQR problem using an accelerated version of the proximal gradient method. We prove that the method converges to the optimal infinite-horizon sequence and we derive convergence rates for the function values and the iterates. To the best of our knowledge, the proposed scheme is the least computationally intensive and the most scalable of the existing ones. Moreover, an exhaustive numerical analysis demonstrates the good convergence properties of the algorithm when warm-started in order to solve a sequence of optimal control problems, and that it is comparable to (and often faster than) corresponding MPC implementations.

The remaining parts of the thesis depart from centralized embedded control and focus on distributed optimization in multi-agent environments in the presence of a global coordinator. The agents are represented as controllable loads, namely linear models of controlled commercial buildings and battery energy storage systems. Two cases are considered. In Chapter 4 we propose an asynchronous and accelerated optimization splitting scheme that solves a cooperative tracking problem using the mix of agents. Contrary to the majority of asynchronous optimization algorithms, the proposed scheme does not assume any probability distribution for the activation of the agents. We prove linear convergence of the algorithm under some structural assumptions on the involved functions.

In Chapter 5 we reverse the trade-off and favor computation over communication. Making use of fundamental properties from convex analysis we show that an optimizer of an instance of a proximal minimization problem can be located inside a convex set that can be explicitly generated as the intersection of ellipsoids. The agents solve local proximal minimization problems. As a result of the analysis, we propose a communication protocol where the coordinator constructs and maintains one set per agent, so as to locate its solution without communicating. We match the protocol to several distributed optimization algorithms and demonstrate its capability in terms of communication reduction in several examples.

Finally, we conclude the dissertation in Chapter 6 by discussing some possible extensions.

1.2 Publications

The material for Chapters 2, 3, 4 and 5 was taken from a series of papers. More specifically, we have the following:

Parts of Chapter 2 are taken from the following publication:

- G. Stathopoulos, H. Shukla, A. Szucs, Y. Pu, and C. N. Jones, "Operator splitting methods in control", *Foundations and Trends® in Systems and Control*, vol. 3, no. 3, pp. 249–362, 2016. [SSSPJ16]

  Manuscript available at https://infoscience.epfl.ch/record/225464

Chapter 3 is based on the following two publications:

- G. Stathopoulos, M. Korda, and C. N. Jones, "Solving the infinite-horizon constrained LQR problem using splitting techniques", in *19th IFAC World Congress*, 2014. [SKJ14]

  Manuscript available at https://infoscience.epfl.ch/record/252916?&ln=en

- G. Stathopoulos, M. Korda, and C. N. Jones, "Solving the Infinite-Horizon Constrained LQR Problem using Accelerated Dual Proximal Methods", *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1752–1767, 2017. [SKJ17]

  Manuscript available at https://infoscience.epfl.ch/record/225463?ln=en

  C++ code is publicly available on https://github.com/stathopog/InfHorCLQR

Chapter 4 is taken from the following submission:

- G. Stathopoulos and C. N. Jones, "An Inertial Parallel and Asynchronous Forward-Backward Iteration for Distributed Convex Optimization", *Submitted to Journal of Optimization Theory and Applications*, 2018. [SJ18a]

  Manuscript available at https://arxiv.org/abs/1706.00088

  A Julia notebook is available on https://github.com/stathopog/AsyncInertialFBS

Chapter 5 is based on the following two manuscripts:

- G. Stathopoulos and C. N. Jones, "A coordinator-driven communication reduction scheme for distributed optimization using the projected gradient method", in *Proceedings of the 17th IEEE European Control Conference, ECC 2018, Limassol, Cyprus*, 2018. [SJ18c]

  Manuscript available at https://infoscience.epfl.ch/record/253216?ln=en

- G. Stathopoulos and C. N. Jones, "Communication reduction in distributed optimization via estimation of the proximal operator", *Submitted to IEEE Transactions on Control of Network Systems*, 2018. [SJ18b]

  Manuscript available at https://arxiv.org/abs/1803.07143

Published work related to the topics of this thesis but not discussed or only cited is:

- G. Stathopoulos, A. Szucs, Y. Pu, and C. N. Jones, "Splitting methods in control", in *Proceedings of the 13th IEEE European Control Conference, ECC 2014, Strasbourg, France*, 2014, pp. 2478 –2483. [SSPJ14]

Manuscript available at https://infoscience.epfl.ch/record/201988?ln=en

- T. T. Gorecki, A. Bitlislioglu, G. Stathopoulos, and C. N. Jones, "Guaranteeing input tracking for constrained systems: theory and application to demand response", in *American Control Conference (ACC).*, 2015, pp. 232–237. [GBSJ15]

  Manuscript available at https://infoscience.epfl.ch/record/201865?ln=en

- Y. Liu, J. H. Hours, G. Stathopoulos, and C. N. Jones, "Real-time distributed algorithms for nonconvex optimal power flow", in *2017 American Control Conference (ACC)*, 2017, pp. 3380–3385. [LHSJ17]

  Manuscript available at https://infoscience.epfl.ch/record/225467?ln=en

- L. Ferranti, G. Stathopoulos, T. Keviczky, and C. N. Jones, "Constrained LQR Using Online Decomposition Techniques ", in *Decision and Control (CDC), 2016 IEEE 55th Annual Conference on*, 2016. [FSKJ16]

  Manuscript available at https://arxiv.org/pdf/1609.05799.pdf

- H. Ferreau, S. Almèr, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, G. S. J.L. Jerez, and C. Jones, "Embedded Optimization Methods for Industrial Automatic Control ", *IFAC-PapersOnLine*, vol. 50, pp. 13 194–13 209, 1 2017. [FAVDFDJJJ17]

  Manuscript available at https://www.sciencedirect.com/science/article/pii/S2405896317325764

Chapter 2

Preliminaries

Notation

We follow the notation of classical optimization textbooks like [BV04] and [Ber15].

We use $\mathbb{R}$ to denote the set of real numbers and $\mathbb{N}$ to denote the set of natural numbers. The set of nonnegative real numbers is denoted by $\mathbb{R}_+$ and the set of positive real numbers by $\mathbb{R}_{++}$. The set of column vectors with length n is denoted by $\mathbb{R}^n$, while the set of real-valued $m \times n$ matrices by $\mathbb{R}^{m \times n}$. We use parentheses to construct column vectors from scalar or vector elements, *e.g.,* if $a \in \mathbb{R}^n, b \in \mathbb{R}^m$, then

$$(a, b) = [a^\top \; b^\top]^\top \in \mathbb{R}^{n+m} \; .$$

The inner product between two vectors (or infinite dimensional sequences) is denoted by $\langle \cdot, \cdot \rangle$.

We use the inequality symbols $\succeq$ and $\succ$ to denote positive semidefiniteness and positive definiteness, respectively, of a symmetric real matrix. For example, for $Q \in \mathbb{R}^{n \times n}$ it holds that $Q \succeq 0$ if and only if $x^\top Q x \geq 0$ for any nonzero $x \in \mathbb{R}^n$. We denote by I the identity matrix of proper dimensions that will be clear from the context. We use the regular inequality symbols $\leq, \geq, <, >$ between vectors to denote componentwise inequality.

For a function $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$, we denote the *effective domain* of f by $\operatorname{dom}(f)$, where

$$\operatorname{dom}(f) = \{x \in \mathbb{R}^n \mid f(x) < +\infty\} \, ,$$

and its *epigraph* by $\operatorname{epi}(f)$. The epigraph is the subset of $\mathbb{R}^{n+1}$ given by

$$\operatorname{epi}(f) = \{(x, s) \mid x \in \mathbb{R}^n, s \in \mathbb{R}, f(x) \leq s\} \, .$$

The subdifferential of f at a point x is denoted by $\partial f(x)$, while $\nabla f(x) \in \mathbb{R}^n$ denotes the gradient of f at x, provided that f is differentiable at x. The family of functions $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$ that are closed proper and convex (see Section 2.1 for the definitions) are denoted by $f \in \Gamma_0(\mathbb{R}^n)$.

A polyhedron is an intersection of a finite number of closed halfspaces in $\mathbb{R}^n$, and a polytope is a bounded polyhedron. The expected value of a random variable X is denoted by $\mathrm{E}[X]$, the conditional expectation of X given Y by $\mathrm{E}[X \mid Y]$, while the probability that $X = x$ by $\mathbb{P}[X = x]$.

Finally, we typically use the asterisk symbol $*$ to denote optimality (optimal value of a function, optimizer vector or set of optimizers), while we use the star symbol $\star$ to denote the Fenchel conjugate

of a convex function (see Section 2.1). In addition, superscripts k are used to denote algorithmic iterations in all chapters except for Chapter 4, where the iterations are denoted by a subscript.

2.1 Convex Optimization

In this section we summarize some results from convex analysis that are essential for the comprehension of this thesis.

2.1.1 Function properties

Definition 1 (Convexity). *Let $\mathcal{C}$ be a convex subset of $\mathbb{R}^n$. The function $f : \mathcal{C} \mapsto \mathbb{R}$ is* convex *if*

$$f(\theta z + (1-\theta)x) \leq \theta f(z) + (1-\theta)f(x), \quad \forall z, x \in \mathcal{C},\ \forall \theta \in [0,1] \ .$$

Definition 2 (Strong Convexity). *We say that a differentiable function f is* strongly convex *if there exists $\sigma > 0$ such that*

$$f(z) \geq f(x) + \langle \nabla f(x), z - x \rangle + \frac{\sigma}{2}\|z - x\|^2, \quad \forall z, x \in \mathcal{C} \ .$$

The constant σ is referred to as the stong convexity constant (or modulus).

Remark 1. *Although differentiability is not necessary for strong convexity to hold, whenever we consider a strongly convex function it will also be differentiable in the context of this work. Therefore the above definition is functional for our purposes.*

Definition 3 (Properness). *The function $f : \mathcal{Z} \mapsto [-\infty, +\infty]$, for $\mathcal{Z} \subset \mathbb{R}^n$, is* proper *if $f(z) < +\infty$ for at least one $z \in \mathcal{Z}$ and $f(z) > -\infty$ for all $z \in \mathcal{Z}$.*

Definition 4 (Closedness). *The function $f : \mathcal{Z} \mapsto [-\infty, +\infty]$ is* closed *if $\mathrm{dom}(f)$ is closed and $\mathrm{epi}(f)$ is closed.*

Definition 5 (Lipschitz continuous gradient). *We say that a differentiable function $f : \mathbb{R}^n \mapsto \mathbb{R}$ has* Lipschitz continuous gradient *with constant L if*

$$\|\nabla f(z) - \nabla f(x)\| \leq L\|z - x\|, \quad \forall z, x \in \mathbb{R}^n \ .$$

We denote hereafter the family of functions $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$ that are closed proper and convex by $f \in \Gamma_0(\mathbb{R}^n)$.

2.1.2 Fenchel Duality

Consider the problem

$$\text{minimize} \quad f(z) + g(Az) \ ,$$

with variable $z \in \mathbb{R}^n$, where $A \in \mathbb{R}^{m \times n}$ and $f \in \Gamma_0(\mathbb{R}^n)$, $g \in \Gamma_0(\mathbb{R}^m)$. The problem can be written in the equivalent constrained form

$$\begin{array}{ll} \text{minimize} & f(z) + g(x) \\ \text{subject to} & Az = x \ , \end{array} \tag{2.1}$$

with variables $z \in \mathbb{R}^n$ and $x \in \mathbb{R}^m$. By introducing the dual variables $\lambda \in \mathbb{R}^m$, we can formulate the *Lagrangian function*

$$L(z, x; \lambda) := f(z) + g(x) + \langle \lambda, Az - x \rangle \ , \tag{2.2}$$

and, subsequently, the *dual function*

$$\begin{aligned} q(\lambda) &:= \inf_{z \in \mathbb{R}^n, x \in \mathbb{R}^m} \{ f(z) + g(x) + \langle \lambda, Az - x \rangle \} \\ &= \inf_{z \in \mathbb{R}^n} \{ f(z) + \langle \lambda, Az \rangle \} + \inf_{x \in \mathbb{R}^m} \{ g(x) - \langle \lambda, x \rangle \} \ . \end{aligned}$$

We define the *(Fenchel) conjugate* of g as

$$g^\star(\lambda) := \sup_{x \in \mathbb{R}^m} \{ \langle \lambda, x \rangle - g(x) \} = - \inf_{x \in \mathbb{R}^m} \{ g(x) - \langle \lambda, x \rangle \} \ , \tag{2.3}$$

where $g : \mathbb{R}^m \mapsto \mathbb{R} \cup \{-\infty, +\infty\}$. Consequently, the minimization (2.1) can be written as

$$\text{minimize} \quad f^\star(-A^\top \lambda) + g^\star(\lambda) \ , \tag{2.4}$$

with variable $\lambda \in \mathbb{R}^m$.

Existence of primal and dual optimal solutions in the Fenchel duality framework as well as strong duality results are provided in several sources (see, *e.g.,* , [Roc70]). They are summarized in [Ber15, Proposition 1.2.1] and, for our cases of interest, we will assume that they all hold.

We subsequently state below a fundamental theorem that stems from the Fenchel duality framework.

Theorem 1 (Conjugate Subgradient Theorem). *Let $f \in \Gamma_0(\mathbb{R}^n)$. The following relation holds:*

$$u \in \partial f(z), \text{ for some } z \in \mathbb{R}^n, \Leftrightarrow z \in \partial f^\star(u) \ .$$

2.1.3 ϵ-subgradient

Definition 6 (ϵ-subgradient). *Given $f \in \Gamma_0(\mathbb{R}^n)$ and a scalar $\epsilon > 0$, we say that a vector g is an* ϵ-subgradient *of f at $x \in \text{dom}(f)$ if*

$$f(z) \geq f(x) + \langle g, z - x \rangle - \epsilon, \quad \forall z \in \mathbb{R}^n \ .$$

The ϵ-subdifferential *is the set of all ϵ-subgradients of f at x, i.e.,*

$$\partial_\epsilon f(x) := \{ g \mid f(z) \geq f(x) + \langle g, z - x \rangle - \epsilon, \quad \forall z \in \mathbb{R}^n \} \ .$$

It holds that

$$\partial_{\epsilon_1} f(x) \subset \partial_{\epsilon_2} f(x) \quad 0 < \epsilon_1 < \epsilon_2 \ ,$$

and that

$$\lim_{\epsilon \to 0} \cap \partial_\epsilon f(x) = \partial f(x) \ .$$

2.2 Proximal Splitting Methods

For $f \in \Gamma_0(\mathbb{R}^n)$, its *proximal operator* $\mathbf{prox}_f : \mathbb{R}^n \mapsto \mathbb{R}^n$ is defined as

$$\mathbf{prox}_{\gamma f}(x) := \underset{z \in \mathbb{R}^n}{\arg\min} \left\{ f(z) + \frac{1}{2\gamma} \|z - x\|^2 \right\} \ , \tag{2.5}$$

for some $\gamma > 0$. The proximal operator firstly appeared in the seminal work of Moreau [Mor62; Mor65]. The operator is evaluated at a given point x and looks for a minimizer that makes a compromise between the minimizer of the function f and the point x.

The proximal operator is the source of numerous iterative approximation methods that find a minimizer of the convex function f. The main idea is that the generated sequence $\{z^k\}$ is obtained by solving at each k an approximate problem,

$$z^{k+1} \in \underset{z \in \mathbb{R}^n}{\arg\min} \ F^k(z) \ ,$$

where F^k is a function that approximates f [Ber15, Chapter 5]. More details about the properties of the operator can be found in the recent survey [PB14a]. The results discussed in this section have appeared in [SSSPJ16] and [Ber15].

2.2.1 Proximal Point Algorithm (PPA)

PPA is the most straightforward application of the proximal operator to the minimization of the function f described above. It is given by the iteration

$$z^{k+1} := \mathbf{prox}_{\gamma^k f}(z^k) \ , \tag{2.6}$$

i.e., it minimizes f while not moving too far away from the previous minimizer. The distance to z^k is controlled by the sequence $\{\gamma^k\}$. The algorithm converges for $\gamma^k > 0$ and $\sum_{k=1}^{\infty} \gamma^k = \infty$ while satisfying a cost function decrease property at each iterate, even if the set of optimizers $\mathcal{Z}^*$ is empty or $f^* = -\infty$ [Ber15, Chapter 5]. In addition, the rate of convergence varies from sublinear to superlinear[1], depending on the *curvature of the function f close to $\mathcal{Z}^*$ and the choice of γ^k* [Ber15, Proposition 5.1.4]. The aforementioned properties render PPA a robust algorithm that works under mild assumptions.

The method was introduced in [Mar70; Mar72], while a more general form than the one presented here appeared in [Roc76]. Convergence properties have been further analyzed in [GÜ91]. On the downside, the PPA is only useful when the proximal operator of f is easy to compute.

[1] The different types of convergence are discussed in Section 2.2.6.

2.2.2 Moreau Envelope

An interesting interpretation of the proximal minimization problem (2.5) can be obtained by considering the value function

$$f^\gamma(x) = \min_z \left\{ f(z) + \frac{1}{2\gamma}\|z - x\|^2 \right\} \ . \tag{2.7}$$

Function (2.7) is also known as the *Moreau envelope of* f. In general, $f^\gamma(x)$ approximates f from below and, the smaller the parameter γ, the tighter the approximation of f by the envelope.

When f is a closed proper convex function, the Moreau envelope is *convex and differentiable, with Lipschitz continuous gradient with constant* $1/\gamma$. Moreover, *the set of minima of* f *and of* f^γ *coincide.* A discussion about the envelope and its properties can be found in [Ber15, Section 5.1], while the Moreau envelope of a scalar linear function is depicted in Figure 2.2. It is also shown in [Ber15, Proposition 5.1.7] that *the unique solution to the proximal minimization* $z^\gamma(x) = \mathbf{prox}_{\gamma f}(x)$ *can be written as*

$$z^\gamma(x) = x - \gamma \nabla f^\gamma(x) \ , \tag{2.8}$$

i.e., it is the point at which the gradient iteration of f^γ*, evaluated at* x*, lands.*

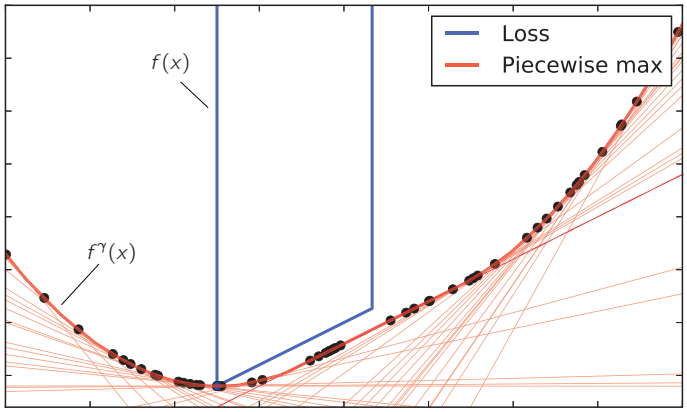


Figure 2.1: The Moreau envelope of a linear function constrained in [-3,1]. The proximal evaluations at several randomly generated points gradually form the envelope. Note that the minimum values coincide.

2.2.3 Proximal Gradient Method (PGM)

Consider the case that we want to minimize $f(z) + g(z)$, where $f \in \Gamma_0(\mathbb{R}^n)$ is *differentiable with Lipschitz continuous gradient* $L > 0$ and $g \in \Gamma_0(\mathbb{R}^n)$. The proximal gradient method is the iteration

$$z^{k+1} := \mathbf{prox}_{\gamma^k g}\left(z^k - \gamma^k \nabla f(z^k)\right) \ , \tag{2.9}$$

where $\gamma^k > 0$ is a stepsize, either constant or determined by line search methods.

It is interesting to rewrite the method as

$$\begin{aligned} z^{k+1} &= \arg\min_{z\in\mathbb{R}^n} \left\{ g(z) + \frac{1}{2\gamma^k}\|z - (z^k - \gamma^k \nabla f(z^k))\|^2 \right\} \\ &= \arg\min_{z\in\mathbb{R}^n} \left\{ g(z) + f(z^k) + \langle \nabla f(z^k), z - z^k \rangle + \frac{1}{2\gamma^k}\|z - z^k\|^2 \right\} \quad . \end{aligned}$$

Consequently, the method minimizes the sum of the (possibly) nonsmooth function g and a *quadratic approximation* of the continuously differentiable function f centered at the previously computed optimizer z^k. If the stepsize is chosen to be fixed in the range $(0, 1/L]$, then the quadratic model upper bounds f around z^k and the method can be shown to converge at a rate $O(1/k)$ while satisfying a cost function decrease property similar to that of PPA [Ber15, Proposition 6.6.3]. In practice the method converges for any $\gamma \in (0, 2/L)$, but the quadratic model does not necessarily act as an upper bound for stepsizes that are larger than $1/L$.

2.2.4 Dual Proximal Gradient Method and Alternating Minimization Algorithm

PGM can be applied to solve the dual problem (2.4) when the composite minimization problem involves more complicated arguments, as described in (2.1). Similar to (2.9), the dual proximal gradient iteration now takes the form

$$\lambda^{k+1} := \mathbf{prox}_{\gamma^k g^\star}\left(\lambda^k - \gamma^k A \nabla f^\star(A^\top \lambda^k)\right) \quad . \tag{2.10}$$

To be able to use iteration (2.10) to solve (2.1), it is necessary to assume that the function $f^\star(A^\top \lambda)$ is differentiable. The gradient $\nabla f^\star(A^\top \lambda^k)$ can be computed by making use of Theorem 1 and is given by

$$\begin{aligned} \nabla f^\star(A^\top \lambda^k) &= \arg\max_{z\in\mathbb{R}^n} \left\{ \langle z, -A^\top \lambda^k \rangle - f(z) \right\} \\ &= \arg\min_{z\in\mathbb{R}^n} \left\{ f(z) + \langle A^\top \lambda^k, z \rangle \right\} \quad . \end{aligned}$$

Differentiability of $f^\star(A^\top \lambda)$ can be ensured if the minimum of the above problem is uniquely attained for all $\lambda \in \mathbb{R}^m$. For this to happen we need $f(z)$ to be strongly convex with constant $\sigma_f > 0$.

We have, therefore, the Dual Proximal Gradient Method (DPGM) algorithm for a fixed stepsize $\gamma^k = \gamma$:

Algorithm 1 Dual Proximal Gradient Method

Require: Initialize $\gamma < \frac{2\sigma_f}{\|L_f\|^2}$.
loop
1: $z^{k+1} = \arg\min_{z\in\mathbb{R}^n} \left\{ f(z) + \langle A^\top \lambda^k, z\rangle \right\}$
2: $\lambda^{k+1} = \mathbf{prox}_{\gamma g^\star}\left(\lambda^k + \gamma A z^{k+1}\right)$
end loop

We will subsequently see the connection of DPGM with the Alternating Minimization Algorithm (AMA) of Tseng [Tse91]. In order to proceed we state below the Moreau identity, a useful lemma that associates a convex function with its conjugate and allows the computation of the proximal operator of a conjugate function when one knows the proximal operator of the original one.

Lemma 1. *Let $f \in \Gamma_0(\mathbb{R}^n)$. Then for any $z \in \mathbb{R}^n$*

$$\mathbf{prox}_{\gamma f^\star}(z) + \gamma\, \mathbf{prox}_{f/\gamma}(z/\gamma) = z, \;\; \forall\, 0 < \gamma < +\infty \;\; .$$

We denote $\mu^k = \lambda^k + \gamma A z^{k+1}$ and Lemma 1 suggests that

$$\lambda^{k+1} = \mathbf{prox}_{\gamma g^\star}\left(\mu^k\right) = \mu^k - \gamma\, \mathbf{prox}_{g/\gamma}(\mu^k/\gamma) \;\; .$$

By introducing the variable $y^{k+1} = \mathbf{prox}_{g/\gamma}(\mu^k/\gamma) = \mathbf{prox}_{g/\gamma}(Az^{k+1} + \lambda^k/\gamma)$, we conclude with the AMA:

Algorithm 2 Alternating Minimization Algorithm (AMA)

Require: Initialize $\lambda_0 \in \mathbb{R}^m$, and $0 < \gamma < \frac{2\sigma_f}{\|L_f\|^2}$
loop
1: $z^{k+1} = \arg\min_{z\in\mathbb{R}^n} \left\{ f(z) + \langle A^\top \lambda^k, z\rangle \right\}$
2: $y^{k+1} = \mathbf{prox}_{\frac{1}{\gamma} g}\left(Az^{k+1} + \lambda^k/\gamma\right)$
3: $\lambda^{k+1} = \lambda^k + \gamma(Az^{k+1} - y^{k+1})$
end loop

2.2.5 Douglas-Rachford Splitting and Alternating Direction Method of Multipliers

Let us consider again the dual problem (2.4) and denote $h(\lambda) := f^\star(A^\top \lambda)$. This time we make no extra assumptions on f, *i.e.,* $f \in \Gamma_0(\mathbb{R}^n)$, $g \in \Gamma_0(\mathbb{R}^m)$.

The Douglas-Rachford Splitting (DRS) scheme is constituted of the three iterations:

$$v^{k+1} = \mathbf{prox}_{\gamma h}\left(\lambda^k - w^k\right) \tag{2.11}$$

$$\lambda^{k+1} = \mathbf{prox}_{\gamma g^\star}\left(v^{k+1} + w^k\right) \tag{2.12}$$

$$w^{k+1} = w^k + v^{k+1} - \lambda^{k+1} \ . \tag{2.13}$$

Note that the function h need not be differentiable in this case. It is shown in, *e.g.,* the lecture notes [Van10], that ADMM can be derived from DRS. Since the derivation is not straightforward, we give it in Appendix A for the sake of completeness. We state below the ADMM iterations.

Algorithm 3 Alternating Direction Method of Multipliers (ADMM)

Require: Initialize $y_0 \in \mathbb{R}^m$, $\lambda_0 \in \mathbb{R}^m$, and $\gamma > 0$

loop

1: $z^{k+1} = \underset{z\in\mathbb{R}^n}{\arg\min}\left\{f(z) + \langle\lambda^k, Az\rangle + (\gamma/2)\|Az - y^k\|^2\right\}$

2: $y^{k+1} = \mathbf{prox}_{\frac{1}{\gamma}g}\left(Az^{k+1} + \lambda^k/\gamma\right)$

3: $\lambda^{k+1} = \lambda^k + \gamma(Az^{k+1} - y^{k+1})$

end loop

Note that the method has no restrictions on the parameter γ. It is often chosen in practical applications as the most reliable among the algorithms described.

2.2.6 Convergence Rates and Inertial Acceleration

The decomposition methods that are discussed above make use of first-order information about the functions involved in the optimization problems, namely they compute the iterates based on gradients and subgradients. Slow convergence is usually the case with first-order methods, and the algorithms presented above are no exception to this rule. In this aspect, the algorithms presented here cannot achieve a rate better than the existing *worst case* lower complexity bounds for first-order methods [NY83], [Nes04b][2]

In the chapters that follow we frequently refer to convergence *in function values* and *in sequence values*. By saying that 'we have $O(1/k^q)$, $q \in (0, 2]$ global rate of convergence in *function values* for some function f', we mean that

$$\lim_{k\to+\infty} k^q(f(z^k) - p^*) \leq M \ ,$$

where p^* is the optimal value of f and $M > 0$. Accordingly, 'global $O(1/k^q)$ convergence rate *of a sequence* $\{z^k\}$' means that

$$\lim_{k\to+\infty} k^q(\|z^k - z^*\|) \leq M \ ,$$

[2]Nesterov's results point to the existence of a problem for which the algorithms cannot converge at a rate faster than the one presented in the aforementioned works. It is often (and hopefully) the case that for specific problem instances the algorithms behave much better than the lower complexity bounds.

where z^* is an optimizer and $M > 0$.

Accordingly, we refer to convergence rate of $o(1/k^q)$ for a sequence $\{z^k\}$ (or in the function values of f) when

$$\lim_{k\to+\infty} k^q(\|z^k - z^*\|) = 0, \quad \lim_{k\to+\infty} k^q(f(z^k) - p^*) = 0 \ ,$$

respectively.

Intuitively, big-O says that 'the sequence in the parenthesis (function values or iterates values) *can decay no slower* than the sequence $\{1/k^q\}$', while the little-o convergence rate means that 'the sequence in the parenthesis *will decay strictly faster* than $\{1/k^q\}$', hence is a stronger statement.

First order methods typically come with *sublinear convergence rates*, both in function values and in terms of the iterates. These rates are given from $q \in (0, 2]$ as presented in the previous section, depending on the structure of the problem at hand. Very roughly speaking, the following hold:

1. When *no structural assumptions on the functions are made*, sublinear convergence rates typically amount to a $O(1/k)$ global rate in function values.

2. Under the assumption of *differentiability*, several acceleration techniques can be applied, resulting in a $O(1/k^2)$ global rate in function values.

3. The convergence rate of the sequences of variables can usually be recovered as $O(1/\sqrt{k^q})$ from the function value's convergence rate.

4. Under the assumption of *differentiability and strong convexity*, linear convergence rates of the form $O(\omega^k)$, $\omega \in (0, 1)$ can be recovered.

Until very recently, almost all the convergence rate results regarding splitting methods, when no extra structural properties hold, were of big-O complexity. This landscape changed significantly with the works [DY16; Dav15b], where a faster rate of little-o complexity is proven to hold for the majority of the known splitting schemes.

More specifically, ADMM converges at a global rate of $o(1/k)$ if function values [Dav15b] and at $O(1/\sqrt{k})$ in the sequences of primal and dual variables, for an arbitrarily large parameter γ [ST14]. Since AMA (Algorithm 2) is equivalent to PGM applied to the dual problem, an $O(1/k)$ convergence rate in the dual function values is proven in [BT09, Theorem 3.1], enhanced to $o(1/k)$ in [Dav15b].

By making further assumptions on the function's structure, faster convergence rates can be recovered. In his work [Pol64], Polyak proposed a way to speed up the gradient method, namely to use the modified update

$$\begin{aligned} \hat{z}^k &= z^k + \beta^k(z^k - z^{k-1}) \\ z^{k+1} &= \hat{z}^k - \gamma^k \nabla f(z^k) \ , \end{aligned}$$

on the differentiable convex function f, with $\beta^k \in [0, 1)$. This method is commonly known as the *heavy ball method* and has guaranteed convergence under a Lipschitz continuity assumption on ∇f. In this way the next iterate depends on the last gradient update and the previous step $z^k - z^{k-1}$, which is called a *momentum sequence*. Polyak's motivation came from the observation

that the gradient descent iteration can be seen as a discretization of the ODE $\ddot{z} = -\nabla f(z)$, the iterates of which tend to 'zig-zag' in directions that do not point straight towards the minimizer. Adding 'inertia' or 'damping' to the iteration should help to keep the method on track towards the solution, thus he proposed to take the ODE $\gamma\ddot{z} + \dot{z} = -\nabla f(z),\ \gamma > 0$, leading to his heavy ball method [Lor15]. After discretization, the new iterate is udpated as a linear combination of the two previous iterates. This seemingly small change greatly improves the performance of the original gradient scheme.[3]

The method has been significantly improved in order to tackle more general problems, leading to the appearance of projected, proximal [OBP15] as well as incremental [GOP17] variants. In addition, its convergence rate has been studied and analyzed. More specifically, the method can be shown to converge linearly [Pol87] when the function f is both differentiable with Lipschitz continuous gradient and strongly convex.

In his seminal paper [Nes83], Nesterov modified the heavy ball method by simply evaluating the gradient at the extrapolated point $\hat{z}^k$ instead of z^k. In addition, he proposed a special formula for computing the relaxation sequence $\{\beta^k\}$, resulting in an optimal convergence rate for the scheme. The simple update formula is:

$$\begin{aligned} \beta^k &= \left(1 + \sqrt{4(\beta^{k-1})^2 + 1}\right)/2 \\ \hat{z}^k &= z^k + \frac{\beta^{k-1} - 1}{\beta^k}(z^k - z^{k-1}) \\ z^{k+1} &= \hat{z}^k - \gamma^k \nabla f(\hat{z}^k)\ , \end{aligned} \tag{2.14}$$

with $\beta^0 = 1$. Nesterov's scheme admits a similar ODE interpretation to that of Polyak's [SBC14]. Subsequently, Güler extended Nesterov's results for the PPA iteration (2.6) [Gül92], while Beck and Teboulle extended it for the PGM [BT09]. Tseng [Tse08] unified the analysis of fast PGM and proposed a condition for the acceleration sequence under which convergence is ensured. More specifically, the sequence $\{\beta^k\}$ needs to satisfy

$$\frac{1 - \beta^{k+1}}{(\beta^{k+1})^2} \leq \frac{1}{(\beta^k)^2}\ .$$

Application of such schemes results in an $O(1/k^2)$ global rate of convergence in function values; a rate that is *optimal for first-order methods involving a differentiable and a nonsmooth function.* Convergence in terms of the sequence was not proven until recently [CD15], and the derivation of a rate is still open (unless further assumptions on the structure of the function are made). Apart from the theoretical results, the scheme has been observed to practically accelerate convergence in numerous problem instances. In addition, the extra computational cost is insignificant. Finally, a better convergence rate of order $o(1/k^2)$ for variants of Nesterov's method has been proven in [AP15].

[3] The blog [Goh17] provides a nice graphical and intuitive explanation of the mechanism behind the momentum sequence.

2.3 Monotone Operators

We discussed in Section 2.2 how the proximal operator can help us generate a variety of iterative approximation schemes that find a minimizer of a (sum of) convex function(s). Except for the use of the proximal operator, these schemes are seemingly disparate. We will see in this section that the schemes presented above (and many more) can be put under a common framework if viewed through the lens of *monotone operators*. Casting the iterative schemes we have seen above in this framework simplifies signficantly the convergence analysis when it comes to devising new schemes, as will see in Chapter 4 of this work. The rough idea behind this modeling approach is that the problem of finding a minimizer of a convex program can be cast as an equivalent problem, namely that of finding a zero of a monotone operator. This problem is in turn transformed into finding a fixed point of a function related to the monotone operator. Finally, the fixed point is found by the fixed-point iteration, yielding an algorithm for the original problem.

Our discussion follows the style of [RB16], which gives a comprehensive, yet simple analysis on the topic of monotone operators. A more in-depth treatment can be found in the book [BC11]. We choose to perform the analysis on a Hilbert space setting since (i) it is more general and encompasses the Euclidean case and (ii) will be useful for the upcoming analysis in Chapter 4.

2.3.1 Basics on Monotone Operators

A *relation* or *operator* T in $\mathcal{H}$ is a subset of $\mathcal{H} \times \mathcal{H}$, where $\mathcal{H}$ is a Hilbert space. We write $u = T(z) = Tz$ to denote the set $\{u \mid (z, u) \in T\}$. The relation T has *Lipschitz constant* L if for all $u \in T(z)$ and $v \in T(x)$ it holds that $\|u - v\| \leq L\|z - x\|$. If $L < 1$ we call T a contraction, while if $L \leq 1$, then T is called *nonexpansive*. A point z is a *fixed point* for T if $z = Tz$, denoted as $z \in \operatorname{fix} T$. This is equivalent to $z \in \operatorname{zer} S$, *i.e.,* $0 \in \operatorname{zer} Sz$, where $S = I - T$ and I is the identity operator such that $I = \{(z, z) \mid z \in \mathcal{H}\}$. The set of fixed points of a nonexpansive operator with full domain ($\operatorname{dom} T = \mathcal{H}$) is closed and convex. The operator is called *averaged* if $T = (1 - \theta)I + \theta G$ with $\theta \in (0, 1)$, where G is a nonexpansive operator. It follows that T is nonexpansive and has the same fixed points as G. Compositions of nonexpansive operators are nonexpansive, as well as compositions of averaged operators result in averaged operators.

Monotonicity is another important property of a relation. T is *monotone* if $\langle z - x, Tz - Tx\rangle \geq 0$ for all $z, x \in \mathcal{H}$. The relation is *maximal monotone* if there is no monotone operator that properly contains it.

Two operators that play a crucial role in finding a zero of a relation are the *resolvent* of T, defined as $J_T = (I + T)^{-1}$, and the *reflection* of T defined as $R_T = 2J_T - I$. The following properties hold:

- If T is monotone, then J_T and R_T are nonexpansive functions.
- If T is maximal monotone, then J_T and R_T have full domain $\mathcal{H}$.
- T shares the same fixed points with J_T and R_T.

In convex optimization problems, we are interested in specific operators, the most important of which is the *subdifferential* ∂f of a convex closed and proper function $f : \mathcal{H} \mapsto \mathbb{R} \cup \{+\infty\}$. If $f \in \Gamma_0(\mathcal{H})$, then ∂f is maximal monotone.

The subdifferential is important because the solution to a convex optimization problem can be cast as finding a zero of the subdifferential, *i.e.,* $0 \in \partial f(z)$. This can be equivalently written as $z \in (I + \partial f)(z) \Leftrightarrow z \in (I + \partial f)^{-1}(z) \Leftrightarrow z = J_{\partial f}(z)$. Hence the set of optimizers coincides with the fixed points of the resolvent and the reflection of the subdifferential. These operators take special forms in the framework of convex optimization. More specifically, the proximal operator introduced in (2.5) is the resolvent of the subdifferential of f evaluated at x, *i.e.,* $\mathbf{prox}_{\gamma f}(x) = (I+\gamma\partial f)^{-1}(x) = \arg\min\left\{f(z) + (1/(2\gamma))\|z - x\|^2\right\}$, $\gamma > 0$. Accordingly, the reflection operator is denoted as $\mathbf{refl}_{\gamma f} : \overset{z}{\mathcal{H}} \to \mathcal{H}$ and is defined as $\mathbf{refl}_{\gamma f} = 2\,\mathbf{prox}_{\gamma f} - I$.

The algorithms discussed in Section 2.2 can be derived as fixed-point iterations of functions associated to relevant monotone operators. The full derivations can be found in [RB16].

PPA Consider the function $f \in \Gamma_0(\mathcal{H})$ and the subdifferential operator $A = \partial f$. The proximal operator is a *function* of A defined above as $T = J_{\gamma A} = \mathbf{prox}_{\gamma f}$. The latter gives us access to the fixed-point iteration $z^{k+1} = Tz^k$, which is the PPA for $\gamma^k = \gamma > 0$, and converges to a fixed point z^*.

PGM Consider the problem of solving the inclusion $0 \in (A+B)(z)$ and assume that B is single-valued. By manipulating slightly the inclusion, one can derive the fixed-point iteration $z^{k+1} = Tz^k$, where $T = T_A T_B$ and $T_A = J_{\gamma A}$, $T_B = I - \gamma B$ for γ lying inside a specified range of values. This scheme is known as the *forward-backward splitting (FBS)* iteration [Pas77].

Let us now consider the minimization of $f(z) + g(z)$, $f \in \Gamma_0(\mathbb{R}^n)$ differentiable with Lipschitz continuous gradient L_f and $g \in \Gamma_0(\mathbb{R}^n)$. The problem can be cast as the inclusion $0 \in (A + B)(z)$, where $A = \partial g$ and $B = \nabla f$. Applying the FBS, the resulting iteration is the PGM as given in (2.9) for $\gamma^k = \gamma \in (0, 2/L_f)$, and converges to a fixed point z^*.

DRS Consider the same problem of minimizing $f(z) + g(z)$, with $f \in \Gamma_0(\mathbb{R}^n)$ and $g \in \Gamma_0(\mathbb{R}^n)$. Casting the problem as the inclusion $0 \in (A+B)(z)$, where $A = \partial g$ and $B = \nabla f$, we can introduce the operator $T = \frac{1}{2}(T_A T_B + I)$ by introducing the functions $T_A = R_{\gamma A} = \mathbf{refl}_{\gamma f}$ and $T_B = R_{\gamma B} = \mathbf{refl}_{\gamma g}$. The resulting iteration is the DRS algorithm as given in equations (2.11),(2.12),(2.13) for $\gamma^k = \gamma > 0$, and converges to a fixed point z^*.

As previously mentioned, viewing the iterative schemes via the lens of monotone operators allows for some generalizations. Along these lines, we introduce the following definitions:

Definition 7 (Cocoercivity.). *The operator A is δ-cocoercive with $\delta > 0$ if*

$$\langle x - y, Ax - Ay\rangle \geq \delta\|Ax - Ay\|^2,\ \forall x, y \in \mathcal{H}.$$

Definition 8 ((Quasi-)Strong monotonicity.). *The operator A is μ-strongly monotone with $\mu > 0$ if*

$$\langle x - y, Ax - Ay\rangle \geq \mu\|x - y\|^2,\ \forall x, y \in \mathcal{H}.$$

If the inequality holds only for $y \in \operatorname{zer} A$, i.e.,

$$\langle x - y, Ax\rangle \geq \mu\|x - y\|^2,\ \forall x \in \mathcal{H},$$

then the operator is quasi-μ-strongly monotone.

For a differentiable convex function f, cocoercivity of $A = \nabla f$ implies Lipschitz continuity of ∇f with constant δ. Similarly, strong monotonicity for $A = \partial f$ implies strong convexity of f with constant μ.

2.3.2 The Krasnosel'skiĭ-Mann Iteration

Instead of computing the next iterate z^{k+1} by applying the operator T to z^k, the fixed-point iterations discussed above can be relaxed by considering a point on the line segment that connects z^k and Tz^k. The Krasnosel'skiĭ-Mann (KM) iteration is probably the most popular *interpolated fixed-point iteration* and takes the form

$$z^{k+1} = z^k + \eta^k(Tz^k - z^k) \ , \tag{2.15}$$

where $T : \mathcal{H} \mapsto \mathcal{H}$ is a nonexpansive operator and $\eta^k \in [0, 1]$, $\sum_{k\in\mathbb{N}} \eta^k(1 - \eta^k) = +\infty$ is a relaxation constant. The KM iteration converges to a solution if one exists [Kra55; Man53], [BC11, Theorem 5.14].

Since the associated operators T of the three methods discussed in the previous section are nonexpansive by construction (modulo a pertinent choice of the parameter γ for FBS), a KM iteration results in convergent iterative schemes. More specifically we have that:

- PPA is a KM iteration with $T = J_{\gamma A}$ that converges for $\eta^k \in (0, 2)$ (the iteration can be over-relaxed and convergence is maintained) [Ber15, Proposition 5.1.10].
- FBS is a KM iteration with $T = J_{\gamma A}(I - \gamma B)$ that converges for $\eta^k \in [0, \delta]$, where $\delta = \min\{1, L/\gamma\} + 1/2$, provided that B is L-cocoercive and $\gamma \in (0, 2/L)$ [BC11, Theorem 25.8].
- DRS is a KM iteration with $T = R_{\gamma A}R_{\gamma B}$ and $\eta^k = 1/2$ that converges for any $\gamma > 0$.

2.4 Model Predictive Control

Throughout this work we consider linear discrete-time systems of the form

$$x_{i+1} = A_i x_i + B_i u_i \ , \tag{2.16}$$

where $x_i \in \mathbb{R}^n$ is the state at the current time step, $u_i \in \mathbb{R}^m$ is the current input, while the matrices $A_i \in \mathbb{R}^{n\times n}, B_i \in \mathbb{R}^{n\times m}$ are typically derived from either physical modeling or/and by employing system identification algorithms. For the linear time-invariant (LTI) case we may simplify the notation to

$$x^+ = Ax + Bu \ ,$$

where the successor state is now denoted by x^+. Note that here $x \in \mathbb{R}^n$.

The results that appear in this section can be found in classical (model predictive) control texts (see, *e.g.,* [RM09]). We borrow here a few definitions and theorems essential for the comprehension of this thesis from [Jon12] and [Dom13].

2.4.1 Invariance and Lyapunov Stability

If system (2.16) is controlled by the control law $u_i = \kappa(x_i)$, the closed-loop system is given by

$$x_{i+1} = Ax_i + B\kappa(x_i) = f(x_i) \ . \tag{2.17}$$

Definition 9 (Positively invariant set). *A set $\mathcal{C} \subseteq \mathbb{R}^n$ is said to be a positively invariant set for the autonomous system* (2.17) *if*

$$x_i \in \mathcal{C} \Rightarrow x_{i+1} \in \mathcal{C}, \ i \in \mathbb{N} \ .$$

Definition 10 (Maximal positively invariant set). *Consider a constraint set $\mathcal{X}$ for the states of system* (2.17)*. The set $\mathcal{C}_\infty \subset \mathcal{X}$ is a maximal invariant set with respect to $\mathcal{X}$ if $0 \in \mathcal{C}_\infty$, $\mathcal{C}_\infty$ is invariant and $\mathcal{C}_\infty$ contains all invariant sets that contain the origin.*

Definition 11 (K-class function). *A real-valued function $\alpha : \mathbb{R}_+ \mapsto \mathbb{R}_+$ belongs to class K if it is continuous, strictly increasing and $\alpha(0) = 0$.*

Definition 12 (Lyapunov function). *Let $\mathcal{C}$ be a positively invariant set for system* (2.17) *containing a neighborhood of the origin $\mathcal{N}$ in its interior and let $\underline{\alpha}$, $\overline{\alpha}$ and β be K-class functions. A nonnegative function $V : \mathbb{R}^n \mapsto \mathbb{R}_+$ with $V(0) = 0$ is called a (asymptotic) Lyapunov function in $\mathcal{C}$ if:*

- $V(x_i) \geq \underline{\alpha}(\|x_i\|) \quad \forall x_i \in \mathcal{C}$
- $V(x_i) \leq \overline{\alpha}(\|x_i\|) \quad \forall x_i \in \mathcal{N}$
- $V(f(x_i)) - V(x_i) \leq -\beta(\|x_i\|) \quad \forall x_i \in \mathcal{C} \ .$

Theorem 2 (Global Lyapunov Stability). *If a system admits a (asymptotic) Lyapunov function, then the equilibrium point at the origin is asymptotically stable.*

2.4.2 Linear Quadratic Regulator

Consider the optimal control problem

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\sum_{i=0}^{\infty} x_i^\top Q x_i + u_i^\top R u_i \\ \text{subject to} & x_{i+1} = Ax_i + Bu_i, \quad i \in \mathbb{N} \\ & x_0 = x_{\text{init}} \ , \end{array} \tag{2.18}$$

with variables $u = (u_0, \ldots)$, while $Q = Q^T$ and $Q \in \mathbb{R}^n_+$, $R = R^T$ and $R \in \mathbb{R}^n_{++}$. Problem (2.18) is referred to as the *(Infinite Horizon) Linear Quadratic Regulator (LQR)* problem, and it has a closed form solution given by Kalman in [Kal60]. More specifically, the *optimal control policy* that solves (2.18) has the form of a *linear state feedback* given by $u = Kx$, where

$$K = -(R + B^\top P B)^{-1} B^\top P A \tag{2.19}$$

and P is the solution to the *Discrete Algebraic Riccati Equation (DARE)*

$$P = Q + A^\top PA - A^\top PB(R + B^\top PB)^{-1}B^\top PA \ . \tag{2.20}$$

The infinite horizon controller is stable, as the following theorem suggests.

Theorem 3 (Stability of the LQR controller). *Assume that $Q \succeq 0$, $R \succ 0$, (A, B) is stabilizable and (Q, A) is detectable. There exists a unique positive semidefinite solution P to the DARE (2.20) and the closed-loop system matrix $A - B(R + B^\top PB)^{-1}B^\top PA$ is stable.*

2.4.3 Linear Model Predictive Control

A linear MPC problem is an optimal control problem of the form

$$\begin{aligned} V^*(x_{\text{init}}) = \ & \min_{u} \ \tfrac{1}{2}\sum_{i=0}^{T-1} l(x_i, u_i) \\ & \text{s.t.} \ \ x_{i+1} = Ax_i + Bu_i, \quad i = 1, \dots, T-1 \\ & \qquad x_0 = x_{\text{init}} \\ & \qquad (x_i, u_i) \in \mathcal{X} \times \mathcal{U}, \quad i = 1, \dots, T-1 \ . \end{aligned} \tag{2.21}$$

The optimization is performed over the input variables $u = (u_0, \dots, u_{T-1}) \in \mathbb{R}^{mT}$, while the sets $\mathcal{X}$ and $\mathcal{U}$ are closed and convex. The stage cost function is strictly nonnegative, *i.e.,* $l : \mathbb{R}^n \times \mathbb{R}^m \mapsto \mathbb{R}_{++}$ and $l(0, 0) = 0$. The resulting controller is applied in a *receding-horizon fashion*, *i.e.,* only the first input $u_0^*(x_{\text{init}})$ is applied to the system, and the resulting state $x^+ = Ax_{\text{init}} + Bu_0^*(x_{\text{init}})$ is used as the initial state $x_0 = x^+$ when problem (2.21) is resolved. We define the *feasible set* for problem (2.21), as follows:

Definition 13 (Feasible set). *The feasible set $\mathcal{X}_T$ is defined as the set of initial states $x_{\text{init}} \in \mathbb{R}^n$ for which the MPC problem (2.21) with horizon T is feasible, i.e.,*

$$\mathcal{X}_T := \{x_{\text{init}} \mid \exists (u_0, \dots, u_{T-1}) \text{ s.t. } (x_i, u_i) \in \mathcal{X} \times \mathcal{U}, \ i = 1, \dots, T\} \ .$$

We desire two properties when it comes to deriving an optimization-based controller from (2.21): *Stability* and *recursive feasibility*. The definition of the latter is given below.

Definition 14 (Recursive feasibility). *The MPC problem (2.21) is called recursively feasible if for all states residing in the feasible set described in Definition 13, constraint satisfaction is guaranteed at every state along the closed-loop trajectory.*

Ideally, we would like to solve (2.21) for an infinite horizon, since stability and recursive feasibility are then guaranteed. This problem is rather challenging due to the infinite number of decision variables involved. A way to approximate it is to choose a horizon T that is long enough so that no constraints are active from the state x_T onwards, and find a local controller that, starting from x_T will regulate the state sequence to the origin and whose infinite-horizon cost can be expressed in closed form. However, choosing the right T a priori is not trivial. We employ, instead, a terminal (positively invariant) constraint set $\mathcal{X}_f \subseteq \mathcal{X}$, such that $x_T \in \mathcal{X}_f$, to enforce recursive constraint

satisfaction and a terminal cost V_f to 'fake' the tail of the infinite-horizon problem. The full MPC scheme with stability and recursive feasibility guarantees becomes:

$$\begin{aligned} V^*(x_{\text{init}}) = \quad & \min_{u} \quad \tfrac{1}{2}\sum_{i=0}^{T-1} l(x_i, u_i) + V_f(x_T) \\ & \text{s.t.} \quad x_{i+1} = Ax_i + Bu_i, \quad i = 1, \dots, T-1 \\ & \qquad x_0 = x_{\text{init}} \\ & \qquad (x_i, u_i) \in \mathcal{X} \times \mathcal{U}, \quad i = 1, \dots, T-1 \\ & \qquad x_T \in \mathcal{X}_f \ . \end{aligned} \tag{2.22}$$

The terminal cost function is positive definite, *i.e.,* $V_f : \mathbb{R}^n \mapsto \mathbb{R}_{++}$. The terminal set $\mathcal{X}_f \subseteq \mathcal{X}$ is convex compact and contains the origin in its interior. The following result holds:

Theorem 4 (Stability and Recursive Feasibility for Linear MPC). *The following assumptions hold true:*

1. *The stage cost l is a positive definite function.*
2. *The terminal set is invariant under the local control law $\kappa_f(x_i)$:*

$$x_{i+1} = Ax_i + B\kappa_f(x_i) \in \mathcal{X}_f \quad \forall x_i \in \mathcal{X}_f \ .$$

   All state and input constraints are satisfied in $\mathcal{X}_f$:

$$\mathcal{X}_f \subseteq \mathcal{X}, \ \kappa_f(x_i) \in \mathcal{U} \quad \forall x_i \in \mathcal{X}_f \ .$$

3. *The terminal cost is a continuous Lyapunov function in the terminal set $\mathcal{X}_f$:*

$$V_f(x_{i+1}) - V_f(x_i) \leq -l(x_i, \kappa_f(x_i)) \quad \forall x_i \in \mathcal{X}_f \ .$$

Then the closed-loop system under the MPC control law $u_0^*(x_{\text{init}})$ is stable and the system $x^+ = Ax_{\text{init}} + Bu_0^*(x_{\text{init}})$ is invariant in the feasible set $\mathcal{X}_T$.

Theorem 4 roughly states that the infinite horizon problem (2.21) (where $T = \infty$) can be approximated by forcing the final state to be in an invariant set for which there exists an invariance-inducing controller. In addition, the infinite-horizon cost of the system operating under this controller should be expressed in closed form. In other words, an approximate solution to problem (2.21) with $T = \infty$ can be acquired by properly choosing $V_f, \mathcal{X}_f$ and κ_f and solving (2.22).

2.5 Distributed Ancillary Service Provision with Controllable Buildings

We developed the distributed optimization algorithms of Chapters 4 and 5 of this thesis with a specific setup in mind, a topology with a global node, usually called the *coordinator* and several local nodes, *the agents*. The motivation behind this setting is drawn from modern electricity market setups, where the provision of services flows bidirectionally, namely the loads can also participate in

relieving the grid in several situations. Loads that participate in such electricity markets are called *prosumer loads*.

The growing interest in turning once passive loads into active prosumers, along with the introduction of dispatchable energy storage elements in the grid, is motivated by a rapid and significant increase of renewable production into the generation mix. Renewable energy sources are inherently uncertain and volatile and they pose new challenges to the classic control paradigm of the power grid, but they also raise new opportunities for alternative use of existing resources, development of new technologies and novel market designs [EC10; Nurb; Nura]. At the same time, the introduction of demand side management via large loads as, *e.g.,* commercial buildings, is inherently designed in a distributed fashion. The services are usually provided through aggregators of flexible loads [Abr17] and several research works propose peer-to-peer implementations. In both cases, the design calls for distributed (or decentralized) implementations.

(Commercial) buildings can be excellent controllable loads by manipulating their thermal flexibility. Overcoming the obstacle of modeling can be challenging and often costly for real-world implementations, however several modeling tools have been developed over the last years. A magnitude of studies has shown both financial and non-financial benefits in offering ancillary services with commercial buildings [SGMS16; FGQBLJ17; GFQJ17; VKMAC17a; VKMAC17b].

In the chapters of this thesis we are going to consider two problems with rather similar nature, *i.e.,* in both cases we have a population of buildings (and potentially an electric battery) trying to track some signal transmitted by the network operator, while they simultaneously seek to minimize some private cost while operating inside their feasible regions (in terms of comfort and actuation constraints). We briefly present the two problems below.

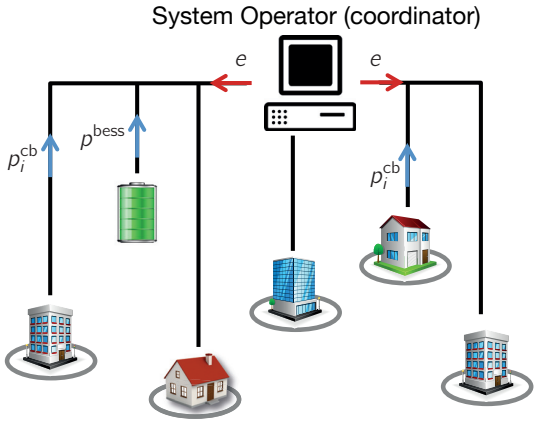


Figure 2.2: Sketch of a network of buildings with a battery energy storage system (agents) and a global system operator (coordinator). The agents communicate their individual consumption vectors (p_i^{cb}, p^{bess}) and the coordinator uses them in order to form an incentive signal (e) that will be in turn communicated to them.

Load sharing This is the problem of cooperative tracking of a reference signal from a population of N controllable resources (agents). These problems typically arise in the context of microgrids, where a mixture of energy generation, energy storage elements and loads are coupled together in order to satisfy a predicted power demand profile, as is the case in day ahead electricity markets. The *sharing problem* [BPCPE11, Section 7.2] takes the general form

$$\text{minimize} \quad \sum_{t=0}^{T-1}(\sum_{i=1}^{N} z_i(t) - r(t))^2 + \sum_{i=1}^{N} f_i(z_i, \cdot) \ . \tag{2.23}$$

The variable $z_i(t)$ refers to the total consumption of the i^{th} agent at time instant t, where $i = 1, \ldots, N$, and $z_i = (z_i(0), \ldots, z_i(T-1)) \in \mathbb{R}^T$, $z = (z_1, \ldots, z_N) \in \mathbb{R}^{NT}$. The reference power profile is denoted by $r(t)$, while time spans from $t = 0, \ldots, T-1$, *i.e.,* we have a T-timesteps ahead prediction of the power profile. The first term, $h(z) = \sum_{t=0}^{T-1}(\sum_{i=1}^{N} z_i(t) - r(t))^2$, penalizes the deviation of the total power contribution to the reference power profile. The individual components $f_i(z_i, \cdot)$ are local performance criteria coupled with implicit descriptions of convex sets, constructed by the intersection of linear equations (agent dynamics) and constraints, details that are hidden from the global node. Additional variables that are private to the agents might appear in the f_i's (hence the dependence on other variables except for z_i).

Dispatchable distribution feeder Another problem, first introduced in [SNCP16], is that of the so-called dispatchability of distribution feeders where the main target is to achieve virtually perfect dispatchability of a set of devices consisting of uncontrollable loads and distributed generation. The idea comprises two stages, namely a day-ahead prediction of a dispatch plan and real-time operation, broken further into a high-level and a faster low-level controller, both model-based. Our interest is to solve the high-level MPC problem that takes the abstract form

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i(z_i, \cdot) \\ \text{subject to} \quad & \sum_{i=1}^{N} z_i = r \ . \end{aligned} \tag{2.24}$$

The variable $z_i(t)$ refers to the total consumption as before, but in this case a hard dispatchability constraint is enforced. In [FGNSPJ17], the adequacy of the involved agents to offer the reference r is ensured by the addition of a sized grid-connected battery energy storage system in the mix.

In all simulations of the dissertation, the setups comprise small, medium and large office buildings, generated by the OpenBuild software [GQJ15]. The buildings are described as linear dynamical systems, the input to which is the thermal heat (kW) that is entering or leaving each zone, while the output is the temperature at each zone ($^\circ C$). The energy conversion systems (electrical to thermal) are modeled as a static map, which is represented by a constant coefficient of performance (COP). The buildings can participate in the ancillary service market by *increasing or decreasing their consumption with respect to some baseline power profile.* An individual building seeks to contribute to the tracking objective while respecting temperature as well as operational constraints. The exact

form of the functions that appear in (2.23) and (2.24) are to be specified in the corresponding chapters.

An instance of a local optimization problem for building i is

$$f_i^{\text{cb}}(p_i^{\text{cb}}, u_i, x_i, y_i) := \left\{ \text{Cost}_i \;\middle|\; (p_i^{\text{cb}}, u_i, x_i, y_i) \in \mathcal{C}_i^{\text{cb}} \right\} , \tag{2.25}$$

$$\mathcal{C}_i^{\text{cb}} = \left\{ \begin{array}{l} x_i(t+1) = A_i x_i(t) + B_{u,i} u_i(t) + B_{w,i} \hat{w}_i(t) \\ x_i(0) = x_i^{\text{init}} \\ y_i(t) = C_i x_i(t) \\ y_{\min,i}(t) \leq y_i(t) \leq y_{\max,i}(t) \\ u_{\min,i} \leq u_i(t) \leq u_{\max,i} \\ p_i^{\text{cb}}(t) = \sum_{j=1}^{N_i} u_{ij}(t) \end{array} \right\} ,$$

with $x_i \in \mathbb{R}^{n_i T}$, $u_i = \{u_{ij}\}_{j=1}^{m_i} \in \mathbb{R}^{m_i T}$, where $p_i^{\text{cb}}(t)$ is the total amount (electrical equivalent) of the thermal consumption at time t, and m_i is the number of zones of the building. Zone temperatures are described with the variables $y_i \in \mathbb{R}^{l_i T}$. In addition, the temperature constraints are relaxed outside working hours, hence the time varying constraint limits $y_{\min,i}(t), y_{\max,i}(t)$. Local objectives that the building might accommodate are described by Cost_i.

Whenever a battery is present, it is represented as single-state linear model, with the state being the state-of-charge (SOC) and the input being the active power denoted by $p^{\text{bess}}(t) \in \mathbb{R}$. The battery operates within capacity and power limits, while the purpose is to keep the SOC close to a reference value $SOC^{\text{ref}}(t) \in \mathbb{R}$.

$$f^{\text{bess}}(p^{\text{bess}}) := \left\{ \frac{1}{2} \sum_{t=1}^{T} \| SOC(t) - SOC^{\text{ref}}(t) \|_2^2 \;\middle|\; p^{\text{bess}} \in \mathcal{C}^{\text{bess}} \right\} , \tag{2.26}$$

$$\mathcal{C}^{\text{bess}} = \left\{ \begin{array}{l} SOC(t+1) = aSOC(t) + bp^{\text{bess}}(t) \\ SOC(0) = SOC^{\text{init}} \\ SOC_{\min} \leq SOC(t) \leq SOC_{\max} \\ p_{\min}^{\text{bess}} \leq p^{\text{bess}}(t) \leq p_{\max}^{\text{bess}} \end{array} \right\} .$$

More details are given in the corresponding chapters.

Chapter 3

Infinite-Horizon Constrained Linear Quadratic Regulator

3.1 Introduction

An important extension of the famous result of [Kal60] on the closed form solution of the infinite-horizon linear quadratic regulation (LQR) problem, as presented in Section 2.4, is the case where input and state variables are constrained. This problem is computationally significantly more difficult and has been by and large addressed only approximately. Model predictive control only approximates the infinite-time constrained problem by a finite-time one, whereas stability is then typically enforced by adding a suitable terminal constraint and a terminal penalty as discussed in Section 2.4. The inclusion of a terminal constraint limits the feasible region of the MPC controller, and, consequently, the region of attraction of the closed-loop system. In practical applications, this problem is typically overcome by simply choosing a 'sufficiently' long horizon based on process insight (*e.g.,* dominant time constant). Closed-loop behavior is then analyzed a posteriori, for instance by exhaustive simulation.

In this chapter, we apply the *(accelerated) version of the dual proximal gradient method ((A)DPGM) to the infinite-horizon CLQR problem in order to solve it.* The idea is to condense the problem and describe it in terms of its input variables, then formulate the dual of the corresponding (infinite-dimensional) *quadratic program* (QP), and apply the (A)DPGM method to solve it. More specifically, the method decomposes the QP into two subproblems, the first one being an infinite-dimensional least squares problem and the second one a simple clipping of an infinite sequence to the non-positive orthant. The subproblems are solved repeatedly (with the solution of one influencing the cost function of the other) until convergence to the solution of the original problem. This is in contrast to the approach of [SR98], which requires the solution of a sequence of *constrained* QPs. We show that both subproblems of the proposed algorithm can be solved tractably (which is not a priori obvious since we are working with infinite sequences), the first one by solving a finite-dimensional system of linear equations (with the possibility to pre-factorize the matrices) and the second one by simple clipping of finitely many real numbers on the non-positive real line. Convergence of the scheme (with rate $O(1/k^2)$ for function values and $O(1/k)$ for primal iterates) to the optimal infinite-horizon sequence is guaranteed under mild assumptions. Therefore

the proposed algorithmic scheme provides a means to compute the solution of the infinite-horizon constrained LQR problem with guaranteed convergence. Apart from the theoretical contribution, we provide a fully implementable method, competitive for real-time control. We (i) eliminate the need for knowledge of uncomputable quantities, (ii) propose computationally efficient ways to solve the optimization subproblems and (iii) propose a warm-starting scheme that performs well in practice.

The chapter is organized as follows: In Section 3.3 we introduce the problem of interest. In Section 3.4 the problem is expressed in terms of its dual variables by means of the proximal splitting framework. In Section 3.5 we present the accelerated dual proximal gradient algorithm to solve the problem and show that each iteration of the algorithm can be carried out tractably. The convergence proofs for this scheme are given in Section 3.6, while Section 3.7 discusses the computational aspects that make the algorithm practical to use. Section 3.8 presents two numerical examples: A toy example of an unstable system with two states and one control input illustrates the main features of the algorithm. Subsequently, we demonstrate the practical applicability of the algorithm on a linearized model of a quadcopter with 12 states, four inputs, and polytopic constraints. Finally, Appendices 3.10.1 and 3.10.2 provide the proofs for the results presented in Section 3.6.

3.2 Related work

There have been few results addressing directly the infinite-horizon CLQR problem. Among the most well-known efforts are the works [CM96], [SR98] and [GBTM04]. The authors of [CM96] suggest a scheme for offline computation of a sufficient horizon length. The solution of the corresponding QP is then equivalent to the original infinite dimensional problem. The reported results are somewhat conservative, while the offline part of the proposed algorithm can be computationally prohibitive since it involves the solution of a possibly nonconvex problem, the computation of a positively invariant set as well as a vertex enumeration problem. The authors of [SR98] extend the work of [SD87] by solving a sequence of QPs of finite horizon length, which is monotonically non-decreasing. After each QP has been solved, the inclusion of the final state in a positively invariant set associated to the optimal unconstrained LQ controller is checked; if the final state is not included in the set, the horizon was insufficient and has to be increased. Finally, the authors in [GBTM04] employ parametric quadratic programming and a reachability analysis approach to compute the least conservative horizon length that ensures optimal infinite horizon performance. Although this work provides the exact necessary horizon length for the feasible set of initial states, it suffers from tractability issues since it is based on state-space partitioning and thus can only be applied to small systems.

3.3 Problem statement

The constrained regulation problem for an LTI system can be written in the general form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\sum_{i=0}^{\infty} x_i^\top Q x_i + u_i^\top R u_i \\ \text{subject to} & x_{i+1} = Ax_i + Bu_i, \quad i \in \mathbb{N} \\ & x_0 = x_{\text{init}} \\ & C_x x_i \leq c_x \\ & C_u u_i \leq c_u \ , \end{array} \tag{3.1}$$

with variables $x_i \in \mathbb{R}^n$ and $u_i \in \mathbb{R}^m$, and data $c_x \in \mathbb{R}^{p_x}$ and $c_u \in \mathbb{R}^{p_u}$.

We make the following standing assumptions:

Assumption 1. *The pair (A, B) is stabilizable, the optimal value of problem (3.1) is finite, the set*

$$\mathcal{X} := \{x \in \mathbb{R}^n \mid C_x x \leq c_x\}$$

contains the origin in its interior, the matrix C_u has full column rank, the matrix Q is positive semidefinite and R is positive definite.

Remark 2 (Stability). *Clearly, under Assumption 1, if the problem (3.1) is feasible and the pair $(A, \sqrt{Q})$ detectable, then the control sequence optimal in (3.1) is stabilizing. Therefore, there is no need to enforce stability in an ad hoc way as is commonly done when the infinite-time problem (3.1) is approximated by a finite-time one solved in a receding-horizon fashion.*

From now on we write infinite sequences and infinite-dimensional operators in bold font.

The problem can be rewritten in the dense form, *i.e.,* by writing the states as functions of the inputs. This is done by defining the operators

$$\boldsymbol{A} = \begin{bmatrix} A \\ A^2 \\ \vdots \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} B & 0 & 0 & \cdots \\ AB & B & 0 & \cdots \\ A^2B & AB & B & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} ,$$

$$\boldsymbol{Q} = \operatorname{diag}(Q, Q, \ldots) \ , \quad \boldsymbol{R} = \operatorname{diag}(R, R, \ldots) \ ,$$

$$\boldsymbol{C_x} = \operatorname{diag}(C_x, C_x, \ldots) \ , \quad \boldsymbol{c_x} = (c_x, c_x, \ldots) \ ,$$

$$\boldsymbol{C}_i = \begin{bmatrix} \boldsymbol{e}_i \otimes C_u \\ [\boldsymbol{C_x B}]_i \end{bmatrix}, \quad c_i = \begin{bmatrix} c_u \\ c_x - C_x A^i x_{\text{init}} \end{bmatrix} ,$$

$$\boldsymbol{H} = \boldsymbol{B}^\star \boldsymbol{Q} \boldsymbol{B} + \boldsymbol{R}, \quad \boldsymbol{G} = \boldsymbol{A}^\star \boldsymbol{Q} \boldsymbol{B}, \quad \bar{F} = \boldsymbol{A}^\star \boldsymbol{Q} \boldsymbol{A} + Q \ ,$$

$$\boldsymbol{C} = \begin{bmatrix} \boldsymbol{C}_1^\star & \boldsymbol{C}_2^\star & \ldots \end{bmatrix}^\star, \quad \boldsymbol{c} = [c_1^\star, c_2^\star, \ldots]^\star \ , \tag{3.2}$$

where we denote by $\operatorname{diag}(\cdot, \cdot, \ldots)$ the operator that forms a block diagonal matrix of the provided arguments, $\boldsymbol{e}_i$ the (infinite dimensional) row vector with only one nonzero element equal to one at position i, by $[\cdot]_i$ the i^{th} block row of size $p_x \times \infty$ of the corresponding operator and by * the adjoint

of an operator (i.e., the infinite-dimensional analogue of transpose; see Appendix 3.10.1 for a brief introduction to operators). Using the above, (3.1) can be written in the form

$$\begin{array}{ll} \text{minimize} & (1/2)\langle \boldsymbol{u}, \boldsymbol{H}\boldsymbol{u}\rangle + \langle \boldsymbol{h}, \boldsymbol{u}\rangle + r \\ \text{subject to} & \boldsymbol{C}\boldsymbol{u} \leq \boldsymbol{c} \ , \end{array} \tag{3.3}$$

where $\boldsymbol{h} = \boldsymbol{G}^\star x_{\text{init}}$, $r = (1/2)x_{\text{init}}^\star \bar{F} x_{\text{init}}$, $\boldsymbol{u}$ is the infinite sequence $\boldsymbol{u} := [u_0^\star, u_1^\star, \ldots]^\star$ and $\boldsymbol{H}\colon \mathcal{H}_{\boldsymbol{u}} \to \mathcal{H}_{\boldsymbol{u}}$, $\boldsymbol{C}\colon \mathcal{H}_{\boldsymbol{\lambda}} \to \mathcal{H}_{\boldsymbol{u}}$, $\boldsymbol{c} \in \mathcal{H}_{\boldsymbol{\lambda}}$, where $\mathcal{H}$'s are suitable Hilbert spaces specified next: Any sequence

$$\boldsymbol{z} := (z_0, z_1, \ldots)$$

is viewed as an element of an l^2-weighted (or l_w^2) real Hilbert space $\mathcal{H}_{\boldsymbol{z}}$ (see Appendix 3.10.1, Definition 15) induced by the inner product

$$\langle \boldsymbol{z}, \boldsymbol{y}\rangle = \sum_{i=0}^{\infty} w^i z_i^\top y_i \ , \quad \forall\, \boldsymbol{y} \in \mathcal{H}_{\boldsymbol{z}},\, \boldsymbol{z} \in \mathcal{H}_{\boldsymbol{z}} \ , \tag{3.4}$$

where w is an appropriately chosen weight (see Appendix 3.10.1, Definition 15). The norm of any $\boldsymbol{z} \in \mathcal{H}_{\boldsymbol{z}}$ is thus given by

$$\|\boldsymbol{z}\|_{\mathcal{H}_{\boldsymbol{z}}} := \sqrt{\langle \boldsymbol{z}, \boldsymbol{z}\rangle} = \sqrt{\sum_{i=0}^{\infty} w^i \|z_i\|_2^2} \ .$$

Unless stated otherwise, for the rest of the chapter by a Hilbert space we mean the l_w^2 real Hilbert space as just introduced.

3.4 Dualization

Our purpose is to use (an accelerated variant of) the proximal gradient method on the dual of problem (3.3). This can be derived by making use of Fenchel duality (see Section 2.1). By defining

$$f(\boldsymbol{u}) = (1/2)\langle \boldsymbol{u}, \boldsymbol{H}\boldsymbol{u}\rangle + \langle \boldsymbol{h}, \boldsymbol{u}\rangle + r, \quad g(\boldsymbol{C}\boldsymbol{u} - \boldsymbol{c}) = \delta_-(\boldsymbol{C}\boldsymbol{u} - \boldsymbol{c}) \tag{3.5}$$

and $\delta_-(\cdot)$ being the indicator function for the nonpositive orthant, we can rewrite (3.3) as

$$\min_{\boldsymbol{u}} \{f(\boldsymbol{u}) + g(\boldsymbol{C}\boldsymbol{u} - \boldsymbol{c})\} \ .$$

Using the results in Section 2.1, the problem we are interested in solving can be cast in the form:

$$\text{minimize} \quad F(\boldsymbol{\lambda}) := h^\star(\boldsymbol{\lambda}) + \delta_-(\boldsymbol{\lambda}), \tag{3.6}$$

with variables $\boldsymbol{\lambda}$, $h^\star(\boldsymbol{\lambda}) = f^\star(\boldsymbol{C}^\star\boldsymbol{\lambda}) - \langle \boldsymbol{\lambda}, \boldsymbol{c}\rangle$ a differentiable function in $\Gamma_0(\mathcal{H}_{\boldsymbol{\lambda}})$, and $\delta_-(\boldsymbol{\lambda}) \in \Gamma_0(\mathcal{H}_{\boldsymbol{\lambda}})$. Now the PGM iteration (2.9) can be applied to solve (3.6).

Before proceeding, we elaborate on the reasons why the original problem (3.1) had to be reformulated in order to solve it. There are two reformulations, namely, posing the problem as a function of the input sequences only, resulting in (3.3), and dualization of (3.3), resulting in (3.6). The reason

for considering the condensed formulation is the need for *strong convexity* of the primal objective, which implies the Lipschitz continuity of $\nabla h^\star$ [BC11, Corollary 18.16]. By using the condensed form we avoid the restrictive assumption of $Q \succ 0$ required in [SKJ14].

The reason for considering the dual problem is simplicity in the evaluation of the proximal operator of the function $\delta_-(\cdot)$, which is a simple projection on the non-positive orthant (*i.e.,* componentwise clipping) as opposed to the primal case where one would have to project on a generic polytope of the form $\boldsymbol{C}\boldsymbol{u} \leq \boldsymbol{c}$.

3.5 Solution using Accelerated Dual Proximal Gradient Method

Problem (3.6) is a composite minimization problem (*i.e.,* a minimization of a sum of a smooth and a non-smooth function) and will be solved using an inertial proximal gradient method discussed in Section 2.2, the acceleration coming from a Nesterov-like momentum sequence from [CD15].

The several 'optimal' relaxation sequences that we discussed in Section 2.2 have been put under a common framework in the work [CD15]. The authors showed that any sequence $\{t^k\}$ of the form $t^k = \frac{k+a-1}{a}$, with $a \geq 2$ satisfies the inequality $(t^k)^2 - t^k \leq (t^{k-1})^2,\ k \geq 2$. Then the sequence defined as $\alpha^k = \frac{t^k - 1}{t^{k+1}}$ allows for the optimal $O(1/k^2)$ convergence rate in terms of the function values, starting from $t_1 = 1$, and using a constant stepsize $\gamma \in (0, 1/L_f]$, where L_f is the Lipschitz constant of ∇f. In addition, *weak convergence of the iterates* is achievable for any choice $a > 2$. Any such scheme is optimal in the sense that, for every iterate k, there exists a problem which has a lower complexity bound of the same order. We denote the algorithm emanating from this scheme the Accelerated Dual Proximal Gradient Method (ADPG) algorithm, and write it down for problem (3.6) as follows:

Algorithm 4 ADPG for Problem (3.6)

0: Initialize $\boldsymbol{\lambda}^0 = \mathbf{0}$, $a > 2$, $\alpha^0 = 0$,
$L_{h^\star} \leftarrow$ a Lipschitz constant of $\nabla h^\star$.
repeat
1: $\alpha^k = \frac{k-1}{k+a},\ k \geq 1$
2: $\hat{\boldsymbol{\lambda}}^k = \boldsymbol{\lambda}^k + \alpha^k(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^{k-1})$
3: $\boldsymbol{\lambda}^{k+1} = \min\left\{\hat{\boldsymbol{\lambda}}^k - (1/L_{h^\star})\nabla h^\star(\hat{\boldsymbol{\lambda}}^k), 0\right\}$
until termination condition is satisfied.

The iterative scheme above is very simple and ultimately boils down to a variant of the fast projected gradient method as proposed by Nesterov [Nes04a]. In order to apply Algorithm 4, we need to be able to

- evaluate the gradient of $h^\star$
- represent $\boldsymbol{\lambda}^k$ and $\hat{\boldsymbol{\lambda}}^k$ using a finite amount of memory.

The remaining steps of the algorithm are simple scalar or vector updates or componentwise clipping on the non-positive orthant (Step 3), both of which can be carried out inexpensively provided that $\boldsymbol{\lambda}^k$ can be represented using finite amount of memory.

In the rest of the text $P_{\rm LQ}$ and $K_{\rm LQ}$ will denote the positive-semidefinite solution to the discrete-time algebraic Riccati equation and the corresponding LQ optimal state feedback matrix associated with the matrices (A, B, Q, R).

We start by finding the gradient of $h^\star$. To this end, define the Lagrangian of the (infinite dimensional) problem (3.3), written in its equivalent sparse form and truncated at any $T \geq 0$, as

$$\mathcal{L}(\boldsymbol{u}, \boldsymbol{x}, \hat{\boldsymbol{\lambda}}|T) \;=\; x_T^\top P_{\rm LQ} x_T + \sum_{i=0}^{T-1} x_i^\top Q x_i + u_i^\top R u_i - w^i \begin{bmatrix} C_x x_i - c_x \\ C_u u_i - c_u \end{bmatrix}^\top \hat{\lambda}_i \;. \tag{3.7}$$

Lemma 2. *If $\hat{\boldsymbol{\lambda}}$ is such that $\hat{\lambda}_i = 0$ for all $i \geq T$, then*

$$[\nabla h^\star(\hat{\boldsymbol{\lambda}})]_i = w^i \begin{bmatrix} C_x \hat{x}_i - c_x \\ C_u \hat{u}_i - c_u \end{bmatrix}, \tag{3.8}$$

where

$$(\hat{\boldsymbol{x}}, \hat{\boldsymbol{u}}) = \arg\min_{\boldsymbol{u}, \boldsymbol{x}} \{\mathcal{L}(\boldsymbol{u}, \boldsymbol{x}, \hat{\boldsymbol{\lambda}}|T) \mid x_{i+1} = Ax_i + Bu_i,\, x_0 = x_{\rm init}\} \tag{3.9}$$

for $i \in \{0, \ldots, T\}$, and

$$\hat{x}_i = (A + BK_{\rm LQ})^{i-T} \hat{x}_T, \quad \hat{u}_i = K_{\rm LQ} \hat{x}_i$$

for $i > T$.

Proof: This is a standard result from duality (see Algorithm 1 in Chapter 2), noticing that the minimization in (3.9) is equivalent to the minimization of the Lagrangian of (3.3) under the assumption that $\hat{\lambda}_i = 0$ for all $i \geq T$. □

Lemma 2 gives us a way to compute the gradient of $h^\star$. Clearly, this gradient is an infinite sequence and therefore cannot be stored directly, but it is available to us explicitly for $i \in \{0, \ldots, T\}$ and implicitly, through the dynamics of the system $\hat{x}_+ = (A + BK_{\rm LQ})\hat{x}$, for $i > T$.

Now we show that $\boldsymbol{\lambda}^k$ and $\hat{\boldsymbol{\lambda}}^k$ can be represented using a finite amount of memory.

Lemma 3. *If $\boldsymbol{\lambda}^k$ and $\hat{\boldsymbol{\lambda}}^k$ are generated by Algorithm 4, then for each k there exists a $T^k < \infty$ such that $\lambda_i^k = 0$ and $\hat{\lambda}_i^k = 0$ for all $i \geq T^k$.*

Proof: We have $\lambda_i^0 = 0$ and $\hat{\lambda}_i^0 = 0$ for all $i \geq 0$ and hence $T^0 = 0$. Assume now $k > 0$ and $\lambda_i^k = 0$ and $\hat{\lambda}_i^k = 0$ for all $i \geq T^k$. Then, evaluating $\nabla h^\star(\boldsymbol{\lambda}^k)$ using Lemma 2 with $T = T^k$, we see that the sequence $(\hat{x}_i, \hat{u}_i)$ is generated by the unconstrained LQ controller for $i > T^k$ and hence converges to the origin. Since the set $\mathcal{X}$ has the origin in the interior we conclude that there exists a time $\hat{T}^{k+1} < \infty$ such that $C_x \hat{x}_i \leq c_x$ and $C_u \hat{u}_i \leq c_u$ for all $i \geq \hat{T}^{k+1}$. We define $T^{k+1} = \max\{T^k, \hat{T}^{k+1}\}$. In view of (3.8), we conclude that $\hat{\boldsymbol{\lambda}}^k - \frac{1}{L_{h^\star}} \nabla h^\star(\hat{\boldsymbol{\lambda}}^k) \geq 0$ for all $i \geq T^{k+1}$. Therefore, in view of Step 3 of Algorithm 4, we have $\lambda_i^{k+1} = 0$ for all $i \geq T^{k+1}$. Finally, $\hat{\boldsymbol{\lambda}}^{k+1}$ is a linear combination of $\boldsymbol{\lambda}^k$ and $\boldsymbol{\lambda}^{k+1}$ and hence $\hat{\lambda}_i^{k+1} = 0$ for all $i \geq T^{k+1}$. □

To determine T^{k+1} computationally (given T^k and $\hat{\boldsymbol{x}}^{k+1}$ and $\hat{\boldsymbol{u}}^{k+1}$) we simply find the first time $T_{\mathcal{S}}$ that $\hat{x}_i^{k+1}$ enters a given subset $\mathcal{S}$, with $0 \in \text{int}(\mathcal{S})$, of the maximal positively invariant set of

the system $\hat{x}_+ = (A + BK_{\mathrm{LQ}})\hat{x}$ subject to the constraint $\begin{bmatrix} C_u K_{\mathrm{LQ}} \\ C_x \end{bmatrix} \hat{x} \leq \begin{bmatrix} c_u \\ c_x \end{bmatrix}$. The time T^{k+1} is then equal to the first time no less than T^k such that $C_x \hat{x}_{i+1}^{k+1} \leq c_x$ and $C_u \hat{u}_i^{k+1} \leq c_u$ simultaneously hold for all $i \in \{T^{k+1}, \ldots, T_{\mathcal{S}}\}$. More formally, we have the equality

$$T^{k+1} = \min \left\{ T \geq T^k \mid \exists T_{\mathcal{S}} \text{ s.t. } C_x \hat{x}_{i+1}^{k+1} \leq c_x,\ C_u \hat{u}_i^{k+1} \leq c_u \ \forall i \in \{T, \ldots, T_{\mathcal{S}}\},\ \hat{x}_{T_{\mathcal{S}}}^{k+1} \in \mathcal{S} \right\}. \quad (3.10)$$

Remark 3 (Computation of T^k). *In practice, to determine T^{k+1} after solving (3.9), we iterate forward the system dynamics $\hat{x}_+ = (A + BK_{\mathrm{LQ}})\hat{x}$ starting from the initial condition $\hat{x}_{T^k+1}^{k+1}$ until $\hat{x}_i^{k+1} \in \mathcal{S}$.*

Remark 4 (Set $\mathcal{S}$). *The set $\mathcal{S}$ is determined offline and is* not *required to be invariant. A good candidate is the set $\{x \mid x^\top P_{\mathrm{LQ}} x \leq 1\}$ scaled such that it is included in $\left\{ x \mid \begin{bmatrix} C_u K_{\mathrm{LQ}} \\ C_x \end{bmatrix} x \leq \begin{bmatrix} c_u \\ c_x \end{bmatrix} \right\}$, or any subset of this set containing the origin in the interior.*

Now we are ready to formulate an implementable version of the abstract Algorithm 4:

Algorithm 5 ADPG method for the CLQR problem

Require: x_{init}, $Q \succeq 0, R \succ 0$, C_u of full column rank

a: Determine P_{LQ}, K_{LQ} solving the unconstrained LQR problem associated with the matrices (A, B, Q, R).

b: Determine a set $\mathcal{S}$, with $0 \in \text{int}(\mathcal{S})$, included in any positively invariant set for the system $x_+ = (A + BK_{\text{LQ}})x$ subject to the constraint $\begin{bmatrix} C_u K_{\text{LQ}} \\ C_x \end{bmatrix} x \leq \begin{bmatrix} c_u \\ c_x \end{bmatrix}$. See Remark 4.

c: Initialize $\boldsymbol{\lambda}^0 = \mathbf{0}$, $T^0 = 0$, $w = \min\left\{1, \frac{1}{\lambda^2_{\max}(A)}\right\}$, $a > 2$, $\alpha^0 = 0$, $L^0 > 0$ or $L_{h^\star}$ a Lipschitz constant of $\nabla h^\star$ (optional).

for $k = 0, \ldots$ **do**

1: $\alpha^k = \frac{k-1}{k+a}$, $k \geq 1$

2: $\hat{\lambda}_i^k = \lambda_i^k + \alpha^k(\lambda_i^k - \lambda_i^{k-1}), \quad i = 1, \ldots, T^k$

3: Set

$$(\hat{\boldsymbol{x}}^{k+1}, \hat{\boldsymbol{u}}^{k+1}) = \arg\min_{\boldsymbol{u},\boldsymbol{x}} \{\mathcal{L}(\boldsymbol{u}, \boldsymbol{x}, \hat{\boldsymbol{\lambda}}^k | T^k) \mid x_{i+1} = Ax_i + Bu_i,\, x_0 = x_{\text{init}}\}, \quad i = 0, \ldots, T^k,$$
$$\hat{x}_{i+1}^{k+1} = (A + BK_{\text{LQ}})\hat{x}_i^{k+1}, \quad i > T^k$$

4: Determine T^{k+1} (see Remark 3)

6: Choose stepsize γ (see Remark 5)

7: Set $\lambda_i^{k+1} = \min\left(\hat{\lambda}_i^k - \gamma^{k+1} w^i \begin{bmatrix} C_x \hat{x}_i^{k+1} - c_x \\ C_u \hat{u}_i^{k+1} - c_u \end{bmatrix}, 0\right)$.

8: If termination condition is satisfied, solve KKT system (see Remark 7)

end for

Remark 5 (Stepsize). *In Step 6 of the algorithm, a stepsize is selected. One option is to fix a constant stepsize $\gamma = 1/L_{h^\star}$, which needs a global Lipschitz constant of $\nabla h^\star$. This can be computed offline (see Section 3.7.1). Alternatively, one can use a* backtracking stepsize rule*, inspired from [BT09], which can be used in combination with the global estimate. The procedure is analyzed in Section 3.10.3.*

Remark 6 (Role of the weight w). *It is worth mentioning that working in the weighted Hilbert space l^2_w is more than a mathematical formalism and has serious practical implications. In the case of unstable systems, a nontrivial sequence of weights has to be chosen such that the growth of the largest unstable eigenvalue of the state matrix A is bounded by a faster decaying sequence so that the operator $\boldsymbol{C}$ remains bounded, an assumption necessary for applying the proposed method. At the same time the sequence of weights will act as a left preconditioner of $\boldsymbol{C}$, hence a left and right preconditioner on the Hessian operator of the quadratic form in (3.6). This scaling can seriously affect the numerical performance of the proposed algorithm, as we will see in subsequent sections. Note that for stable systems the sequence can be trivially set to one, hence no scaling occurs. These claims are explained in more detail in Appendix 3.10.2.*

Remark 7 (Termination). *Algorithm 5 terminates when a prespecified accuracy is reached in terms of the progress of the dual sequence. The extracted primal sequence is given by*

$$(\boldsymbol{x}^k, \boldsymbol{u}^k) = \arg\min_{\boldsymbol{u},\boldsymbol{x}} \{\mathcal{L}(\boldsymbol{u}, \boldsymbol{x}, \boldsymbol{\lambda}^k | T^k) \mid x_{i+1} = Ax_i + Bu_i,\ x_0 = x_{\text{init}}\} \tag{3.11}$$

for $i \in 0, \ldots, T^k$ *and* $x_i^k = (A + BK_{\text{LQ}})^{i-T^k} x_{T^k}$ *for* $i > T^k$*. In Theorem 5 below it is proven that* $(\boldsymbol{x}^k, \boldsymbol{u}^k)$ *tends to the optimal constrained LQ solution. At any finite iterate, however, the sequence* $(\boldsymbol{x}^k, \boldsymbol{u}^k)$ *may violate the constraints. In order to remedy this we solve upon termination an equality-constrained QP where we minimize the objective function subject to the active constraints at optimality. The active constraints can be (approximately) detected by looking at the nonzero values of the dual vector* $\boldsymbol{\lambda}^k$ *at termination. This step comes at a very small cost since it involves one solution of a KKT system of linear equations.*

3.6 Convergence results

In the previous section we gave an implementable algorithmic scheme that computes the solution to the CLQR problem. Here we provide all the necessary proofs which allow us to assert that the solution generated by Algorithm 5 via (3.11) indeed converges to the true optimizer of the CLQR problem. In what follows $\boldsymbol{\lambda}^\infty$ denotes any optimal solution to the dual problem (3.6) (which exists under Assumption 1 but may not be unique) and $(\boldsymbol{u}^\infty, \boldsymbol{x}^\infty)$ the optimal solution to the primal problem (3.1). Our main result is:

Theorem 5 (Main Theorem). *Suppose Assumption 1 holds and let* $\boldsymbol{\lambda}^k$ *be a sequence of iterates generated by Algorithm 4 and* $(\boldsymbol{x}^k, \boldsymbol{u}^k)$ *the associated primal sequence given by (3.11) and let* L *be a Lipschitz constant of* $\nabla h^\star$*. The following statements hold:*

1. *The composite function* $F(\boldsymbol{\lambda}) = h^\star(\boldsymbol{\lambda}) + \delta_-(\boldsymbol{\lambda})$ *as defined in (3.6) converges as*

$$F(\boldsymbol{\lambda}^k) - F(\boldsymbol{\lambda}^\infty) \leq \frac{a^2 L}{2(k+a-1)^2} \|\boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^\infty\|^2_{\mathcal{H}_{\boldsymbol{\lambda}}} \ .$$

2. *The sequence of the dual iterates* $\{\boldsymbol{\lambda}^k\}$ *converges weakly (see Definition 17 in Appendix 3.10.1) to an optimizer, that is,*

$$\boldsymbol{\lambda}^k \rightharpoonup \boldsymbol{\lambda}^\infty$$

for some $\boldsymbol{\lambda}^\infty \in \arg\min\ F$*.*

3. *The input sequence* $\{\boldsymbol{u}^k\}$ *converges strongly to the unique minimizer as*

$$\|\boldsymbol{u}^k - \boldsymbol{u}^\infty\|_{\mathcal{H}_{\boldsymbol{u}}} \leq a\sqrt{\frac{L}{\mu}} \frac{\|\boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^\infty\|_{\mathcal{H}_{\boldsymbol{\lambda}}}}{(k+a-1)} \ ,$$

where $\mu > 0$ *is the strong convexity constant of* $f(\boldsymbol{u})$*.*

4. *The state sequence* $\{\boldsymbol{x}^k\}$ *converges strongly to the unique minimizer as*

$$\|\boldsymbol{x}^k - \boldsymbol{x}^\infty\|_{\mathcal{H}_{\boldsymbol{x}}} \leq a\sqrt{\frac{\|\boldsymbol{B}\|^2 L}{\mu}}\frac{\|\boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^\infty\|_{\mathcal{H}_{\boldsymbol{\lambda}}}}{(k+a-1)} \ .$$

5. *The sequence* $\{T^k\}$ *is bounded.*

Proof: 1. Convergence of $F(\boldsymbol{\lambda}^k)$ with a constant stepsize is proven in [CD15, Theorem 1]. Convergence at the same rate with an adaptive stepsize generated from the backtracking Algorithm 6 is proven in Lemma 11, Appendix 3.10.3.

2. The proof is stated in [CD15, Theorem 3].

3. The idea is to upper bound the input sequence's convergence rate making use of the first point's result. In order to do so we make use of strong duality. The proof is inspired from [BT14, Theorem 4.1] and is as follows:
Let $\boldsymbol{\lambda}^k \leq 0$ generated from Step 7 of Algorithm 5. Denote

$$\boldsymbol{u}^k = \underset{\boldsymbol{u}\in\mathcal{H}_{\boldsymbol{u}}}{\arg\min}\left\{f'(\boldsymbol{u}) := f(\boldsymbol{u}) + \langle \boldsymbol{\lambda}^k, \boldsymbol{c} - \boldsymbol{C}\boldsymbol{u}\rangle\right\} \ , \tag{3.12}$$

where $f(\boldsymbol{u}) = (1/2)\boldsymbol{u}^\star\boldsymbol{H}\boldsymbol{u} + \boldsymbol{h}^\star\boldsymbol{u} + r$ as defined in Section 3.1. We then have that the Lagrangian of (3.3) evaluated at $\boldsymbol{\lambda}^k$ is $\mathcal{L}(\boldsymbol{u}, \boldsymbol{\lambda}^k) = f'(\boldsymbol{u})$. The function $f(\boldsymbol{u})$ is strongly convex with constant $\mu \geq \lambda_{\min}(R) > 0$, where $\lambda_{\min}(R)$ denotes the smallest eigenvalue of R. Strong convexity of $f'(\boldsymbol{u})$ with modulus μ follows directly. Using (3.12), it holds that

$$f'(\boldsymbol{u}) - f'(\boldsymbol{u}^k) \geq \frac{\mu}{2}\|\boldsymbol{u} - \boldsymbol{u}^k\|^2_{\mathcal{H}_{\boldsymbol{u}}}, \quad \forall \boldsymbol{u} \in \mathcal{H}_{\boldsymbol{u}} \ ,$$

or, equivalently,

$$\mathcal{L}(\boldsymbol{u}, \boldsymbol{\lambda}^k) - \mathcal{L}(\boldsymbol{u}^k, \boldsymbol{\lambda}^k) \geq \frac{\mu}{2}\|\boldsymbol{u} - \boldsymbol{u}^k\|^2_{\mathcal{H}_{\boldsymbol{u}}}, \quad \forall \boldsymbol{u} \in \mathcal{H}_{\boldsymbol{u}} \ . \tag{3.13}$$

Substituting $\boldsymbol{u} = \boldsymbol{u}^\infty$ in (3.13) and by observing that $\max_{\boldsymbol{\lambda}\leq 0} \mathcal{L}(\boldsymbol{u}^\infty, \boldsymbol{\lambda}) \geq \mathcal{L}(\boldsymbol{u}^\infty, \boldsymbol{\lambda}^k)$, we have that

$$\mathcal{L}(\boldsymbol{u}^\infty, \boldsymbol{\lambda}^\infty) - \mathcal{L}(\boldsymbol{u}^k, \boldsymbol{\lambda}^k) \geq \frac{\mu}{2}\|\boldsymbol{u}^\infty - \boldsymbol{u}^k\|^2_{\mathcal{H}_{\boldsymbol{u}}}, \quad \forall \boldsymbol{u} \in \mathcal{H}_{\boldsymbol{u}} \ . \tag{3.14}$$

We have managed to derive an upper bound for the distance of the generated sequence of primal minimizers $\{\boldsymbol{u}^k\}$ from the optimal one. The last step is to show that the Lagrangian

$\mathcal{L}(\boldsymbol{u}, \boldsymbol{\lambda})$ is associated to the composite objective $F(\boldsymbol{\lambda})$. This can be easily shown as follows:

$$\begin{aligned}\mathcal{L}(\boldsymbol{u}^k, \boldsymbol{\lambda}^k) &= \min_{\boldsymbol{u}\in\mathcal{H}_{\boldsymbol{u}}} \Big\{ f(\boldsymbol{u}) + \langle \boldsymbol{\lambda}^k, \boldsymbol{c} - \boldsymbol{C}\boldsymbol{u}\rangle \Big\} \\ &= -\max_{\boldsymbol{u}\in\mathcal{H}_{\boldsymbol{u}}} \Big\{ -f(\boldsymbol{u}) + \langle \boldsymbol{\lambda}^k, \boldsymbol{C}\boldsymbol{u}\rangle \Big\} + \langle \boldsymbol{\lambda}^k, \boldsymbol{c}\rangle \\ &= -f^\star(\boldsymbol{C}^\star\boldsymbol{\lambda}^k) + \langle \boldsymbol{\lambda}^k, \boldsymbol{c}\rangle \\ &= -F(\boldsymbol{\lambda}^k), \quad \text{by (3.6)} \ .\end{aligned}$$

From strong duality and the fact that $-F(\boldsymbol{\lambda}^k)$ converges to the optimal dual value (first point), we have that the optimal value of the dual function $-F(\boldsymbol{\lambda})$ coincides with that of the Lagrangian evaluated at the saddle point $(\boldsymbol{u}^\infty, \boldsymbol{\lambda}^\infty)$, *i.e.,* $\mathcal{L}(\boldsymbol{u}^\infty, \boldsymbol{\lambda}^\infty) = \max_{\boldsymbol{\lambda}\in\mathcal{H}_{\boldsymbol{\lambda}}} \{-F(\boldsymbol{\lambda})\} = -F(\boldsymbol{\lambda}^\infty)$ (see [BV04, Section 5.5.5]). Making use of the first point, inequality (3.14) becomes

$$\frac{\mu}{2}\|\boldsymbol{u}^k - \boldsymbol{u}^\infty\|^2_{\mathcal{H}_{\boldsymbol{u}}} \leq F(\boldsymbol{\lambda}^k) - F(\boldsymbol{\lambda}^\infty) \leq \frac{a^2 L\|\boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^\infty\|^2_{\mathcal{H}_{\boldsymbol{\lambda}}}}{2(k+a-1)^2}, \tag{3.15}$$

which concludes the proof.

4. The state sequence is generated by

$$\boldsymbol{x}^k = \boldsymbol{A}x_{\text{init}} + \boldsymbol{B}\boldsymbol{u}^k \ . \tag{3.16}$$

Strong convergence of the input sequence $\{\boldsymbol{u}^k\}$, along with the facts that $\boldsymbol{B}\colon \mathcal{H}_{\boldsymbol{x}} \to \mathcal{H}_{\boldsymbol{u}}$ is bounded (follows directly from Lemma 7 in Appendix 3.10.2) and the uniqueness of $\boldsymbol{u}^\infty$ prove strong convergence of the state sequence with rate $1/k$, *i.e.,*

$$\begin{aligned}\|\boldsymbol{x}^k - \boldsymbol{x}^\infty\|_{\mathcal{H}_{\boldsymbol{x}}} &= \|\boldsymbol{B}(\boldsymbol{u}^k - \boldsymbol{u}^\infty)\|_{\mathcal{H}_{\boldsymbol{x}}} \\ &\leq \|\boldsymbol{B}\|\|\boldsymbol{u}^k - \boldsymbol{u}^\infty\|_{\mathcal{H}_{\boldsymbol{x}}} \\ &\leq a\|\boldsymbol{B}\|\sqrt{\frac{L}{\mu}}\frac{\|\boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^\infty\|_{\mathcal{H}_{\boldsymbol{\lambda}}}}{(k+a-1)} \ ,\end{aligned}$$

with the last inequality following directly from the third point.

5. The proof for the unaccelerated case was presented in [SKJ14]. We give below the complete version of the proof taking into account the appearance of the over-relaxed sequences $\{\hat{\boldsymbol{x}}^k\}$, $\{\hat{\boldsymbol{u}}^k\}$, which renders the derivation of the result slightly more challenging.

   The key piece for the proof is the weak convergence of $\hat{\boldsymbol{x}}^k$ to $\boldsymbol{x}^\infty$. This claim is subsequently proven as a sequence of intermediate results. We show that:

   (a) The relaxed dual sequence $\{\hat{\boldsymbol{\lambda}}^k\}$ converges weakly to a dual minimizer $\boldsymbol{\lambda}^\infty$.

   (b) Provided that the operator $\boldsymbol{C}$ is bounded, the sequence $\{\boldsymbol{H}\hat{\boldsymbol{u}}^k\}$ converges weakly to $\{\boldsymbol{H}\boldsymbol{u}^k\}$, and $\{\hat{\boldsymbol{u}}^k\}$ converges weakly to $\{\boldsymbol{u}^k\}$. From strong duality, $\{\boldsymbol{u}^k\}$ converges to the

primal optimizer $\boldsymbol{u}^\infty$.

(c) Weak convergence of the accelerated state sequence $\{\hat{\boldsymbol{x}}^k\}$ to $\boldsymbol{x}^\infty$ follows directly.

Weak convergence of the relaxed sequence $\{\hat{\boldsymbol{\lambda}}^k\}$ follows from Corollary 2 of [CD15], which states that the error sequence $\{\|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^{k-1}\|^2\}$ converges to zero with rate $1/k^2$. We state the result below.

Lemma 4. *The relaxed sequence $\{\hat{\boldsymbol{\lambda}}^k\}$ converges weakly to $\boldsymbol{\lambda}^\infty$.*

Proof: Since $\|\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^{k-1}\|^2 \to 0$ and α^k is bounded we also have $\boldsymbol{\nu}^k = \sqrt{\alpha^k}(\boldsymbol{\lambda}^k - \boldsymbol{\lambda}^{k-1}) \to 0$. Since strong convergence implies weak convergence we have that $\langle \boldsymbol{\nu}^k, \boldsymbol{y}\rangle \xrightarrow{k\to\infty} 0,\ \forall \boldsymbol{y} \in \mathcal{H}_{\boldsymbol{\lambda}}$. The relaxed sequence of duals $\hat{\boldsymbol{\lambda}}$ can be written as $\hat{\boldsymbol{\lambda}}^k = \boldsymbol{\lambda}^k + \sqrt{\alpha^k}\boldsymbol{\nu}^k$. Consequently, since $\boldsymbol{\lambda}^k \rightharpoonup \boldsymbol{\lambda}^\infty$, we have that

$$\begin{aligned}\langle \hat{\boldsymbol{\lambda}}^k, \boldsymbol{y}\rangle &= \langle \boldsymbol{\lambda}^k + \sqrt{\alpha^k}\boldsymbol{\nu}^k, \boldsymbol{y}\rangle \\ &= \langle \boldsymbol{\lambda}^k, \boldsymbol{y}\rangle + \sqrt{\alpha^k}\langle \boldsymbol{\nu}^k, \boldsymbol{y}\rangle \to \boldsymbol{\lambda}^\infty\end{aligned}$$

for all $y \in \mathcal{H}_{\boldsymbol{\lambda}}$ and hence $\hat{\boldsymbol{\lambda}}^k \rightharpoonup \boldsymbol{\lambda}^\infty$. □

Lemma 5. *The sequence $\{\hat{\boldsymbol{u}}^k\}$ converges weakly to $\boldsymbol{u}^\infty$.*

Proof: Writing down the relation between $\hat{\boldsymbol{u}}$ and $\hat{\boldsymbol{\lambda}}$ from Lemma 2 in terms of the operators, we get $\hat{\boldsymbol{u}}^k = \boldsymbol{H}^{-1}(\boldsymbol{C}^\star \boldsymbol{W}\hat{\boldsymbol{\lambda}}^k - \boldsymbol{h})$. Similarly we have $\boldsymbol{u}^k = \boldsymbol{H}^{-1}(\boldsymbol{C}^\star \boldsymbol{W}\boldsymbol{\lambda}^k - \boldsymbol{h})$. Now, from Theorem 5 we have $\boldsymbol{u}^k \to \boldsymbol{u}^\infty$ and from Lemma 4 we have $\hat{\boldsymbol{\lambda}}^k - \boldsymbol{\lambda}^k \rightharpoonup 0$. Therefore, since $\boldsymbol{C}^\star$, $\boldsymbol{W}$ and $\boldsymbol{H}^{-1}$ are bounded operators (see Lemmas 7 and 8 in Appendix 3.10.2) and since weak convergence is preserved under bounded linear mappings, we conclude that $\hat{\boldsymbol{u}}^k \rightharpoonup \boldsymbol{u}^\infty$. □

Lemma 6. *The sequence $\{\hat{\boldsymbol{x}}^k\}$ converges weakly to $\boldsymbol{x}^\infty$.*

Proof: Exactly as we did at the fourth point of Theorem 5, the accelerated state sequence can be written as

$$\hat{\boldsymbol{x}}^k = \boldsymbol{A}x_{\text{init}} + \boldsymbol{B}\hat{\boldsymbol{u}}^k \ .$$

Weak convergence $\hat{\boldsymbol{u}}^k \rightharpoonup \boldsymbol{u}^\infty$ and boundedness of $\boldsymbol{B}$ prove weak convergence of the accelerated state sequence to $\boldsymbol{x}^\infty$. □

We have, hence, proven the weak convergence of the accelerated state sequence to the optimal one. For $\{T^k\}$ to be bounded, it is sufficient to show that

$$\limsup_{k\to\infty} T^k < \infty. \tag{3.17}$$

To prove (3.17), define the sequence of the first hitting times of the interior of $\mathcal{S}$ as

$$\tau^k := \inf\{i \geq T^k \mid x_i^k \in \text{int}(\mathcal{S})\},\ \ k \in \mathbb{N} \cup \{+\infty\},$$

where $\tau^\infty < \infty$ is the hitting time of the optimal state sequence $\boldsymbol{x}^\infty$. Clearly, $\tau^k \geq T^k$ and $\tau^k < \infty$ since the origin is in the interior of $\mathcal{S}$ and for each $k \in \mathbb{N}$ the sequence $\{\hat{x}_i^k\}_{i=0}^\infty$ generated by Algorithm 5 converges to the origin as $i \to \infty$. We shall prove that $\limsup_{k\to\infty} \tau^k \leq \tau^\infty < \infty$, which implies (3.17). For the purpose of contradiction assume that there exists a subsequence τ^{k_j}, $j \in \mathbb{N}$, with $\lim_{j\to\infty} \tau^{k_j} \geq \tau^\infty + 1$. Since the sequence of hitting times τ^k is integer valued, this implies that there exists a $j^* \in \mathbb{N}$ such that $\tau^{k_j} \geq \tau^\infty + 1$ for all $j \geq j^*$. We now use this to contradict the weak convergence of $\hat{\boldsymbol{x}}^k$ to $\boldsymbol{x}^\infty$. To this end, observe that $x_{\tau^\infty}^\infty \in \operatorname{int}(\mathcal{S})$ whereas $\hat{x}_{\tau^\infty}^{k_j} \notin \operatorname{int}(\mathcal{S})$ for all $j \geq j^*$. By the definition of the interior there exists an $\epsilon > 0$ such that $z \in \operatorname{int}(\mathcal{S})$ for all z with $\|z - x_{\tau^\infty}^\infty\|_2 < \epsilon$. Therefore $\|\hat{x}_{\tau^\infty}^{k_j} - x_{\tau^\infty}^\infty\|_2 \geq \epsilon$ for all $j \geq j^*$, and consequently

$$\begin{aligned}\langle \hat{\boldsymbol{x}}^{k_j} - \boldsymbol{x}^\infty, \boldsymbol{z}\rangle &= \sum_{i=0}^{\infty} w^i (\hat{x}_i^{k_j} - x_i^\infty)^\top z_i \\ &\geq w^{\tau^\infty} (\hat{x}_{\tau^\infty}^{k_j} - x_{\tau^\infty}^\infty)^\top z_{\tau^\infty} \\ &\geq w^{\tau^\infty} \epsilon^2 > 0 \ ,\end{aligned}$$

for a sequence $\boldsymbol{z}$ with $z_{\tau^\infty} = \hat{x}_{\tau^\infty}^{k_j} - x_{\tau^\infty}^\infty$ and $(\hat{x}_i^{k_j} - x_i^\infty)^\top z_i \geq 0 \ \forall i \neq \tau^\infty$, and for all $j \geq j^*$, contradicting the weak convergence of $\hat{\boldsymbol{x}}^k$ to $\boldsymbol{x}^\infty$ asserted by Lemma 6.

□

3.7 Computational aspects and warm-starting

Having presented the algorithm and its convergence results, we now focus on the computational aspects that render the algorithm a practical option to alternatives such as MPC. We start with explaining how no prior knowledge of a fixed stepsize is needed, give some references concerning the solution of the linear system, which is the most expensive operation of the method, and we conclude the section by suggesting a warm-starting scheme.

3.7.1 Stepsize selection

The stepsize used in Algorithm 5 is computed as the reciprocal of the Lipschitz constant of $h^\star$. For the problem discussed here this can be explicitly computed. We have that

$$\begin{aligned}\|\nabla h^\star(\boldsymbol{\lambda}_1) - \nabla h^\star(\boldsymbol{\lambda}_2)\| &= \|\boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{C}^\star\boldsymbol{W}(\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2)\| \\ &\leq \|\boldsymbol{C}\boldsymbol{H}^{-1}\boldsymbol{C}^\star\boldsymbol{W}\|\|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\| \\ &= \|\boldsymbol{H}^{-1}\|\|\boldsymbol{C}\|^2\|\boldsymbol{\lambda}_1 - \boldsymbol{\lambda}_2\| \ ,\end{aligned}$$

where $\boldsymbol{W}$ is the diagonal operator constituted of the decaying weighting sequence, *i.e.,* $\boldsymbol{W} = \operatorname{diag}(I, wI, w^2 I, \ldots)$. The last equality follows from the fact that $\boldsymbol{W}$ contains a non-increasing sequence with the largest element being one. Hence $L_{h^\star} = \|\boldsymbol{H}^{-1}\|\|\boldsymbol{C}\|^2$, which requires computation of the operator norms $\|\boldsymbol{H}^{-1}\|$ and $\|\boldsymbol{C}\|$. The proofs for boundedness of the two operators, as well as the computations of their bounds are derived in Appendix 3.10.2, Lemmas 7 and 8.

Although valid, this offline computation of the stepsize tends to be conservative in many cases, due to the conservativeness of the computed upper bounds. An elegant and practical method to achieve faster convergence is to employ an algorithm that locally estimates the Lipschitz constant online, at every iteration of Algorithm 5. In order to do so, we use the *backtracking stepsize rule* suggested in [BT09]. The idea is simple: after each iteration of the algorithm, we make a quadratic approximation model of the function around the successor point, making use of the knowledge of the exact point and its gradient value. A quadratic term with varying curvature is added on top of the linear (first-order Taylor) approximation, and the curvature is adapted recursively until our quadratic approximant upper bounds the original function, centered around the given point. Thus, the quadratic model's curvature is an estimate of the Lipschitz constant of the gradient of the original function. It is proven in [BT09] that the locally evaluated Lipschitz constant L^k is related to the global one by $\beta L_{h^\star} \leq L^k \leq \gamma L_{h^\star}$, where $\beta = \frac{L^0}{L_{h^\star}}$ and $\gamma > 1$, $L^0 > 0$ being an initial estimate. Consequently the rule allows for smaller L's and hence larger stepsizes, *i.e.,* faster practical convergence.

Remark 8 (Backtracking). *Although points (1), (3) and (4) of Theorem 5 can be easily shown to hold for a stepsize generated from the backtracking procedure described above, the same does not hold for point (2), and, consequently, for point (5). More specifically, weak convergence of the dual sequence is based on the assumption that no stepsize larger than $2/L_{h^\star}$ is allowed at any iteration [CD15, Theorem 3].*

3.7.2 Complexity

The most expensive operation of Algorithm 5 is the linear system solve in Step 3. There is a variety of ways to perform this step, *i.e.,* solve the KKT system or perform the Riccati recursion when both states and inputs are considered, invert the dense Hessian when only the inputs are considered. In the first case, a sparse (permuted) $\mathrm{LDL}^\top$ factorization can be performed with cost $T(n+m)^3$ flops, followed by a forward-backward solve at $T(n+m)^2$ flops. This approach has been followed in [OSB13]. A discussion on the KKT system solve and the Riccati recursion approach is contained in [FJ13b], where the corresponding complexities are analyzed and compared in detail.

In the case of the condensed formulation, the linear system solve can be efficiently performed by first applying a Cholesky factorization on H (being a finite truncation of the $\boldsymbol{H}$ operator in (3.3)), followed by a forward-backward substitution, (see [BV04, Appendix C]). Although the condensed form of the optimal control problem that is used in the derivations is, generally, not advised for long horizons, recent advancements can render this approach very efficient [AM12], [FJ13a]. More specifically, the two proposed algorithms that perform factorization and solve of the condensed system in [FJ13a] come with a reduced complexity of $O(Tn^3)$ and $O(T^2n^2)$, respectively.

Whether considering the sparse or the dense formulation, note that the factorization steps would have to be performed several times until the 'correct' horizon T^∞ has been identified, since the size of the corresponding matrices (KKT or Hessian H) increase as the algorithm progresses. This typically happens within the first few tens of iterations. A valid alternative to this is the Riccati recursion, which completely eliminates the need for factorization, at the expense of forward (or backward) simulating the trajectories for the gain, the inputs and the states.

The backtracking scheme contributes to the complexity of Algorithm 5 by requiring a number of

function evaluations per iteration, both for the quadratic model and for the original smooth function $h^\star$ (see [BT09] for more details). The rest of the steps are simple vector updates of negligible cost.

3.7.3 Warm-starting

In the nominal case, *i.e.,* when no noise and no model uncertainty are present, the open loop infinite-horizon control sequence generated from Algorithm 5 coincides with the control sequence generated by the optimal closed-loop feedback controller. Consequently, there is no need to re-optimize in a receding-horizon fashion. Solving the CLQR problem just once is sufficient.

In the more realistic scenario where the predicted initial state differs from the measured one, the algorithm has to be re-applied. Provided that the prediction is not very different from the actual state, a good strategy is to initialize the decision variables (states and inputs) of the new problem with the values predicted from the previous one. This is commonly known as warm-starting. In our case, warm-starting has to be performed in the dual variables.

Note that once Algorithm 5 has run once, a hitting time T^∞ has been generated, along with an optimal dual sequence λ^∞ of corresponding length. It is expected that when computing the control law the hitting time should decrease by one at each solve, provided warm-starting from the optimal dual sequence that was generated once in the beginning. Hence, we suggest a heuristic scheme where the 'constrained' (nonzero) part of the preceding shifted dual sequence is used to initialize each subsequent CLQR problem. The computation time thus reduces significantly, with the horizon practically shrinking to zero once the initial state is identified to be inside the maximal positively invariant set of the LQ controller.

Application of the scheme is presented in Section 3.8. It is observed that it behaves particularly well for small perturbations of the initial state.

3.8 Examples

For illustrative purposes, we run the algorithm on two systems, a small unstable system with two states and one input and a linearized model of a quadcopter with 12 states and 4 inputs. We use the small example as a benchmark for graphical illustrations, while the larger one exhibits the computational efficiency of the proposed scheme. The comparison is performed against the same implementation of the ADPG algorithm for finite horizon lengths. It is of course understood that there exist several methods capable of solving a finite horizon MPC problems (see interior point, active set, etc.), among which, optimal first-order methods have gained considerable attention over the last few years, rendering them a competitive alternative [RJM09], [SSSPJ16]. Consequently, comparing against an optimal first-order method provides a valid basis for evaluating the potential of our scheme. In the two examples the termination criterion is simply set as $\|\boldsymbol{\lambda}^k - \hat{\boldsymbol{\lambda}}^{k+1}\| \leq 10^{-4}$.

3.8.1 Toy system

Consider the following system defined as

$$A = \begin{bmatrix} 1.1 & 2 \\ 0 & 0.95 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 0.0787 \end{bmatrix},$$

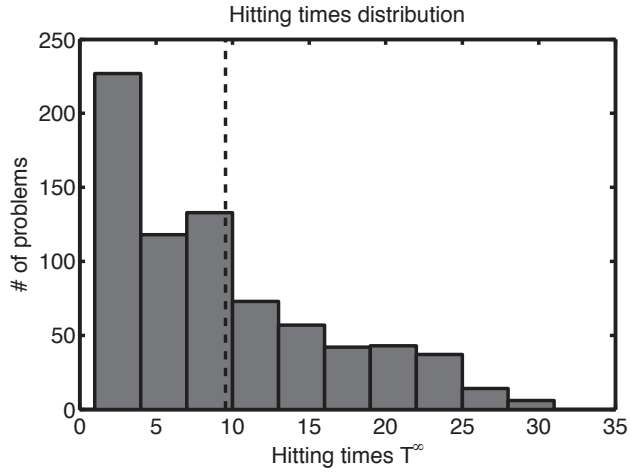


Figure 3.1: Histogram of $T^\infty = \max_k\{T^k\}$ for 750 initial conditions of the 2 state system sampled from a normal distribution around $(-3, 0.3)$ with covariance matrix $\text{diag}(4, 0.4)$.

$$x_{i+1} = Ax_i + Bu_i,$$

with constraints

$$\|x\|_\infty \leq 10, \quad \|u\|_\infty \leq 1$$

and $Q = \begin{bmatrix} 2 & -2 \\ -2 & 2 \end{bmatrix}$, $R = 2I$. Note that the system is unstable, hence a nonzero sequence of weights has to be chosen in order to ensure boundedness of the $\boldsymbol{C}$ operator. We choose $w = 1/1.1^2$ and the value of a in Step 1 of Algorithm 5 is set to $a = 4$.

The system is simulated for 750 different initial conditions x_0. In Figure 3.1 the distribution of $T^\infty = \max_k\{T^k\}$ is depicted. We see that T^∞ goes up to 30, while the mean value is nine.

We compare our method to the MPC approach both from the control and the algorithmic performance perspective. Regarding the former, we perform a comparison of the feasible sets for finite horizon implementations versus the maximal control invariant set in the case of CLQR, as well as the optimal value of the cost function. Regarding the latter, we perform comparisons in terms of the average number of iterations needed for convergence for several horizon length values in MPC versus the CLQR case. We also evaluate the conservativeness of our approach by computing the actual 'optimal' horizon length for each initial condition we simulate. For both CLQR and MPC we make use of the same AFBS algorithm with backtracking employed and termination tolerance set to 10^{-4} as stated before.

We perform the following simulations: we sample 243 initial conditions $x_i^0,\ i = 1, \ldots, 243$ from the maximal control invariant set (31-step) of the aforementioned constrained system (see Figure 3.2). For each point we compute:

1. The minimum horizon length $T_{\min}$, such that $x^*_{T_{\min}}(x_i^0)$ resides in the maximal positively invariant set of the autonomous system $x^+ = (A + BK_{\text{LQ}})x$, used as a terminal set.

2. The hitting time $T = T^\infty$ generated by our proposed scheme.

The following scenarios are generated: Firstly, an MPC problem with terminal set and horizon length $T_{\min}$ is solved, as described above. Subsequently we remove the terminal constraint and solve the MPC problem again for the same horizon length $T_{\min}$. We repeat the procedure described above (MPC with and without terminal set) for horizons $T = 2T_{\min}$ and $T = T^*$, where T^* is specified by identifying the first horizon length for which the feasible solution of the MPC problem had the terminal constraint inactive. Finally, a comparison with the proposed CLQR approach is performed, with horizon $T = T^\infty$. In this way, six MPC problems with finite horizon as well as the CLQR problem are solved, for each of the 243 initial conditions. The total number of iterations is averaged by the number of corresponding initial conditions with the same minimum horizon length $T_{\min}$. The results are presented in Figure 3.3.

The first observation from the plot is that inclusion of the terminal constraint generally increases the number of required iterations since more constraints become active at optimality. This is especially the case when the horizon is relatively short and the terminal constraint satisfaction is imposed as we see in the blue curves. As the horizon increases (red and magenta color), the terminal constraint might be inactive and there is no significant difference in the number of iterations in the two cases.

A trend of increase in the number of iterations in all methods as T^* increases can be observed. This is expected since increase of T^* amounts to sampling of the initial state from regions of the feasible set that are further away from the origin. A consequence of the latter is that more constraints become active at optimality.

Short horizons in combination with a terminal set increase significantly the iteration count, as well as unnecessarily long horizons (*e.g.,* $2T_{\min}$). In that sense, one can observe that MPC with $T = T^*$ behaves better than the other lengths. The curve corresponding to CLQR is denoted with black lines with asterisks. The method is comparable to the MPC approaches where $T = T_{\min}$ and $T = T^*$ with terminal set, performing relatively better in small to medium horizon lengths and worse for larger horizon lengths. The reason for the latter is that the weighting operator $\boldsymbol{W}$ becomes quickly ill-conditioned as T grows, since its diagonal elements decay exponentially. This leads to ill-conditioning of the finite-dimensional truncation of the dual problem (3.6) since $\boldsymbol{W}$ preconditions (through the weighted l_2 inner product (3.4)) the objective of (3.6). As is commonly known first-order methods struggle with ill-conditioning. Efficient preconditioning of the operators should improve this and is a topic of further investigation. As a matter of fact, when the weights are set to one, the CLQR approach (black line with circles) clearly outperforms all MPC approaches with terminal set, as well as most of those instances without terminal set. Further simulations suggest that the proposed method performs very well, were we to drop the weights or when dealing with stable systems, when no scaling has to be performed.

Another interesting point is that the average hitting times T^∞ generated from Algorithm 5 are almost identical to the averaged optimal ones T^*, with a very slight increase. This fact is depicted in Figure 3.4, where the ratio is always very close to one. On the contrary, the minimum required horizon length $T_{\min}$ is observed to be up to 45% smaller than the optimal length T^* in some cases, leading to a significant increase in the objective's cost when compared to the infinite horizon approach.

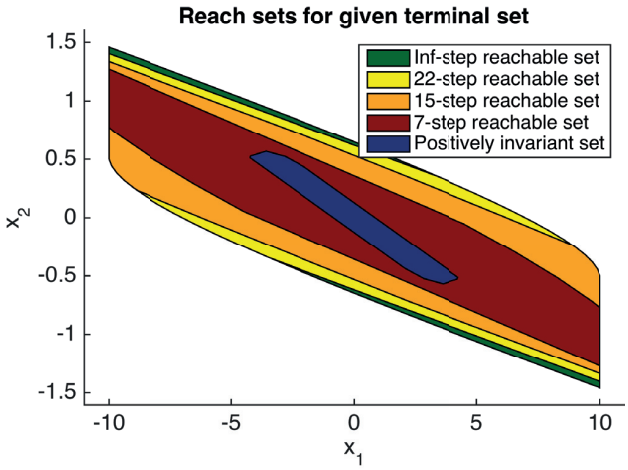


Figure 3.2: Reachable sets for several horizon lengths and the LQ terminal set. The computation was done using the MPT3 toolbox [HKJM13]. It is apparent that a short horizon length reduces significantly the feasible region of the problem.

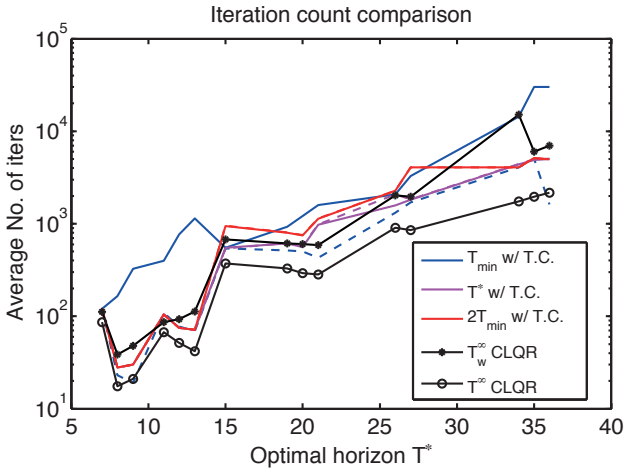


Figure 3.3: Comparison of MPC with finite horizon length, for several horizons, with CLQR. The horizontal axis corresponds to the optimal horizon length T^* per initial condition. As T^* increases, the corresponding states are sampled further from the origin. MPC with terminal set is depicted with the solid lines, while without terminal set with dashed lines. CLQR is performed both with the weight sequence, denoted as T_w^∞ CLQR, and without the weights, denoted as T^∞ CLQR.

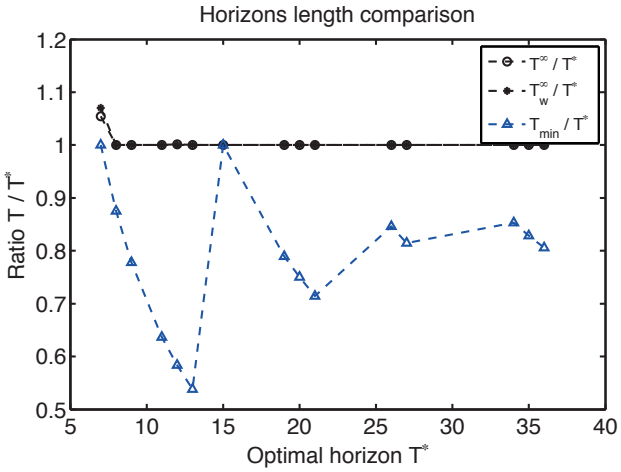


Figure 3.4: Evolution of the ratios $\frac{T_{\min}}{T^*}$, $\frac{T^\infty_w}{T^*}$ and $\frac{T^\infty}{T^*}$ for the sampled initial conditions.

3.8.2 Quadcopter system

The next system we consider is a quadcopter linearized in a hovering equilibrium. The system has 12 states which correspond to position, angle and the corresponding velocities. There are four inputs corresponding to the four propellers. There are box constraints on all states and inputs, mainly ensuring the validity of the linearized model. The system is marginally stable; thus the weight w is set to one.

We simulate the algorithm starting from initial conditions randomly selected as follows: starting from a random feasible initial condition, we generate random directions on a unit ball centered around it and sample points along each of them. The points are generated from a normal distribution with standard deviation 0.15. Finally, we keep the initial conditions that result in feasible closed loop problems. The result of this step is 272 feasible initial conditions for the CLQR Algorithm 5. A histogram of $T^\infty = \max_k\{T^k\}$ is presented in Figure 3.5.

We conclude the quadcopter example by applying the warm-starting heuristic scheme suggested in Section 3.7. We consider two scenarios; in the first one we uniformly perturb the initial state by 0.5% of its nominal value and run Algorithm 5 in closed loop for 78 different initial conditions. In the second scenario we perturb the state by 1%, for 68 different initial conditions. We solve 15 consecutive problems per initial condition and subsequently compare the average number of iterations as well as the generated hitting times per problem solve with and without warm-starting the dual variables. The results are summarized in Figure 3.6. It is evident that warm-starting consistently reduces the number of iterations in both cases.

3.8.3 Timings

We conclude this section with a numerical evaluation. For this purpose, we compare an implementation of the proposed algorithm with several modern convex optimization solvers, employed to solve the corresponding finite horizon MPC problems. In addition, we compare our proposed

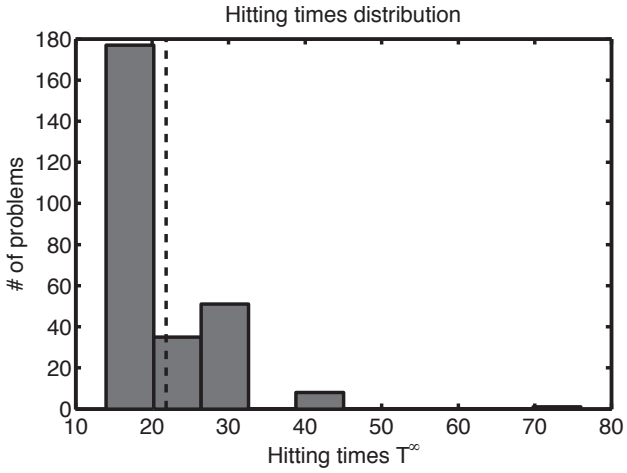


Figure 3.5: Histogram of $T^{\infty} = \max_k\{T^k\}$ for 272 initial conditions sampled with a Hit-And-Run algorithm.

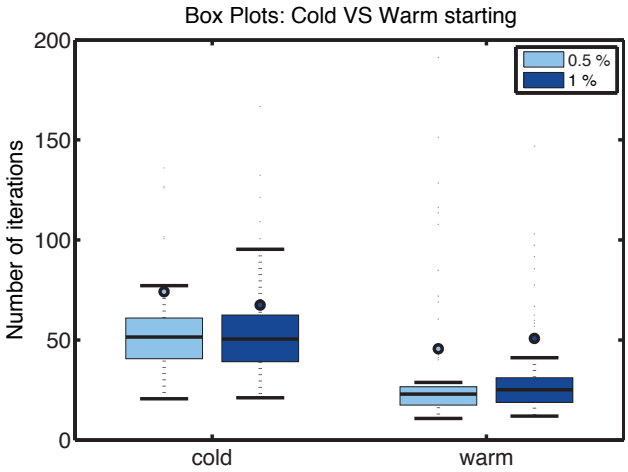


Figure 3.6: Box plots for the average number of iterations of the warm-starting policy, given 0.5% and 1% uniform perturbations of the initial state are depicted. The horizontal line inside the box corresponds to the median, the edges of the box are the 25^{th} and 75^{th} percentiles while the horizontal lines outside the boxes correspond to the most extreme data points not considered outliers. Finally, the colored dots correspond to mean values. Warm-starting improves the performance of the suggested method in both cases, in all the depicted statistical measures.

algorithm with its finite horizon implementation, solving the same optimal control problems in a receding-horizon mode. Given that MPC is an approximation to the solution of the infinite horizon problem, a fair comparison between the two approaches cannot be easily performed. Hence, we consider a 'criterion of fairness' *the time required by Algorithm 5 versus a rolling-horizon approach so that the infinite-horizon optimal solution to a given regulation problem is achieved.*

The difficulty with the aforementioned approach is to construct a rolling horizon scheme that can recover the infinite-horizon optimal solution. This can be achieved only if a sufficiently long horizon is used, so that the state sequence converges leisurely to the origin. Such a horizon is the output of our scheme. Thus, we perform the following simulation with the toy example presented above: We solve a series of optimal control problems for each of the 750 sampled initial conditions, the hitting times of which were depicted in Figure 3.1, using the resulting T^{∞} from the CLQR approach, without imposing any terminal set. The problems are solved in a receding-horizon fashion until the initial state resides inside the maximal positively invariant set of the LQ controller, depicted in Figure 3.2 in dark blue. Subsequently, we measure the time required to compute the closed-loop solution with the MPC algorithms and we compare against the time required by Algorithm 5.

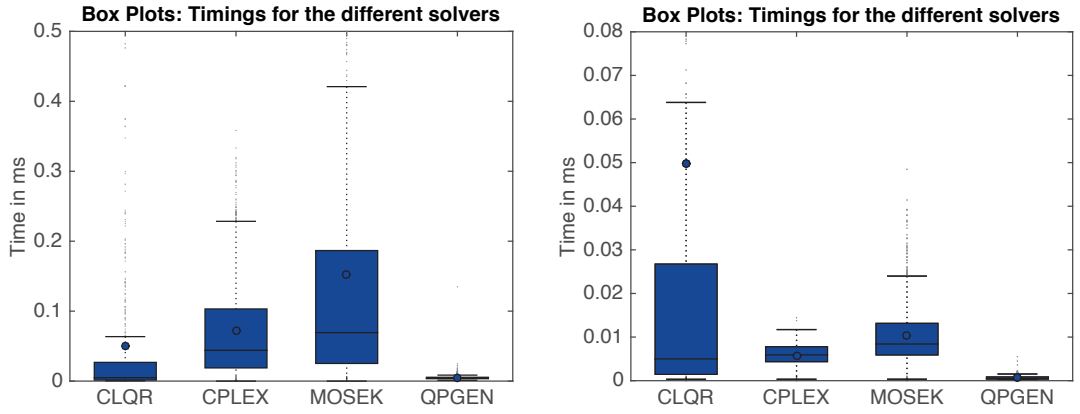
The algorithm is implemented in C++. The algorithmic values a and L^0 are initialized to $a = 5$ and $L^0 = 0.01$. The linear algebra library 'Armadillo' [San10] is used for the operations, which is a wrapper for BLAS and LAPACK. The implementation is tested against the (commercial) solvers CPLEX (IBM) and MOSEK, as well as the open solver QPgen [GB14a; GB15].

The results regarding the first experiment are depicted in Figure 3.7a. Since no noise is injected in the state, the CLQR controller is computed only once, in contrast to the MPC controller where multiple problems have to be solved in order to acquire the closed-loop solution. As a consequence, the total time needed to solve the problem with CLQR is smaller than what most of the solvers in use can achieve. The relative accuracy of the generated solutions is of the order 10^{-5}, while the times are given in ms. In addition, Figure 3.8 elaborates on the iteration count, as well as the number of factorizations that had to be performed while running CLQR, as another computational complexity measure for the algorithm.

Subsequently, the closed-loop MPC times are averaged over the number of optimal control problems solved, and are compared to the timings of Algorithm 5 in Figure 3.7b. It is clear that in this case the proposed approach is not as competitive as MPC. However, it should be mentioned that the cases depicted in Figure 3.7 are two extremes. Figure 3.7a assumes that no noise is present, while Figure 3.7b implicitly assumes that Algorithm 5 is cold-started at every subsequent problem, making no use of the prior information gained from the convergence to the optimal multiplier sequence $\boldsymbol{\lambda}^{\infty}$.

In view of the facts that (i) the above packages run optimized C code and (ii) the employed algorithms differ quite significantly from our proximal method, we devise a simulation where we use *the exact same C++ code*[1] *that we have developed for Algorithm 5, but rather in order to solve a series of finite horizon MPC problems. Ultimately, we apply the dual proximal gradient method to do MPC.* In this way we control for most of the irrelevant factors and focus on the comparison between CLQR and MPC. Once again, we consider the toy example and sample randomly 200 initial conditions from the set of the 750 ones utilized above. We inject disturbances with magnitude up to 3% of the predicted initial state and solve in closed loop until the initial state enters the terminal

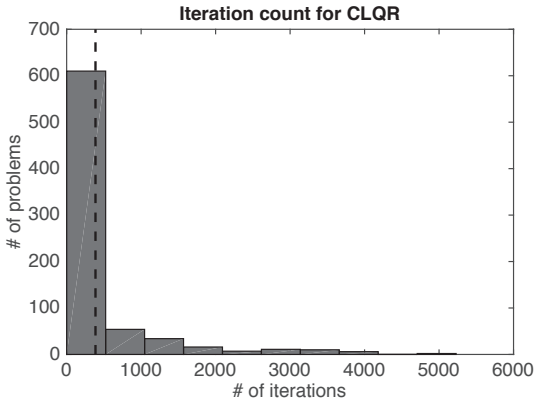
[1]The code is publicly available on https://github.com/stathopog/InfHorCLQR.



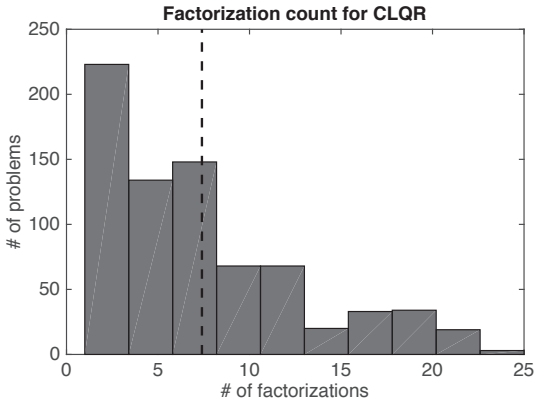
(a) One solve (nominal case). **(b)** Average solve times for MPC and CLQR.

Figure 3.7: Solve times of the CLQR solver versus CPLEX, MOSEK and QPgen for the toy system, sampled from 750 initial conditions. In **(a)**, the CLQR generally requires less time than solving a series of finite horizon optimal control problems, leading to the same optimal solution. Only the solver QPgen clearly outperforms our proposed approach, which makes use of the same, in principle, splitting method, enhanced by several add-on's that enable speedup. In **(b)** the three solvers are faster. The large variance of the CLQR approach in comparison to the others is a typical characteristic of first-order methods, and can be significantly reduced with appropriately conditioning of the problem.

invariant set. We compare CLQR to MPC with horizon length $T = \lceil \frac{2}{3} T^{\infty} \rceil$ and terminal set constraint, as well as with $T = 2T^{\infty}$, without terminal set constraint, exactly as we did before. Regarding the algorithmic parameters, a is set to 5 and the stopping criterion is set to a very low tolerance ($\|\boldsymbol{\lambda}^k - \hat{\boldsymbol{\lambda}}^{k+1}\| / \|\boldsymbol{\lambda}^k\| \leq 10^{-12}$) so as to eliminate any numerical discrepancies resulting from early termination. We solve the Lagrangian minimization (Step 3 in Algorithm 5) by means of a Riccati recursion. Note that the complexity is similar to solving a linear system since only the affine parts of the Lagrangian function vary per iteration, allowing us to precompute the Riccati matrix and the feedback gain. For the same reasons, the same complexity holds when solving the MPC problem (see, *e.g.,* [FJ13b] for the time-varying Riccati recursion). We are interested in four metrics: The number of infeasible problems when doing MPC (indicator of the volume of the region of attraction), the number of iterations, the execution times and the quality of the controller (evaluated via the objective function value that the two schemes achieve). The results are summarized in Table 3.1. Out of the 200 sampled initial conditions, 19 resulted in infeasible closed-loop problems due to the additive state disturbance. The short-viewed MPC policy (3^{d} column) resulted in 69 infeasible problems. The times achieved are significantly higher than the CLQR times. The deterioration of the objective function value is negligible. When the horizon increases (4^{th} column), the average and median times go down, being comparable, but still not as good as the ones achieved with CLQR. The reduced iteration count is a result of the warm-starting policy which is more efficient when applying CLQR instead of MPC, in the sense that, firstly, the multipliers are initialized at their optimal values, and secondly, that the horizon (hitting time T^{∞}) shrinks as time progresses, essentially resulting in smaller QPs.



(a) Histogram for the number of iterations required over 750 problems.



(b) Number of factorizations required over 750 problems.

Figure 3.8: Iteration and factorization counts for the toy system. It can be seen in **(a)** that more than 80% of the problems needed less than 400 iterations in order to converge, while the maximum number observed is around 5000 iterations. The mean value of factorizations in **(b)** is seven while no problem instance needed more than 25 factorizations.

Table 3.1: Closed loop comparison between CLQR and MPC

	CLQR	**MPC** $\lceil \frac{2}{3} T^{\infty} \rceil$ **w/ T.S.**	**MPC** $2T^{\infty}$ **w/o T.S.**
# **Infeasible Problems**	19	69	19
Times (ms)			
Average	124.8	169.9	92.1
Median	30.6	64.6	50.3
Iterations			
Average	1118.0	1537.8	1461.9
Median	290.0	591.0	317.0
# **Average Objective Increase**	-	0.04%	-

3.9 Conclusion

In this chapter, we presented an algorithmic scheme capable of solving the constrained linear quadratic regulator problem in real time. The algorithm is an accelerated version of the proximal gradient algorithm and belongs to the wider family of forward-backward splitting schemes. The approach is to write the problem in its condensed form and dualize, which leads to the minimization of an infinite-dimensional quadratic functional subject to non-positivity constraints. The resulting infinite dimensional problem can be tackled in finite dimensions by observing that the dual sequence has always only finitely many non-zero elements. The proposed algorithm makes no use of terminal invariant sets and provably converges to the optimal solution of the infinite-horizon

problem. Regarding the implementation aspects, the algorithm can be highly competitive since it enjoys the convergence properties of optimal first-order methods whose computational per iteration overhead is small. In addition, it requires minimal a priori information since the most crucial quantities are computed online, and there are no unreasonable or conservative assumptions on the problem's structure.

There are mainly two ways to extend the constrained infinite-horizon regulation result: Computational and theoretical enhancements. From a computational viewpoint, the number of iterations needed for convergence can be quite large for very unstable (large) systems, or for ill-conditioned problem data. To this end, preconditioning of the problem is necessary in order for the method to be practical. The several recent advancements regarding proximal gradient methods [GB14b; GB15; STP17] could be applicable to our approach, with the possible difficulty of adapting these results to infinite dimensional spaces.

Theoretical extensions could, again, be developed in two directions. With regard to the 'type' of the optimal control problem, an obvious extension is the setpoint-tracking case, possibly a soft-constrained optimal control formulation, as well as the interesting possibility of application to stochastic infinite horizon problems. Regarding the type of constraint sets that our approach can tackle, going beyond the polytopic case would be useful in several settings. These extensions are discussed in more detail in Chapter 6.

3.10 Appendices

3.10.1 Required Operator Theory

The subsequent results hold for general real Hilbert spaces, including the special case of l_w^2 we consider. We write variables in normal font, and we use the bold font to describe the infinite-dimensional variables we are manipulating in our problem description.

Definition 15. *The l^2-weighted (or l_w^2) real Hilbert space $\mathcal{H}$ is defined by*

$$\mathcal{H} = \left\{ \boldsymbol{z} = \{z_i\} \ : \ \sum_{i=0}^{\infty} \|z_i\|_2^2 w^i < \infty \right\}, \quad w > 0 \ .$$

Definition 16. *A linear operator (mapping) $F \colon \boldsymbol{\mathcal{H}_1} \to \boldsymbol{\mathcal{H}_2}$ between two Hilbert spaces is said to be bounded if the operator norm $\|F\|$ of F, defined as*

$$\|F\| \coloneqq \sup_{\|x\|_{\boldsymbol{\mathcal{H}_1}}=1} \|Fx\|_{\boldsymbol{\mathcal{H}_2}} \ ,$$

satisfies $\|F\| < \infty$. The set of bounded operators between two Hilbert spaces $\boldsymbol{\mathcal{H}_1}$ and $\boldsymbol{\mathcal{H}_2}$ is denoted as $\mathcal{B}(\boldsymbol{\mathcal{H}_1}, \boldsymbol{\mathcal{H}_2})$.

Theorem 6. *Let $\boldsymbol{\mathcal{H}_1}, \boldsymbol{\mathcal{H}_2}$ be real Hilbert spaces and $F \in \mathcal{B}(\boldsymbol{\mathcal{H}_1}, \boldsymbol{\mathcal{H}_2})$. The adjoint of F is the unique operator $F^\star \in \mathcal{B}(\boldsymbol{\mathcal{H}_2}, \boldsymbol{\mathcal{H}_1})$ that satisfies*

$$\langle Fx, y \rangle = \langle x, F^\star y \rangle \quad \forall x \in \boldsymbol{\mathcal{H}_1}, \ \forall y \in \boldsymbol{\mathcal{H}_2} \ .$$

Moreover, $\|F\| = \|F^\star\|$.

Subsequently, we introduce the notions of weak and strong convergence.

Definition 17. *Let* $\mathcal{H}$ *be a Hilbert space. We say that* $\{x^k\}$ *converges weakly to* x *if* $\forall y \in \mathcal{H}$ $\langle y, x^k\rangle \xrightarrow{k\to\infty} \langle y, x\rangle$. *We denote weak convergence as* $x^k \rightharpoonup x$.

Definition 18. *Let* $(x^k)_{k\in\mathbb{N}}$ *be a sequence in* $\mathcal{H}$. *Then* $\{x^k\}$ *converges strongly to* x *if* $\|x^k - x\| \xrightarrow{k\to\infty} 0$. *We denote strong convergence as* $x^k \to x$.

Definition 19. *An operator* $F : \mathcal{H} \to \mathcal{H}$ *is* positive-definite *if it is bounded,* $F = F^\star$ *and* $\langle Fx, x\rangle \geq \alpha\|x\|_{\mathcal{H}}$ *for some* $\alpha > 0$, *for all* $x \in \mathcal{H}$.

We recall that a positive definite operator F is invertible and the inverse operator F^{-1} is bounded.

3.10.2 Boundedness of several operators

Lemma 7. *The operators* $\boldsymbol{C}$ *and* $\boldsymbol{W}$ *are bounded.*

Proof: Boundedness of $\boldsymbol{W}$ is trivial since it is a diagonal operator with non-increasing elements on the diagonal.

The operator $\boldsymbol{C}$ can be expressed as the following sum:

$$\boldsymbol{C} = \begin{bmatrix} C_u & 0 & \cdots \\ 0 & 0 & \cdots \\ 0 & C_u & \cdots \\ 0 & 0 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 0 & \cdots \\ C_xB & 0 & \cdots \\ 0 & 0 & \cdots \\ C_xAB & C_xB & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}}_{\boldsymbol{C_x}},$$

and by using the triangle inequality, we have

$$\|\boldsymbol{C}\| \leq \sigma_{\max}(C_u) + \|\boldsymbol{C_x}\|. \tag{3.18}$$

We thus have to show that $\boldsymbol{y} = \boldsymbol{C_x u}$ is bounded, *i.e.,*

$$\sup_{\|\boldsymbol{u}\|_{\mathcal{H}_{\boldsymbol{u}}}=1} \|\boldsymbol{y}\|_{\mathcal{H}_{\boldsymbol{\lambda}}} < \infty.$$

In order not to carry the zero rows of $\boldsymbol{C_x}$, we define $\bar{\boldsymbol{y}} \in \bar{\mathcal{H}_{\boldsymbol{\lambda}}}$ by dropping the zero elements of $\boldsymbol{y}$. This infinite-dimensional vector consists of the elements $\bar{y}_i = [\boldsymbol{C_x}]_i \boldsymbol{u} = \sum_{j=0}^{i-1} C_x A^{i-j-1} B u_j \in \mathbb{R}^{p_x}$. Note that

$$\sup_{\|\boldsymbol{u}\|_{\mathcal{H}_{\boldsymbol{u}}}=1} \|\bar{\boldsymbol{y}}\|_{\bar{\mathcal{H}_{\boldsymbol{\lambda}}}} = \sup_{\|\boldsymbol{u}\|_{\mathcal{H}_{\boldsymbol{u}}}=1} \|\boldsymbol{y}\|_{\mathcal{H}_{\boldsymbol{\lambda}}} \ .$$

Focusing on the operator of interest, we have that

$$\|\bar{\boldsymbol{y}}\|_{\bar{\mathcal{H}}_{\boldsymbol{\lambda}}} = \sqrt{\sum_{i=1}^{\infty} \|\bar{y}_i\|_2^2 w^i} = \sqrt{\sum_{i=1}^{\infty} w^i \| \sum_{j=0}^{i-1} C_x A^{i-j-1} B u_j \|_2^2} \ . \tag{3.19}$$

Then

$$\begin{aligned} \|\bar{\boldsymbol{y}}\|_{\bar{\mathcal{H}}_{\boldsymbol{\lambda}}}^2 &= \sum_{i=1}^{\infty} w^i \| \sum_{j=0}^{i-1} C_x A^{i-j-1} B u_j \|_2^2 \\ &= \sum_{i=1}^{\infty} \| \sum_{j=0}^{i-1} C_x (A\hat{w})^{i-j-1} B u_j \hat{w}^{j+1} \|_2^2 \\ &= \hat{w}^2 \sum_{i=1}^{\infty} \| \sum_{j=0}^{i-1} C_x \hat{A}^{i-j-1} B \hat{u}_j \|_2^2 \ , \end{aligned}$$

where we introduced $\hat{w} = w^{1/2}$, $\hat{u}_j = u_j \hat{w}^j$ with $\hat{\boldsymbol{u}} \in l^2$ and $\hat{A} = A\hat{w}$. Observing the above expression, one can identify that $\sum_{j=0}^{i-1} C_x \hat{A}^{i-j-1} B \hat{u}_j$ is the *convolution sum* of the impulse response of the system

$$\Sigma := \left(\begin{array}{c|c} \hat{A} & B \\ \hline C_x & 0 \end{array} \right)$$

with an input $\hat{u}$. More specifically, borrowing the notation from [Ant98] we denote the impulse response of Σ as $h_{\Sigma,i} = C_x \hat{A}^{i-1} B s_i$, where $s_i = 1, \ i \geq 0$ is the unit step function. Then the convolution operator is defined as the linear map $S_\Sigma \colon \hat{\boldsymbol{u}} \to \bar{\boldsymbol{y}}$ with $\bar{y}_i = (h_\Sigma * \hat{\boldsymbol{u}})_i = \sum_{j=0}^{i-1} h_{\Sigma,i-j} \hat{u}_j = (S_\Sigma \hat{\boldsymbol{u}})_i$. Thus we have that

$$\begin{aligned} &\|\bar{\boldsymbol{y}}\|_{\bar{\mathcal{H}}_{\boldsymbol{\lambda}}}^2 = \hat{w}^2 \sum_{i=1}^{\infty} \| \left(S_\Sigma \hat{\boldsymbol{u}}\right)_i \|_2^2 = \hat{w}^2 \|S_\Sigma \hat{\boldsymbol{u}}\|_2^2 \Rightarrow \\ &\|\bar{\boldsymbol{y}}\|_{\bar{\mathcal{H}}_{\boldsymbol{\lambda}}} = \hat{w} \|S_\Sigma \hat{\boldsymbol{u}}\|_2 \leq \hat{w} \sup_{\|\hat{\boldsymbol{u}}\|_2 \leq 1} \|S_\Sigma \hat{\boldsymbol{u}}\|_2 \ . \end{aligned}$$

From the definition of the induced 2-norm of Σ, denoted here as $\|\Sigma\|_2$, we have that $\sup \|S_\Sigma \hat{\boldsymbol{u}}\|_2 = \|\Sigma\|_2 \|\hat{\boldsymbol{u}}\|_2$ and by assuming (without loss of generality) that $\|\hat{\boldsymbol{u}}\|_2 = \|\boldsymbol{u}\|_{\mathcal{H}_{\boldsymbol{u}}} = 1$, we end up having that $\|\bar{\boldsymbol{y}}\|_{\bar{\mathcal{H}}_{\boldsymbol{\lambda}}} \leq \hat{w} \|\Sigma\|_2$. Finally, the operator $\boldsymbol{C_x}$ is bounded by the $\mathcal{H}^\infty$ norm of the transfer matrix $H_\Sigma(z)$, or $\|\bar{\boldsymbol{y}}\|_{\bar{\mathcal{H}}_{\boldsymbol{\lambda}}} \leq \hat{w} \mathcal{H}^\infty(\Sigma)$. Subsequently, we have from (3.18) that the operator $\boldsymbol{C}$ is bounded by $\sigma_{\max}(C_u) + \hat{w} \mathcal{H}^\infty(\Sigma)$. □

Remark 9. *Following the discussion from Section 3.1, the weight $\hat{w}$ can be chosen to render any unstable system stable by shrinking the eigenvalues of the matrix A ($\hat{A} = \hat{w}A$).*

Lemma 8. *The operator $\boldsymbol{H}^{-1}$ is bounded and $\|\boldsymbol{H}\| \leq 1/\lambda_{\min}(R)$.*

Proof: The operator $\boldsymbol{H}$ is given by $\boldsymbol{H} = \boldsymbol{B}^\star \boldsymbol{Q} \boldsymbol{B} + \boldsymbol{R}$; see (3.2). Since $\boldsymbol{B}^\star \boldsymbol{Q} \boldsymbol{B}$ is positive semidefinite (i.e., $\langle \boldsymbol{B}^\star \boldsymbol{Q} \boldsymbol{B} \boldsymbol{u}, \boldsymbol{u} \rangle \geq 0$ for all $\boldsymbol{u} \in \mathcal{H}_{\boldsymbol{u}}$) and $\boldsymbol{R}$ is positive definite according to Definition 19, we

conclude that $\boldsymbol{H}$ is positive definite and hence has a bounded inverse.

In order to compute the bound for $\boldsymbol{H}^{-1}$ we observe that $\|(\boldsymbol{B}^\star \boldsymbol{Q} \boldsymbol{B} + \boldsymbol{R})^{-1}\| \leq \|\boldsymbol{R}^{-1}\| \leq 1/\lambda_{\min}(R)$, which concludes the proof. □

3.10.3 Backtracking stepsize rule

In this appendix we briefly revise the backtracking stepsize rule that allows for local estimates of the curvature of $h^\star$ and show that points (i), (iii) and (iv) of Theorem 1 also hold in this case. The arguments are in line with [BT09].

We denote as f the smooth and g the nonsmooth convex functions of interest. Note that in our case $f = h^\star$ and $g = \delta_-$. For any $L > 0$ consider the quadratic approximation of $F(\boldsymbol{\lambda}) = f(\boldsymbol{\lambda}) + g(\boldsymbol{\lambda})$ at a point $\boldsymbol{y}$:

$$Q_L(\boldsymbol{\lambda}, \boldsymbol{y}) := f(\boldsymbol{y}) + \langle \boldsymbol{\lambda} - \boldsymbol{y}, \nabla f(\boldsymbol{y}) \rangle + \frac{L}{2} \|\boldsymbol{\lambda} - \boldsymbol{y}\|^2 + g(\boldsymbol{\lambda}) \ . \tag{3.20}$$

We also define the unique minimizer parametrized by the point $\boldsymbol{y}$ as

$$\begin{aligned} p_L(\boldsymbol{y}) &:= \arg\min_{\boldsymbol{\lambda}} \{Q_L(\boldsymbol{\lambda}, \boldsymbol{y})\} && (3.21) \\ &= \arg\min_{\boldsymbol{\lambda}} \left\{ g(\boldsymbol{\lambda}) + \frac{L}{2} \|\boldsymbol{\lambda} - \left(\boldsymbol{y} - \frac{1}{L} \nabla f(\boldsymbol{y}) \right) \|^2 \right\} && (3.22) \\ &= \mathbf{prox}_g \left(\boldsymbol{y} - \frac{1}{L} \nabla f(\boldsymbol{y}) \right) \ , && (3.23) \end{aligned}$$

which is the basic step of Algorithm 4, *i.e.,* Step 3. The following holds:

Lemma 9 (Lemma 2.3 [BT09]). *Let $\boldsymbol{y} \in \mathcal{H}_{\boldsymbol{\lambda}}$ and $L > 0$ be such that*

$$F(p_L(\boldsymbol{y})) \leq Q_L(p_L(\boldsymbol{y}), \boldsymbol{y}) \ .$$

Then for any $\boldsymbol{\lambda} \in \mathcal{H}_{\boldsymbol{\lambda}}$,

$$F(\boldsymbol{\lambda}) - F(p_L(\boldsymbol{y})) \geq \frac{L}{2} \|p_L(\boldsymbol{y}) - \boldsymbol{y}\|^2 + L \langle \boldsymbol{y} - \boldsymbol{\lambda}, p_L(\boldsymbol{y}) - \boldsymbol{y} \rangle \ .$$

The backtracking procedure as described in [BT09] is as follows:

Algorithm 6 Backtracking for stepsize computation

0: Take $L^0 > 0$, some $\eta > 1$, and $\boldsymbol{\lambda}^0 \in \mathcal{H}_{\boldsymbol{\lambda}}$.
repeat
1: Find the smallest nonnegative integer i^k such that with $\bar{L} = \eta^{i^k} L^{k-1}$
until $F(p_{\bar{L}}(\boldsymbol{y}^k)) \leq Q_{\bar{L}}(p_{\bar{L}}(\boldsymbol{y}), \boldsymbol{y})$

Note that for any $L^k = \bar{L}$ generated by Algorithm 6 Lemma 9 holds.

We also have the following instrumental Lemma from [CD15]:

Lemma 10 (Lemma 1 [CD15]). *Let $L \geq L(f)$, $\boldsymbol{\lambda}, \boldsymbol{y} \in \mathcal{H}_{\boldsymbol{\lambda}}$ and $p_L(\boldsymbol{y}) := \mathbf{prox}_g(\boldsymbol{y} - \frac{1}{L}\nabla f(\boldsymbol{y}))$. Then for all $\boldsymbol{\lambda}$*

$$F(p_L(\boldsymbol{y})) + \frac{L}{2}\|p_L(\boldsymbol{y}) - \boldsymbol{\lambda}\|^2 \leq F(\boldsymbol{y}) + \frac{L}{2}\|\boldsymbol{\lambda} - \boldsymbol{y}\|^2 \ .$$

Supposing that Lemma 10 holds, convergence of the function values with rate $1/k^2$ can be proven under no further assumptions using Theorem 2 and Corollary 1 in [CD15]. We are going to show that all stepsizes $\gamma^k = 1/L^k$ generated from Algorithm 6 satisfy Lemma 10.

Lemma 11. *Consider $F = f + g$, with $f = h^\star$ and $g = \delta_-$ as defined in Section 3.4. The iterates $\boldsymbol{\lambda}^k$ generated from Algorithm 4 with a backtracking stepsize rule generated from Algorithm 6 satisfy:*

$$F(\boldsymbol{\lambda}^k) - F(\boldsymbol{\lambda}^\infty) \leq \frac{a^2\bar{L}}{2(k+a-1)^2}\|\boldsymbol{\lambda}^0 - \boldsymbol{\lambda}^\infty\|^2 \ .$$

Proof: All local Lipschitz estimates $\bar{L}$ generated from Algorithm 6 satisfy $F(p_{\bar{L}}(\boldsymbol{y}^k)) \leq Q_{\bar{L}}(p_{\bar{L}}(\boldsymbol{y}), \boldsymbol{y})$ and, consequently, from Lemma 9,

$$F(\boldsymbol{\lambda}) - F(p_{\bar{L}}(\boldsymbol{y})) \geq \frac{\bar{L}}{2}\|p_{\bar{L}}(\boldsymbol{y}) - \boldsymbol{y}\|^2 + \bar{L}\langle \boldsymbol{y} - \boldsymbol{\lambda}, p_{\bar{L}}(\boldsymbol{y}) - \boldsymbol{y}\rangle \ ,$$

or

$$\frac{2}{\bar{L}}\left(F(\boldsymbol{\lambda}) - F(p_{\bar{L}}(\boldsymbol{y}))\right) \geq \|p_{\bar{L}}(\boldsymbol{y}) - \boldsymbol{y}\|^2 + 2\langle \boldsymbol{y} - \boldsymbol{\lambda}, p_{\bar{L}}(\boldsymbol{y}) - \boldsymbol{y}\rangle \ .$$

It follows from the Pythagorean theorem that

$$\|b - a\|^2 + 2\langle b - a, a - c\rangle = \|b - c\|^2 - \|a - c\|^2 \ ,$$

and hence

$$\frac{2}{\bar{L}}\left(F(\boldsymbol{\lambda}) - F(p_{\bar{L}}(\boldsymbol{y}))\right) \geq \|p_{\bar{L}}(\boldsymbol{y}) - \boldsymbol{\lambda}\|^2 - \|\boldsymbol{y} - \boldsymbol{\lambda}\|^2 \ ,$$

from which Lemma 10 follows. Theorem 2 and Corollary 1 of [CD15] then lead to the desired result. □

Chapter 4

Inertial Parallel and Asynchronous Forward-Backward Iteration for Distributed Convex Optimization

4.1 Introduction

The inherent parallelization potential of operator splitting schemes has spurred a significant amount of research in asynchronous implementations. Asynchronous parallel methods have been mostly motivated from memory allocation applications, when, *e.g.,* a vector is stored in the shared memory space of a multicore computer and can be accessed and altered by the cores in an intermittent manner [LW15; PXYY16].

In this chapter, we focus on another application area that motivates asynchronicity, namely the existence of an inhomogeneous mixture of agents, where their local updates need not occur at a common rate. This type of problems appears in a setting different from the machine learning ones, *i.e.,* in *multi-agent distributed optimization* problems, usually in the presence of a *global coordinator*. As an example, in a smart grid setting with distributed resources (agents) and a central operator (coordinator), the local update of a particular agent is the solution to an optimization problem of different complexity than other local subproblems of different agents. In addition, the agents' updates need not occur uniformly, or as a matter of fact, need not draw from any stationary distribution since intermittent failures and delays occur. This would require that the computations of different subproblems are initiated at different time instances and that the agents communicate their solutions to the coordinator in arbitrary sequences. Asynchronous schemes like the one described above pose several challenges in terms of proving convergence in comparison to their synchronous counterparts (see [BT89; Wri15] for interesting overviews).

Our work brings together acceleration techniques with asynchronous implementations of a rather wide family of operator splitting schemes, this of *forward-backward splitting methods (FBS) [BC11, Chapter 25]*. More specifically, we devise an asynchronous iteration in which the coordinates update with varying, arbitrary frequencies and, under some common assumptions, we show that the distance to the set of fixed points of an inertial version of this asynchronous FBS iteration will converge *linearly* to zero provided that all the coordinates are visited at least once in a given (bounded) time

interval.

The outline of the chapter is as follows: In Section 4.2 we make a short reference to the main theoretical tools that are going to be used in this chapter, along with existing works regarding relaxed and/or inertial fixed-point iterations. The problem of interest is first formulated and explained in Section 4.3, where the contributions of this work are also outlined. Section 4.4 illustrates a sketch of the convergence proof of the proposed scheme, while the detailed steps are presented in the Appendices 4.8. In Section 4.5 we draw the connections bwtween our scheme and existing algorithms and how it gives rise to new versions of the latter. Finally, Section 4.6 illustrates the performance of the method in comparison to its regular counterpart for a load sharing problem in the context of a smart distribution grid.

4.2 Related work

The preliminary material for this chapter is covered in Section 2.2 of Chapter 2. We repeat below some of the basic concepts presented there for ease of access and completeness.

Our purpose is to combine the heavy ball iteration of Polyak with a relaxed fixed-point iteration in an asynchronous context and achieve faster practical convergence. Polyak's heavy ball method (inertial acceleration), as described in Section 2.2 is a modification of the gradient descent iteration that generates a sequence of iterates $\{x_k\}$ that minimize a differentiable, convex function f

$$x_{k+1} = x_k - \gamma \nabla f(x_k) + \beta(x_k - x_{k-1}) \ , \tag{4.1}$$

with γ being an (admissible) stepsize and $\beta \in (0, 1)$. The method has been generalized in the context of finding a zero of a maximal monotone operator S.

$$\text{find } x_* \in \mathcal{H} \ \text{ such that } \ 0 \in Sx_* \ .$$

In [AA01], the authors proposed an Inertial-Prox algorithm that generalizes (4.1) to

$$x_{k+1} = J_{\gamma_k S}(x_k + \beta_k(x_k - x_{k-1})) \ , \tag{4.2}$$

where $J_{\gamma_k S}$ is the resolvent of S. Iteration (4.2) generalizes the proximal point algorithm and finds a zero of a maximal monotone operator S by making use of the momentum term. In [MO03], the authors extended the inertial scheme (4.2) to find a zero of the sum of two maximal monotone operators.

The combination of inertia and relaxation has been proposed in [Alv04], where the zero of a maximal monotone operator S is recovered by means of the iteration

$$x_{k+1} = y_k + \eta_k(J_{\gamma_k S} y_k - y_k) \ ,$$

where $y_k = x_k + \beta_k(x_k - x_{k-1})$. The iteration converges weakly to a fixed point of $J_{\gamma_k S}$ in a Hilbert space setting.

The works [AA01; MO03; Alv04] are put under a common framework in [Mai08], where the

author develops convergence theorems for a generic inertial KM-type iteration of the form

$$x_{k+1} = y_k + \eta_k(T_k y_k - y_k) \ ,$$

with $y_k = x_k + \beta_k(x_k - x_{k-1})$. Convergence is proven under different choices for the parameter sequences $\{\eta_k\}$, $\{\beta_k\}$ as well as the operator sequence $\{T_k\}$. Finally, in the recent work [IH16] the authors employ relaxed and inertial schemes to accelerate the KM iteration by means of algorithms that auto-tune the involved parameters.

4.3 Problem description

4.3.1 Asynchronous updates

The proposed setting involves N agents, each one assigned to update one (block of) coordinate(s) of x, *i.e.,* $x = (x[1], \dots, x[N]) \in \mathcal{H}$, and $x[i] \in \mathcal{H}_i$. The agents seek convergence to a fixed point of a nonexpansive operator T. One way to achieve this is to perform block-coordinate updates of the KM iteration

$$z_{k+1} = x_k + \eta_k(Tx_k - x_k) \ , \tag{4.3}$$

where $T : \mathcal{H} \mapsto \mathcal{H}$ is a nonexpansive operator and $\eta_k \in [0, 1]$, $\sum \eta_k(1 - \eta_k) = +\infty$ is a relaxation constant.

Such a scheme has been proposed and analyzed in [PXYY16]. The iteration reads

$$x_{k+1}[i] = x_k[i] - \eta_k(Sx^i_{\text{read}})[i] \ , \tag{4.4}$$

where $S = I - T$, hence the set of fixed points of T is the set of zeros of S, *i.e.,* $\operatorname{fix} T = \operatorname{zer} S$.

Iteration (4.4) assumes *the existence of a global coordinator, associated with a global clock. All agents update continuously and in parallel, while the global clock updates the subscript k every time that an agent updates.* The variable x^i_{read} *represents the state of the vector x as it existed at the coordinator when the agent that is about to update (i) requested it (a 'read' operation).* Each update involves the most recent state of x, denoted by x_k, and the result of the operator S acting on an outdated version x^i_{read}. The distinction between x_k and x^i_{read} is important, since, on the coordinator's level, several components of x_k have possibly been altered since the time instant that x^i_{read} was read. Every global clock count k is uniquely associated to an updated i^{th} group of coordinates. In this way, only the i^{th} block of rows of the operator S contributes to the next update of x, and only the $x[i] \in \mathcal{H}_i$ block is updated.

Our goal is to propose an accelerated version of (4.4) in order to achieve better practical performance without increasing the computational complexity of the iteration.

4.3.2 An asynchronous inertial forward-backward iteration

We propose an asynchronous inertial KM iteration scheme for finding a zero of S. We confine our interest to not just any KM iteration, but we rather assume that the operator T can be written as the composition of two operators $T_A : \mathcal{H} \mapsto \mathcal{H}$ and $T_B : \mathcal{H} \mapsto \mathcal{H}$, the properties of which will

be analyzed in the course of this section. In addition, we assume that the operator T_A is separable into N components, *i.e.,* $T_A = (T_{A_1}, \ldots, T_{A_N})$, $T_{A_i} : \mathcal{H}_i \mapsto \mathcal{H}_i$.

The scheme comprises $N + 1$ main blocks, one associated to the coordinator and N associated to the agents. The operators T_{A_i} are private to the agents, while T_B is owned by the coordinator. Before proceeding to the algorithm, we introduce the quantities associated to the coordinator and the agents. Let us start by denoting all variables stored at the coordinator by 'x' and all variables stored at the agents by 'y'.

- Coordinator
  - x - the current value of the (global) optimization variable
  - x^i_{write} - the value of x at the time of receipt of value from agent i
  - x^i_{read} - the value of x at the time of transmission to agent i
  - z^i - the last value received from agent i ($z^i = T_{A_i}\left(y_B + \beta(y_{\text{write}} - y^{\text{prev}}_{\text{write}})\right)$)
- The following variables are local to agent i
  - $y_{\text{write}} = x^i_{\text{write}}[i]$ - the value of $x[i]$ after updating x with the latest z^i
  - $y^{\text{prev}}_{\text{write}}$ - the value of $x[i]$ before y_{write}
  - $y_B = (T_B x^i_{\text{read}})[i]$ - quantity computed by the coordinator and transmitted at the same time as y_{write}

Agent i essentially waits to receive the quantities y_B and y_{write} from the coordinator. Once received, the privately owned operator T_{A_i} is applied to the expression $y_B + \beta(y_{\text{write}} - y^{\text{prev}}_{\text{write}})$ and the result, *i.e.,* z^i, is transmitted back to the coordinator, which, in turn, uses it in order to update the i^{th} component of the global variable x.

The algorithmic scheme can be described by two interacting and distinct blocks, one refering to an agent and one to the coordinator.

Algorithm 7 Agent

wait until
Receive y_{write}, y_B from coordinator
Compute:

$$z^i = T_{A_i}\left(y_B + \beta(y_{\text{write}} - y^{\text{prev}}_{\text{write}})\right) \tag{4.5}$$

Transmit z^i to coordinator
$y^{\text{prev}}_{\text{write}} \leftarrow y_{\text{write}}$
end

Algorithm 8 Coordinator

Write Thread
Initialize $\mathcal{W} = \varnothing$, $\mathcal{R} = \varnothing$, $k = 0$
repeat
  Receive z^i from agent i
  $\mathcal{W} \leftarrow \mathcal{W} \cup \{i\}$
until stopping condition holds

Compute Thread
repeat
  Choose $i \in \mathcal{W}$ or **block until** $\mathcal{W} \neq \varnothing$
  $\mathcal{W} \leftarrow \mathcal{W} \setminus \{i\}$
  $z[i] \leftarrow z^i$
  Compute:

$$x_{k+1} = (1-\eta)x_k + \eta z \qquad (4.6)$$

  $x^i_{\text{write}} \leftarrow x_k$
  $\mathcal{R} \leftarrow \mathcal{R} \cup \{i\}$
  $k \leftarrow k+1$
until stopping condition holds

Read Thread
repeat
  Choose $i \in \mathcal{R}$ or **block until** $\mathcal{R} \neq \varnothing$
  $\mathcal{R} \leftarrow \mathcal{R} \setminus \{i\}$
  $x^i_{\text{read}} \leftarrow x_k$

  Transmit $x^i_{\text{write}}[i]$, $(T_B x^i_{\text{read}})[i]$

until stopping condition holds

We make the following observations:

- Algorithms 7 and 8 are executed continuously and in parallel. There is one algorithmic block described by Algorithm 7 per agent i, $i = 1, \ldots, N$, and these N blocks execute in parallel. Each block is activated *upon receival* of the required info from the coordinator.
- There are three threads of control in Algorithm 8, namely a *Write thread*, a *Compute thread* and a *Read thread*. The threads are *concurrent* and their execution is determined by two buffers, the *read* buffer denoted by $\mathcal{R}$ and the *write* buffer denoted by $\mathcal{W}$.
- Whenever the coordinator receives an update from an agent, $\mathcal{W}$ is updated. Receivals contribute, therefore, in *filling-up* the write buffer. The coordinator eventually decides to pull an agent from the buffer and use its corresponding value to update $x_{k+1}[i]$ with $z[i]$ (equation (4.6)), while the rest of the coordinates are updated based on the previous values of $z[j]$, $j \neq i$. Once the update has occured, the index of the corresponding agent is removed from $\mathcal{W}$ and added to $\mathcal{R}$, signaling that the agent is ready to 'listen' from the coordinator. Similarly, whenever the coordinator decides to transmit to an agent (Read thread), its index is removed from $\mathcal{R}$. In this way, the buffers control the execution of Algorithm 8, which follows a *producer-consumer pattern*.
- Note that whenever $\mathcal{W}$ is emptied, the Compute thread is blocked until at least one index is added to its stack. The same holds for $\mathcal{R}$ and the Read thread.
- As indicated by Assumption 5 below, each agent has to contribute in updating (4.6) at least once every τ time epochs. Consequently, the buffers cannot remain empty for longer than τ.
- The relaxation parameter η and the inertia constant β that appear in (4.6) and (4.5), respectively, will be restricted within intervals in the subsequent sections so as to ensure convergence of the algorithm.
- The workhorse behind the conceptual scheme derived above is essentialy the asynchronous fixed-point iteration (4.6) that is based on the agents' updates (4.5).

- The coordinator's variables $x, x^i_{\text{write}}, x^i_{\text{read}}$ and z are containers that get updated from the agents and lie in $\mathcal{H}$. The local variables $y_{\text{write}}, y^{\text{prev}}_{\text{write}}, y_B$ and z^i lie in $\mathcal{H}_i$.

Figure 4.1 demonstrates graphically the information flow.

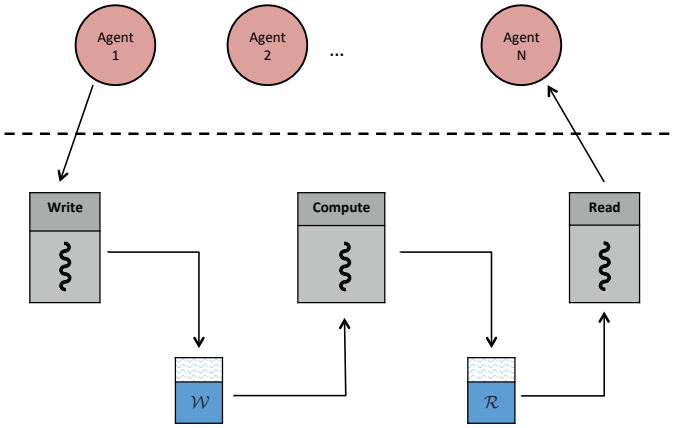


Figure 4.1: Buffer $\mathcal{W}$ is filled by the Write thread and emptied by the Compute thread. Similarly, $\mathcal{R}$ is filled by the Compute thread and emptied by the Read thread. The scheme executes continuously and asynchronously, both at the coordinator level (concurrent threads) and at the interface between the agents and the coordinator.

We make the following standing assumptions:

Assumption 2. *The operator T_A is nonexpansive.*

Assumption 3. *The operator $T_B = I - \gamma B$, $\gamma > 0$, and the operator B is $1/L$-cocoercive.*

Corollary 1. *The operator $S = I - T$ is $1/2$-cocoercive.*

The proof can be found in Appendix 4.8.6.

Assumption 4. *The operator $S = I - T$ is quasi-ν-strongly monotone for some $\nu > 0$.*

Iteration (4.6) generalizes the inertial proximal iteration in [OBP15], with T_A replacing the proximal operator and T_B the operator $I - \gamma\nabla f$. Assumption 4 can be met for a relatively wide class of operators T_A and T_B. One such instance is derived in Appendix 4.8.6, where μ-strong monotonicity of the operator B is assumed in order for the property to hold. In the case of the proximal gradient method with $T_A = \mathbf{prox}_{\gamma g}$ and $T_B = I - \gamma\nabla f$, this assumption would translate to strong convexity of f.

Finally, the following assumption regards the frequency of the updates:

Assumption 5. *Each agent 'writes' to the coordinator state at least once every τ time epochs.*

Assumption 5 categorizes our scheme with the *partially asynchronous parallel methods* as introduced in [BT89, Chapter 7].

4.3.3 Main contribution

We prove *linear convergence* of the sequence $\{\mathbf{dist}_k\}$ generated from Algorithms 7 and 8, where $\mathbf{dist}_k := \|x_k - x_*\|$. The result is based on Lemma 12 below, which originally appeared in [FAJ14] and has been extensively used in recent works for proving linear convergence of sequences with errors.

Lemma 12. *Let $\{V_k\}$ be a sequence of nonnegative real numbers satisfying*

$$V_{k+1} \leq rV_k + q \max_{k-\tau \leq l \leq k} V_l, \quad l \geq 0,$$

for some nonnegative constants r and q. If $r + q < 1$, then

$$V_k \leq s^k V_0, \quad k \geq 1,$$

where $s = (r+q)^{\frac{1}{1+\tau}}$.

Our convergence proof follows the styles of [GOP17] and [PXYY16]. In the former, the authors prove linear convergence of the incremental aggregated unconstrained gradient method, while in the latter an asynchronous KM iteration is developed. Our contributions are summarized below.

1. *We prove convergence of an asynchronous and parallel forward-backward iteration of a sequence involving an inertial term.* To the best of our knowledge, this is the first result on accelerated asynchronous fixed-point iterations.

2. Contrary to the majority of existing popular schemes, *the proposed asynchronous iteration is deterministic, i.e., an arbitrary (block of) coordinate(s) can be selected to update at each iteration.* The coordinates can be chosen with varying frequencies, the only assumption being that each coordinate is updated at least once within a fixed time interval. The only work known to us that treats asynchronous updates in a deterministic way, though in a different setting, is [CE16].

3. The proposed iteration is quite general and encompasses many known algorithms as special cases, *i.e.,* several forms of the forward-backward splitting method (see Section 4.5). Indicatively, we propose new inertial and asynchronous instances of two existing and commonly used algorithms and we list some more that can be derived.

4.4 Convergence proof

We want to use Lemma 12, with $V_k = \|x_k - x_*\|^2,\ x_* \in \operatorname{zer} S$. In order to do so, we first express the outdated versions of the global vector x that appear in (4.5) (and consequently in iteration (4.6)) with respect to the original ones, perturbed by some additive errors. Subsequently, these errors are going to be upper-bounded by $\max_{k-K \leq m \leq k} \|x_m - x_*\|$, for some bounded delay K. We are going to go through the proof in steps.

4.4.1 Express delayed variables as additive error

The variables $y_B, y_{\text{write}}, y_{\text{write}}^{\text{prev}}$ that appear in the update (4.5) depend on outdated components of x, namely on the state of the vector x when a 'read' or 'write' operation was performed by agent i. It can be easily seen that *any past vector* $x_{k-l},\ l \in \{1, \dots, k-1\}$ *can be expressed as*

$$x_{k-l} = x_k - \sum_{m=k-l}^{k-1} (x_{m+1} - x_m) \ .$$

Consequently, the vectors that appear in (4.5) can be written as functions of the current vector x_k and some error.

$$\begin{aligned} x_{\text{read}}^i &= x_{k-l_i} = x_k - a_k^i, \quad \text{for some } l_i \in \{1, \dots, 2\tau\} \\ y_{\text{write}} &= x_{\text{write}}^i[i] = x_{k-l_i}[i] = x_k[i] - b_k^i[i], \quad \text{for some } l_i \in \{1, \dots, 2\tau\} \\ y_{\text{write}}^{\text{prev}} &= x_{k-l_i}[i] = x_k[i] - c_k^i[i], \quad \text{for some } l_i \in \{1, \dots, 3\tau\} \ , \end{aligned} \tag{4.7}$$

and the sequences $\{a_k^i\}_{i=1}^N$, $\{b_k^i\}_{i=1}^N$ and $\{c_k^i\}_{i=1}^N$ are all of the form $\sum_{m=k-l_i}^{k-1}(x_{m+1} - x_m)$ for some proper choice of l_i. In other words, equations (4.7) 'undo' all the changes that occured over the last updates, until the corresponding past state is recovered. Note that the l_i's in the three equations above are not the same.

The intervals within which the subscripts $l_i,\ i = 1, \dots, N$ reside are derived based on Assumption 5. The derivation is explained graphically in Figure 4.2.

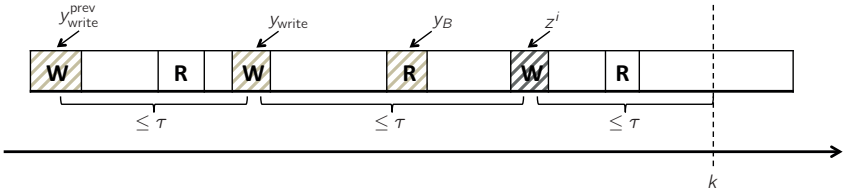


Figure 4.2: An update is about to occur at $k+1$. From Assumption 5, the observed agent will update again no later than τ time epochs after z^i was communicated to the coordinator. Consequently, y_{write} cannot be further than 2τ from the next update, while $y_{\text{write}}^{\text{prev}}$ cannot be further than 3τ.

Lemma 13. *Equation* (4.6) *can be expressed as*

$$x_{k+1} = x_k + \eta(Tx_k - x_k + e_k) = x_k - \eta(Sx_k - e_k) \ , \tag{4.8}$$

where e_k *is an error term whose* i^{th} *term is defined as*

$$e_k := T_A\left(T_B x_k + d_k + \beta(c_k - b_k)\right) - T_A(T_B x_k) \ , \tag{4.9}$$

with $b_k = \{b_k^i\}_{i=1}^N$ *and* $c_k = \{c_k^i\}_{i=1}^N$ *defined in* (4.7)*, and*

$$d_k[i] := \gamma(Bx_k)[i] - \gamma(B(x_k - a_k^i))[i] - a_k^i[i], \quad d_k = (d_k[1], \dots, d_k[N]) \ . \tag{4.10}$$

The proof of Lemma 13 is given in Appendix 4.8.1.

4.4.2 Isolate the error

The form of the iteration derived in Lemma 13 will help us separate the error sequence $\{e_k\}$ from the sequence of interest $\{\mathbf{dist}_k\}$. To this end, the following result holds, the derivation of which is given in Appendix 4.8.2.

Lemma 14. *The distance is upper-bounded as:*

$$\mathbf{dist}_{k+1}^2 \leq (1 - \eta(\nu - \epsilon))\, \mathbf{dist}_k^2 + \eta \left(\frac{1}{\epsilon} + \frac{\eta(1+\delta)}{\delta} \right) \|e_k\|^2 \ , \tag{4.11}$$

for $\eta \in (0, 1/(2(\delta+1)))$ *and any* $\delta > 0$, $\epsilon \in (0, \nu)$*, while* ν *is the quasi-strong monotonicity constant of* S *as introduced in Assumption 4.*

If we manage to bound the last two terms of the equation with respect to the maximum distance from the set of fixed points, inequality (4.11) will be in the form described by Lemma 12. In the next step, we start by bounding the error term $\|e_k\|$.

4.4.3 Bound the error recursively

We want to bound the error term $\|e_k\|$ by means of $\max_{k-K \leq l \leq k} \|x_l - x_*\|$ for some $K \in \mathbb{N}$. We will do so in two phases, first bounding $\|e_k\|$ (given in (4.9)) recursively with respect to itself:

$$\begin{aligned} \|e_k\| &= \|T_A\left(T_B x_k + d_k + \beta(c_k - b_k)\right) - T_A(T_B x_k)\| \\ &\leq \|d_k + \beta(c_k - b_k)\| \\ &\leq \|d_k\| + \beta\|c_k - b_k\| \ , \end{aligned} \tag{4.12}$$

where the first inequality follows from the nonexpansivity of T_A.

It thus suffices to bound $\|d_k\|$ and $\|c_k - b_k\|$ in a recursive way. The result is presented in Lemma 15 below and proven in Appendix 4.8.3.

Lemma 15. *The quantities* $\|c_k - b_k\|$ *and* $\|d_k\|$ *can be bounded recursively as:*

$$\|c_k - b_k\| \leq 2\eta N \Sigma_{3\tau}(k) \tag{4.13a}$$

$$\|d_k\| \leq \eta(1 + \gamma L) N \Sigma_{2\tau}(k) \ , \tag{4.13b}$$

where

$$\Sigma_K(k) := \sum_{m=k-K}^{k-1} \left(\|d_m\| + \beta\|c_m - b_m\| + \|Sx_m\| \right) \tag{Σ}$$

and L *is the inverse cocoercivity constant of the operator* B *from Assumption 3.*

4.4.4 Bound the error with respect to the maximum distance from the set of fixed points

Looking at (4.13a) and (4.13b), what needs to be bounded is the quantity (Σ), and consequently the three sums, *i.e.*, $\sum_{m=k-K}^{k-1}\|d_m\|$, $\sum_{m=k-K}^{k-1}\|c_m-b_m\|$ and $\sum_{m=k-K}^{k-1}\|Sx_m\|$ for $K=\{2\tau,3\tau\}$ with respect to the maximum distance from the set of fixed points of T. Lemma 16 below states the result.

Lemma 16. *The sequence* (Σ) *can be upper bounded by the maximum distance from the set of fixed points of* T *as*

$$\|\Sigma_K(k)\| \leq 2K(YN+1)\max_{k-K-3\tau\leq j\leq k-1}\mathbf{dist}_j \ ,$$

where $Y:=1+\gamma L+2\beta$. *Using the above, the error* $\|e_k\|$ *can be bounded as*

$$\|e_k\| \leq \eta X\max_{k-6\tau\leq j\leq k-1}\mathbf{dist}_j \ , \tag{4.14}$$

where

$$X := N(YN+1)(4\tau(1+\gamma L)+6\beta\tau) \ .$$

The Lemma is proven in Appendix 4.8.4.

4.4.5 Condition for convergence

Let us now recover the condition for the algorithm to converge. By using (4.14) in (4.11), we have the desired result expressed as:

$$\mathbf{dist}_{k+1}^2 \leq r(\eta)\,\mathbf{dist}_k^2 + q(\eta)\max_{k-6\tau\leq j\leq k-1}\mathbf{dist}_j^2 \ ,$$

where

$$r(\eta) := 1-\eta(\nu-\epsilon), \quad q(\eta) := \eta^3X^2\left(\frac{1}{\epsilon}+\frac{\eta(1+\delta)}{\delta}\right). \tag{4.15}$$

Lemma 12 suggests that *the asynchronous inertial FBS iteration* (4.8) *will converge to a zero of* S *at a linear rate* $(r(\eta)+q(\eta))^{\frac{1}{1+6\tau}}$ *if the condition*

$$1-\eta(\nu-\epsilon)+\eta^3X^2\left(\frac{1}{\epsilon}+\frac{\eta(1+\delta)}{\delta}\right)<1 \tag{4.16}$$

holds.

Theorem 7. *Iteration* (4.8) *will converge at a linear rate as described in Lemma 12 with* $r(\eta)$ *and* $q(\eta)$ *given in* (4.15) *for*

$$\eta < \min\left\{\frac{1}{2(1+\delta)}, \frac{1}{X}\sqrt{\frac{2\delta\epsilon(\nu-\epsilon)}{2\delta+\epsilon}}\right\},$$

where $\gamma\in(0,\gamma_{\max})$, $\delta>0$, $\epsilon\in(0,\nu)$ *and* $\beta>0$. *The upper bound* $\gamma_{\max}$ *ensures that the stepsize* γ *is admissible (a possible option is, e.g.,* $\gamma_{\max}=2/L$ *as proven in Appendix 4.8.6).*

Theorem 7 is proven in Appendix 4.8.5.

4.5 Connection to other methods

The operators T_A and T_B that constitute the proposed iteration (4.8) give rise to several asynchronous accelerated versions of known algorithms. The form of T_B, namely the *forward step* $T_B = I - \gamma B$, played an important role in allowing us to express the errors that arise due to the delays and the inertial term in an additive manner. At the same time, this limits the applicability of our iteration to methods that can be cast as forward-backward iterations. Below, we introduce the extensions of some popular algorithms that can be seen as special cases of iteration (4.8).

4.5.1 Gradient descent

Classical gradient descent can be recovered by choosing $T_A = I$ and $T_B = I - \gamma \nabla f$, for a differentiable strongly convex function $f : \mathbb{R}^n \mapsto \mathbb{R}$ with Lipschitz continuous gradient. The inertial asynchronous iteration becomes:

$$x_{k+1}[i] = (1-\eta)x_k[i] + \eta \underbrace{\left(x^i_{\text{read}}[i] - \gamma \nabla_i f(x^i_{\text{read}}) + \beta(y_{\text{write}} - y^{\text{prev}}_{\text{write}})\right)}_{z^i} \ ,$$

where $\nabla_i f(x) := \nabla_{x[i]} f(x) : \mathbb{R}^n \mapsto \mathbb{R}^{n_i}$, $\sum_{i=1}^N n_i = n$, and corresponds to an asynchronous iteration of the heavy ball gradient method.

4.5.2 Proximal gradient

Let us consider the optimization problem

$$\text{minimize } f(x) + \sum_{i=1}^N g_i(x[i]) \ , \tag{4.17}$$

where f is stronlgy convex differentiable with Lipschitz continuous gradient, while $g_i \in \Gamma_0(\mathbb{R}^{n_i})$ and $x \in \mathbb{R}^n = \mathbb{R}^{n_1} \times \cdots \times \mathbb{R}^{n_N}$. The iteration reads:

$$x_{k+1}[i] = (1-\eta)x_k[i] + \eta \underbrace{\mathbf{prox}_{\gamma g_i}\left(x^i_{\text{read}}[i] - \gamma \nabla_i f(x^i_{\text{read}}) + \beta\left(y_{\text{write}} - y^{\text{prev}}_{\text{write}}\right)\right)}_{z^i} \ ,$$

and corresponds to a relaxed and asynchronous version of the proximal Heavy Ball method in [OBP15].

4.5.3 Other methods

Besides the two instances analyzed above, a variety of convex optimization algorithms can be expressed as a forward-backward iteration, and consequently give rise to novel asynchronous implementations. The *generalized forward backward splitting* [RFP13; RL15], the *forward-Douglas-*

Rachford splitting [BA12; Dav15a] and several *primal-dual optimization methods* [CCPV14] can be viewed as candidates, just to name a few.

4.6 Application: Distribution network real-time dispatch

Our goal is to track a 15-minute resolution trajectory, called the dispatch plan, that is computed one day before the beginning of operation. This is achieved by modulating the power consumption of a grid-connected battery energy storage system (BESS) and of the thermal consumption of a fleet of commercial controllable buildings (CB). The problem has been detailed in Section 2.5, and has been originally proposed and solved in [SNCP16] using as benchmark an experimental setup with one controllable office and a large battery. In this example we scale up the problem by considering several CB's and we compare synchronous versus asynchronous implementations, including our proposed scheme.

4.6.1 Modeling the agents

The grid comprises the following entities:

Controllable Loads Small, medium and large office buildings, generated by [GQJ15], as described by (2.25) in Section 2.5. The buildings can contribute to the dispatchability of the network by increasing or decreasing their consumption with respect to some *baseline power profile*. The optimization objective for CB i becomes

$$g_i^{\mathrm{cb}}(p_i^{\mathrm{cb}}, u_i, x_i, y_i) := \left\{ \frac{1}{2}\|y_i(t) - T_i^{\mathrm{ref}}(t)\|_2^2 \;\middle|\; (p_i^{\mathrm{cb}}, u_i, x_i, y_i) \in \mathcal{C}_i^{\mathrm{cb}} \right\}, \tag{4.18}$$

$$\mathcal{C}_i^{\mathrm{cb}} = \left\{ \begin{array}{l} x_i(t+1) = A_i x_i(t) + B_{u,i} u_i(t) + B_{w,i} \hat{w}_i(t) \\ x_i(0) = x_i^{\mathrm{init}} \\ y_i(t) = C_i x_i(t) \\ y_{\min,i}(t) \le y_i(t) \le y_{\max,i}(t) \\ u_{\min,i} \le u_i(t) \le u_{\max,i} \\ p_i^{\mathrm{cb}}(t) = \sum_{j=1}^{N_i} u_{ij}(t) \end{array} \right\},$$

with all the variables described in Section 2.5. The desired zone temperature is denoted with T_i^{ref}.

Storage The setup is completed with a grid-connected Lithium Titanate grid-connected 500kWh BESS described by (2.26) in Section 2.5.

$$g^{\mathrm{bess}}(p^{\mathrm{bess}}) := \left\{ \frac{1}{2}\sum_{t=1}^{T} \|SOC(t) - SOC^{\mathrm{ref}}(t)\|_2^2 \;\middle|\; p^{\mathrm{bess}} \in \mathcal{C}^{\mathrm{bess}} \right\}, \tag{4.19}$$

$$\mathcal{C}^{\mathrm{bess}} = \left\{ \begin{array}{l} SOC(t+1) = aSOC(t) + bp^{\mathrm{bess}}(t) \\ SOC(0) = SOC^{\mathrm{init}} \\ SOC_{\min} \le SOC(t) \le SOC_{\max} \\ p_{\min}^{\mathrm{bess}} \le p^{\mathrm{bess}}(t) \le p_{\max}^{\mathrm{bess}} \end{array} \right\}.$$

Simulation characteristics				
Data	1st January 2000			
Location	Lausanne			
Time	00:00 - 24:00			
Sampling time	15			min
Horizon	96			–
Buildings				
Minimum temperature (day/night)	20/18			°C
Maximum temperature (day/night)	24/28			°C
Heat pump COP	3.0			–
	Small	**Medium**	**Large**	
Number of systems (Case A, B, C, D)	3/6/14/32	2/4/5/16	0/0/1/2	#
Area	511	4982	46320	m^2
Tariff (day/night)	21.6/12.7	13.15/8.3	13.15/8.3	ct./kWh
Number of states	15	54	57	–
Number of inputs	5	18	19	–
Average thermal consumption	4	40	75	W/m^2
Average computation time (prox per agent)	0.070 ± 0.010	0.243 ± 0.005	0.267 ± 0.001	sec
Battery				
Energy storage capacity	500			kWh
C-rate	0.2			–
Material	Lithium-ion			
Average computation time (prox)	0.023 ± 0.003			sec

Table 4.1: Micro-grid case study overview

4.6.2 Modeling the dispatch problem

We cast the dispatch problem in a slightly different way than the one presented by (2.24), namely we penalize the dispatch constraint instead of enforcing it so that we are able to apply the proposed algorithm. The problem reads

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\sum_{t=1}^{T}\left(\|SOC(t) - SOC^{\text{ref}}(t)\|_2^2 + \alpha_1\|p^{\text{bess}}(t)\|_2^2\right) && \text{(4.20a)}\\
& + \frac{1}{2}\sum_{i=1}^{N}\sum_{t=1}^{T}\left(\|y_i(t) - T_i^{\text{ref}}(t)\|_2^2 + \alpha_1\|p_i^{\text{cb}}(t) - \hat{p}_i^{\text{cb}}(t)\|_2^2\right) && \text{(4.20b)}\\
& + \frac{\alpha_2}{2}\sum_{t=1}^{T}\|p^{\text{bess}}(t) + \sum_{i=1}^{N}(p_i^{\text{cb}}(t) - \hat{p}_i^{\text{cb}}(t)) - r(t)\|_2^2 && \text{(4.20c)}\\
\text{subject to} \quad & (p_i^{\text{cb}}, u_i, x_i, y_i) \in \mathcal{C}_i^{\text{cb}},\ i = 1, \dots, N && \text{(4.20d)}\\
& p^{\text{bess}} \in \mathcal{C}^{\text{bess}}\ , && \text{(4.20e)}
\end{aligned}$$

with variables $p_i^{\text{cb}},\ i = 1, \dots, N$ and p^{bess}, and the variables u_i, x_i, y_i local to CB i.

Equations (4.20a) and (4.20b) express the deviation of the BESS SOC from its reference value, set to $SOC^{\text{ref}}(t) = 0.8SOC_{\max}$, as well as the deviation of the indoor temperature from its reference

value (see Table 5.1). The additional quadratic terms penalized with $\alpha_1 = 10^{-2}$ are introduced for regularization purposes. Equation (4.20c) expresses *the deviation of the aggregate buildings' flexibility* $\sum_{i=1}^{N}(p_i^{\text{cb}}(t) - \hat{p}_i^{\text{cb}}(t))$ *along with the BESS flexibility* p^{bess} *from the given reference* r. In a perfectly dispatchable network, this term should be put to zero, hence it is penalized much more heavily than the other terms with $\alpha_2 = 10^4$.

4.6.3 Simulation setup

Our purpose is to solve (4.20) by means of the synchronous, the asynchronous and the inertial asynchronous versions of the FBS algorithm. To this end, let us make the problem more compact by grouping the terms. The terms depicted in blue color are private to the BESS system, the terms in green are private to the CB agents, while red terms comprise the global objective, denoted hereafter as $f(p^{\text{bess}}, p^{\text{cb}})$. Note that the local subproblems in blue and green correspond to the quadratic programs (QP) (5.39) and (4.18), respectively. Since each variable p_i^{cb} is private to agent i and f couples all the variables through the (strongly convex) quadratic objective, the problem takes the form (4.17) and is consequently solved using the proximal gradient method.

We consider four case studies (A, B, C and D), namely a mix of the BESS and $N = 5, 10, 20$ and 50 CB's. The tracking signal r that we assume is the realized *Area Control Signal (ACS)*, as it was broadcast by the Swiss grid operator [Swi03], for the 1$^{\text{st}}$ of January of the year 2000, scaled down by the appropriate factor in each of the four cases so that it becomes (almost) trackable by our mix. The prediction horizon has a length of 24 hours, or $T = 96$ in 15 minutes intervals.

The source of asynchronicity in this framework is the diverse computational load of the different agents (CB's and the BESS). Although problems (4.18) and (5.39) are QP's, their size varies greatly with the number of states and inputs, as indicated in Table 5.1. The delay τ is, therefore, computed based on the number of the updates per agent in a unit of time. In order to compute this number, we solve 100 proximal minimization steps per agent and fit a normal distribution to the solve times. The average computation time is then used to decide upon the frequency of the updates. The communication delays are assumed to be zero in the simulation. The proximal minimization problems are solved using the YALMIP optimizer [LÖ4] with the Gurobi solver. Finally, the relaxation parameter is set to $\eta = 0.9$, which, in spite of the (much) smaller value suggested by Theorem 7, worked well in our setting.

Problem (4.20) is solved using the proximal gradient method. A comparison between (i) the synchronous version of the method (all CB's and BESS update before a new gradient ∇f is communicated), (ii) the asynchronous version with coordinate updates (only $i = i_k$ updates at each global clock count, with $i = 1, \ldots, N + 1$ and $N + 1$ corresponds to the BESS agent), (iii) the asynchronous aggregated version ((4.17) with $\beta = 0$) and (iv) the asynchronous inertial aggregated version ((4.17) with $\beta = 0.99$). Table 4.2 depicts the accuracy reached within $Ts = 40$sec of simulated wall-clock time using the four algorithms presented above in the four case studies. Table 4.3 presents the average number of updates per type of agent within these 40sec.

Several conclusions can be derived. First, the asynchronous version with coordinate updates and the asynchronous aggregated version of the proximal gradient method are almost identical in performace, thus there is neither deterioration (at least in the simulated cases) nor improvement when using the old updates. Second, the asynchronous versions perform considerably better than their synchronous counterpart in all cases. This is an expected outcome since the larger the load

Algo. / N	Sync	Async Coordinate	Async Aggregated	Async Agg. Inert.
5	0.116	0.030	0.030	0.003
10	0.252	0.061	0.061	0.012
20	0.315	0.078	0.078	0.015
50	0.824	0.649	0.649	0.448

Table 4.2: Accuracy reached within $Ts = 40$sec.

Algo. / N	5	10	20	50
Sync	172	156	148	137
Async Agg. Inert.	1639 565.6/171/–	1798 588.5/157/–	1674 515.6/163.2/147	1101 426.37/148.50/142.50

Table 4.3: Average number of updates per agent within simulation time.

imbalance among the agents, the more the algorithm benefits from the asynchronicity, as suggested by the number of updates per agent in Table 4.3. Finally, the proposed inertial acceleration scheme results in considerably better performance in terms of speed of convergence in all cases. A graphical depiction of the convergence performance of the four methods for $N = 5$ is given in Figure 4.3. For Case D ($N = 50$), the area plot in Figure 4.4 elaborates on the contribution of each of the agents in tracking the reference signal.

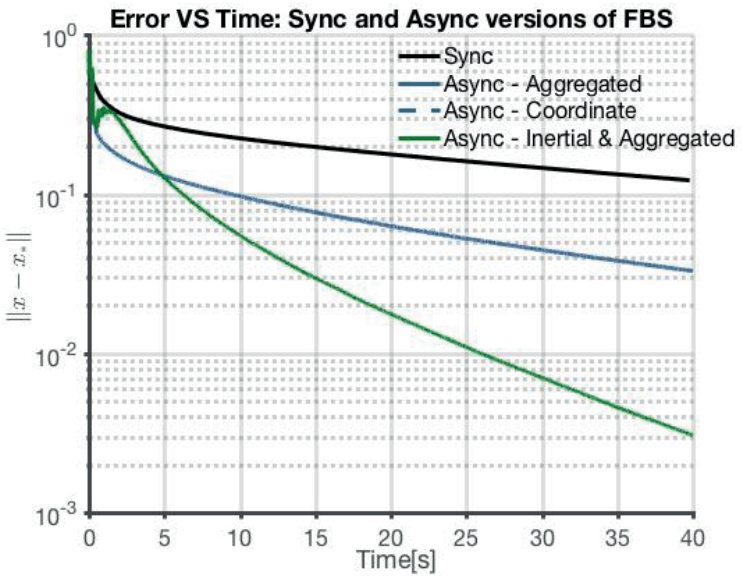


Figure 4.3: Distance from optimizer VS wall-clock time.

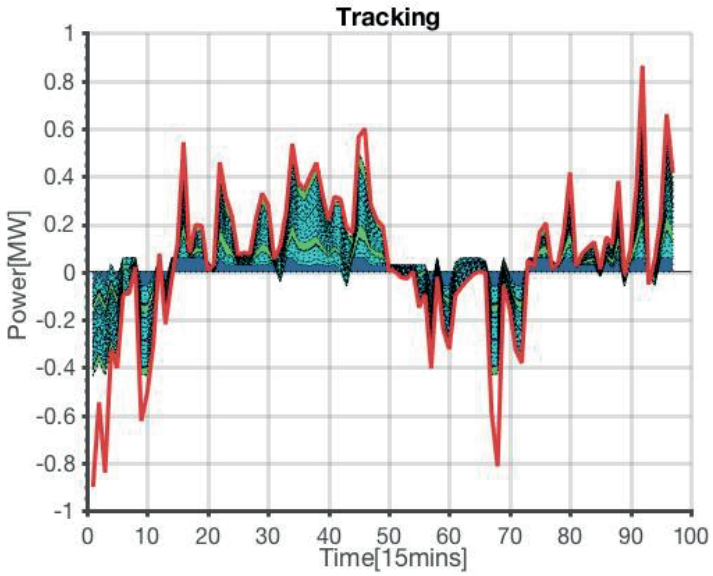


Figure 4.4: The reference signal to be tracked is depicted in red. The contribution of the BESS is colored in dark blue, that of the medium scale buildings in lighter blue and that of the two large buildings in green. The contribution of the small buildings is colored in pink, but is hardly visible due to their small capacity and despite their large population.

Remark 10. *A publicly available version of the code presented in this section, in the from of a Jupyter notebook, can be found on* `https://github.com/stathopog/AsyncInertialFBS`. *The code is implemented in Julia, while the optimization problems are modeled in JuMP [DHL17] and are solved using Gurobi.*

4.7 Conclusion

We proposed an inertial and asynchronous forward-backward iteration for solving monotone inclusion problems. The iteration is tailored for distributed convex optimization problems and differs from existing approaches since (i) the component updates are selected in a deterministic way, (ii) older updates contribute to the upcoming one in a fashion resembling aggregated gradient methods and (iii) the iteration hosts a momentum term that speeds up practical convergence. We derived new versions of two commonly used methods stemming from our approach, and we illustrated the effectiveness of the method when used to solve an optimal dispatch problem in a distribution grid with a pool of heterogeneous energy resources.

There is plenty of space for improving the proposed approach and the like. The first things that naturally come to mind regard dropping the strong monotonicity (strong convexity) assumption and using an optimal Nesterov-like momentum sequence instead of a fixed scalar value. In addition, although the momentum sequence practically boosts the performance of the scheme, the current convergence analysis does not exhibit its benefits. On the contrary, our analysis treats the addi-

tional degree of freedom that momentum offers as an extra perturbation. Since it is known that inertial acceleration improves the rate at which the sequence of iterates converges to a solution in the strongly convex case, we suspect that a similar result is applicable to the asynchronous framework. The connection between asynchronicity and the introduction of momentum to the stochastic gradient method to have been recently studied in [MZHR16], and might result in useful directions regarding the upcoming analysis.

4.8 Appendices

4.8.1 Proof of Lemma 13

Using Assumption 3 and equations (4.7), we can start by rewriting $y_B = (T_B x^i_{\text{read}})[i]$.

$$\begin{aligned}(T_B x^i_{\text{read}})[i] &= x^i_{\text{read}}[i] - \gamma(Bx^i_{\text{read}})[i] \\ &= x_k[i] - a^i_k[i] - \gamma(Bx_k)[i] + \gamma(Bx_k)[i] - \gamma(Bx^i_{\text{read}})[i] \\ &= (T_B x_k)[i] + \gamma\left((Bx_k)[i] - (Bx^i_{\text{read}})[i]\right) - a^i_k[i] \\ &= (T_B x_k)[i] + d^i_k[i] \ ,\end{aligned}$$

where $d_k[i] = \gamma\left((Bx_k)[i] - (Bx^i_{\text{read}})[i]\right) - a^i_k[i]$.

Similarly, we have from equations (4.7) that

$$\begin{aligned}\beta(y_{\text{write}} - y^{\text{prev}}_{\text{write}}) &= \beta(x_k[i] - b^i_k[i] - x_k[i] + c^i_k[i]) \\ &= \beta(c^i_k[i] - b^i_k[i]) \ .\end{aligned}$$

Using the above relations, a coordinate update of iteration (4.6) can be expressed as

$$x_{k+1}[i] = x_k[i] + \eta\Big(T_{A_i}\left((T_B x_k)[i] + d_k[i] + \beta(c^i_k[i] - b^i_k[i])\right) - x_k[i]\Big) \ ,$$

or, equivalently, as

$$x_{k+1}[i] = x_k[i] + \eta\left(T_{A_i}((T_B x_k)[i]) - x_k[i] + e_k[i]\right) \ ,$$

with $e_k[i] = T_{A_i}\left((T_B x_k)[i] + d_k[i] + \beta(c^i_k[i] - b^i_k[i])\right) - T_{A_i}((T_B x_k)[i])$, which concludes the proof.

4.8.2 Proof of Lemma 14

Squaring (4.8) we get:

$$\|x_{k+1} - x_*\|^2 = \|x_k - x_*\|^2 - 2\eta\langle x_k - x_*, Sx_k - e_k\rangle + \eta^2\|Sx_k - e_k\|^2 \ . \tag{4.21}$$

Let us now analyze the second and third term in (4.21).

- Bound $-2\eta\langle x_k - x_*, Sx_k - e_k\rangle$: We will upper-bound the resulting inner product terms. In order to do so, we use both the cocoercivity and the quasi-strong monotonicity of S, the former proven in Appendix 4.8.6, and the latter holding from Assumption 3. Since S is 1/2-

cocoercive, we have that

$$\langle x_k - x_*, Sx_k \rangle \geq \frac{1}{2}\|Sx_k\|^2.$$

From the quasi-ν-strong monotonicity of S we have:

$$\langle x_k - x_*, Sx_k \rangle \geq \nu\|x_k - x_*\|^2.$$

Putting these two together, we get that

$$-2\eta\langle x_k - x_*, Sx_k \rangle \leq -\eta\nu\,\mathbf{dist}_k^2 - \frac{\eta}{2}\|Sx_k\|^2 \ . \tag{4.22}$$

For the second inner product term involving the error we can easily derive the bound

$$2\eta\langle x_k - x_*, e_k \rangle \leq 2\eta\,\mathbf{dist}_k\,\|e_k\| \ . \tag{4.23}$$

Equations (4.22) and (4.23) result in the bound

$$-2\eta\langle x_k - x_*, Sx_k - e_k \rangle \leq -\eta\nu\,\mathbf{dist}_k^2 - \frac{\eta}{2}\|Sx_k\|^2 + 2\eta\,\mathbf{dist}_k\,\|e_k\| \ . \tag{4.24}$$

- Bound $\eta^2\|Sx_k - e_k\|^2$: By developing the square, we have that

$$\eta^2\|Sx_k - e_k\|^2 = \eta^2\left(\|Sx_k\|^2 - 2\langle Sx_k, e_k\rangle + \|e_k\|^2\right) \ . \tag{4.25}$$

The inner product term in (4.25) can be bounded by employing Young's inequality[1] as follows:

$$\begin{aligned} -2\langle Sx_k, e_k\rangle &\leq 2\|Sx_k\|\|e_k\| \\ &\leq 2(\frac{\delta}{2}\|Sx_k\|^2 + \frac{1}{2\delta}\|e_k\|^2) \\ &= \delta\|Sx_k\|^2 + \frac{1}{\delta}\|e_k\|^2 \ , \end{aligned} \tag{4.26}$$

for any $\delta > 0$. Putting together (4.25) and (4.26), we get the bound:

$$\eta^2\|Sx_k - e_k\|^2 \leq \eta^2(1+\delta)\|Sx_k\|^2 + \eta^2\frac{(\delta+1)}{\delta}\|e_k\|^2 \ . \tag{4.27}$$

Using (4.24) and (4.27), inequality (4.21) can be written as

$$\mathbf{dist}_{k+1}^2 \leq (1-\eta\nu)\,\mathbf{dist}_k^2 + \eta\left(-\frac{1}{2} + \eta(1+\delta)\right)\|Sx_k\|^2 + 2\eta\,\mathbf{dist}_k\,\|e_k\| + \eta^2\frac{(\delta+1)}{\delta}\|e_k\|^2 \ . \tag{4.28}$$

The second term in the sum can be eliminated by asumming that

$$-\frac{1}{2} + \eta(1+\delta) < 0 \Rightarrow \eta < \frac{1}{2(1+\delta)} \ , \tag{4.29}$$

[1] For two nonnegative real numbers x and y, it holds that $xy \leq \frac{\delta x^2}{2} + \frac{y^2}{2\delta}$ for every $\delta > 0$.

which gives rise to the inequality

$$\mathbf{dist}_{k+1}^2 \leq (1-\eta\nu)\,\mathbf{dist}_k^2 + 2\eta\,\mathbf{dist}_k\,\|e_k\| + \eta^2\frac{(\delta+1)}{\delta}\|e_k\|^2 \ . \tag{4.30}$$

The complicating term on the right hand side can be eliminated by using once more Young's inequality, *i.e.,*

$$\begin{aligned} 2\eta\,\mathbf{dist}_k\,\|e_k\| &\leq 2\eta\left(\frac{\epsilon}{2}\,\mathbf{dist}_k^2 + \frac{1}{2\epsilon}\|e_k\|^2\right) \\ &= \eta\epsilon\,\mathbf{dist}_k^2 + \frac{\eta}{\epsilon}\|e_k\|^2 \ . \end{aligned}$$

4.8.3 Proof of Lemma 15

We will bound the error term $\|e_k\|$ componentwise. For some arbitrary $i \in \{1,\ldots,N\}\}$ and $k \in \mathbb{N}$, we have from (4.12) that

$$\|e_k[i]\| \leq \|d_k[i]\| + \beta(\|c_k^i[i]\| + \|b_k^i[i]\|) \ .$$

Consequently,

$$\|e_k\| \leq (1+\gamma L)N \max_{1\leq i\leq N}\|a_k^i\| + \beta N \max_{1\leq i\leq N}(\|c_k^i\| + \|b_k^i\|) \ . \tag{4.31}$$

The first term can be recovered by using the $1/L$-cocoercivity of B (Assumption 3) in (4.10), while the second term follows from the inequality

$$\|c_k^i[i]\| + \|b_k^i[i]\| \leq \|c_k^i\| + \|b_k^i\| \ .$$

- We want to bound the two summands of (4.31). Let us start with bounding $\|a_k^i\|$, for which we have for all i:

$$\begin{aligned} \max_{1\leq i\leq N}\|a_k^i\| &\leq \sum_{m=k-2\tau}^{k-1}\|x_{m+1}-x_m\| \\ &= \eta\sum_{m=k-2\tau}^{k-1}\|e_m - Sx_m\| \\ &\leq \eta\left(\sum_{m=k-2\tau}^{k-1}\|e_m\| + \sum_{m=k-2\tau}^{k-1}\|Sx_m\|\right) \\ &\leq \eta\sum_{m=k-2\tau}^{k-1}(\|d_m\| + \beta\|c_m - b_m\| + \|Sx_m\|) \ . \end{aligned} \tag{4.32}$$

  The first inequality follows from the definitions of a_k^i in (4.7), the first equality from (4.8), while the last two inequalities from the triangle inequality and (4.12).

- Bound $\|c_k - b_k\|$: Following the same process as in (4.32), we have that

$$\begin{aligned}\max_{1\leq i\leq N}(\|c_k^i\| + \|b_k^i\|) &\leq \sum_{m=k-3\tau}^{k-1}\|x_{m+1} - x_m\| + \sum_{m=k-2\tau}^{k-1}\|x_{m+1} - x_m\| \\ &\leq \eta\Big(\sum_{m=k-3\tau}^{k-1}(\|d_m\| + \beta\|c_m - b_m\| + \|Sx_m\|) + \\ &\quad \sum_{m=k-2\tau}^{k-1}(\|d_m\| + \beta\|c_m - b_m\| + \|Sx_m\|)\Big) \ . \end{aligned} \tag{4.33}$$

Finally, (4.32), and (4.33) can be bounded by means of the quantity (Σ), and by substituting (4.32) to (4.31), the result follows.

4.8.4 Proof of Lemma 16

Let us start by bounding the quantities involved in (Σ), namely $\|b_k\|$, $\|c_k\|$ and $\|d_k\|$ with respect to the maximum distance from the optimizer. The following inequalities hold:

$$\begin{aligned}\|b_k\| &\leq N\max_{1\leq i\leq N}\|b_k^i\| \\ \|c_k\| &\leq N\max_{1\leq i\leq N}\|c_k^i\| \\ \|d_k\| &\leq (1+\gamma L)N\max_{1\leq i\leq N}\|a_k^i\| \ . \end{aligned} \tag{4.34}$$

Note, also, that $\forall\, i = 1,\dots,N$ and for $l_i \in \{1,\dots,3\tau\}$ holds

$$\begin{aligned}\|x_k - x_{k-l_i}\| &= \|x_k - x_* + x_* - x_{k-l_i}\| \\ &\leq \|x_k - x_*\| + \|x_* - x_{k-l_i}\| \ . \end{aligned}$$

Since the first inequality holds $\forall\, i = 1,\dots,N$, by denoting $i_* = \arg\max_{i\in\{1,\dots,N\}}\|x_k - x_{k-l_i}\|$, we get

$$\begin{aligned}\max_{1\leq i\leq N}\|x_k - x_{k-l_i}\| &= \|x_k - x_{k-l_{i_*}}\| \\ &\leq \big(\|x_k - x_*\| + \|x_* - x_{k-l_{i_*}}\|\big) \ . \end{aligned}$$

From the definition of a_k^i in (4.7) we have that

$$\begin{aligned}\max_{1\le i\le N}\|a_k^i\| &\le \max_{1\le i\le N}\|x_k - x_{k-l_i}\| \\ &\le \left(\|x_k - x_*\| + \|x_* - x_{k-l_{i_*}}\|\right) \\ &\le \left(\|x_k - x_*\| + \max_{k-2\tau\le m\le k-1}\mathbf{dist}_m\right) \\ &\le 2\max_{k-2\tau\le m\le k}\mathbf{dist}_m \ , \end{aligned} \tag{4.35}$$

for some $l_i \in \{1,\dots,2\tau\}$.

Substituting in (4.34), and following developments similar to (4.35), we conclude that

$$\begin{aligned}\|b_k\| &\le 2N\max_{k-2\tau\le m\le k}\mathbf{dist}_m \\ \|c_k\| &\le 2N\max_{k-3\tau\le m\le k}\mathbf{dist}_m \\ \|d_k\| &\le 2(1+\gamma L)N\max_{k-2\tau\le m\le k}\mathbf{dist}_m \end{aligned} \tag{4.36}$$

Using (4.36), the sums can be easily bounded as shown below.

$$\begin{aligned}\sum_{m=k-K}^{k-1}\|a_m\| &\le 2NK\max_{k-K-2\tau\le j\le k-1}\mathbf{dist}_j \\ \sum_{m=k-K}^{k-1}\|b_m\| &\le 2NK\max_{k-K-2\tau\le j\le k-1}\mathbf{dist}_j \\ \sum_{m=k-K}^{k-1}\|c_m\| &\le 2NK\max_{k-K-3\tau\le j\le k-1}\mathbf{dist}_j \\ \sum_{m=k-K}^{k-1}\|d_m\| &\le 2(1+\gamma L)NK\max_{k-K-2\tau\le j\le k-1}\mathbf{dist}_j \\ \sum_{m=k-K}^{k-1}\|Sx_m\| &\le 2K\max_{k-K\le j\le k-1}\mathbf{dist}_j \ , \end{aligned} \tag{4.37}$$

the last inequality following from Corollary 1.

From the definition of $\Sigma_K(k)$ in (Σ) and from (4.37), by introducing

$$Y := 1 + \gamma L + 2\beta,$$

we have that

$$\begin{aligned}\Sigma_K(k) &\leq \sum_{m=k-K}^{k-1} (\|d_m\| + \beta\|c_m\| + \beta\|b_m\| + \|Sx_m\|) \\ &\leq \underbrace{2K(YN+1)}_{W(K)} \max_{k-K-3\tau\leq j\leq k-1} \mathbf{dist}_j\end{aligned}$$

Since (Σ) is bounded, we can accordingly bound (4.13a) and (4.13b):

$$\begin{aligned}\|c_k - b_k\| &\leq \eta N 2\Sigma_{3\tau}(k) \\ &\leq \eta N 2W(3\tau) \max_{k-6\tau\leq j\leq k-1} \mathbf{dist}_j \\ &= \eta 2(3\tau)N(YN+1) \max_{k-6\tau\leq j\leq k-1} \mathbf{dist}_j \qquad (4.38a) \\ \|d_k\| &\leq \eta N(1+\gamma L)\Sigma_{2\tau}(k) \\ &\leq \eta N(1+\gamma L)W(2\tau) \max_{k-5\tau\leq j\leq k-1} \mathbf{dist}_j \\ &= \eta(1+\gamma L)4\tau N(YN+1) \max_{k-5\tau\leq j\leq k-1} \mathbf{dist}_j \ . \qquad (4.38b)\end{aligned}$$

Using (4.38a) and (4.38b), $\|e_k\|$ from (4.12) can be bounded as

$$\begin{aligned}\|e_k\| &\leq \|d_k\| + \beta\|c_k - b_k\| \\ &\leq \eta \underbrace{N(YN+1)(4\tau(1+\gamma L) + 6\beta\tau)}_{X} \max_{k-6\tau\leq j\leq k-1} \mathbf{dist}_j \ , \qquad (4.39)\end{aligned}$$

where we bounded the quantities with the maximum delay that appeared in (4.38a) and (4.38b).

4.8.5 Proof of Thoerem 7

Proof Note that (4.16) simplifies to

$$\eta^2 X^2 \left(\frac{1}{\epsilon} + \frac{\eta(1+\delta)}{\delta}\right) < \nu - \epsilon \ .$$

As a result, we need to find parameters $\eta, \beta, \gamma, \delta, \epsilon$ such that the following set of inequalities are satisfied:

$$\begin{cases} \eta^2 X^2 \left(\frac{1}{\epsilon} + \frac{\eta(1+\delta)}{\delta}\right) < \nu - \epsilon, \\ Y = 1 + \gamma L + 2\beta, \\ X = N(YN+1)(4\tau(1+\gamma L) + 6\beta\tau), \\ \delta > 0, \\ \epsilon > 0, \\ \beta > 0, \\ 0 < \gamma < \gamma_{\max}, \\ 0 < \eta < \frac{1}{2(1+\delta)}. \end{cases} \qquad (4.40)$$

The upper bound $\gamma_{\max}$ ensures that the stepsize γ is admissible (a possible option is, *e.g.,* $\gamma_{\max} = 2/L$ as proven in Appendix 4.8.6). We start be noting that the values of δ and ϵ are irrelevant as long as they are positive. To this end, we can start by choosing ϵ such that $\nu - \epsilon > 0$. From the inequality $\eta < 1/(2(1+\delta))$ it follows that

$$\frac{1}{\epsilon} + \frac{\eta(1+\delta)}{\delta} < \frac{2\delta + \epsilon}{2\delta\epsilon},$$

thus having

$$\eta^2 < \frac{2\delta\epsilon(\nu - \epsilon)}{X^2(2\delta + \epsilon)},$$

from which the result follows.

4.8.6 Cocoercivity and quasi-strong monotonicity of S

Proof of Corollary 1.

From [BC11, Proposition 4.33] we have that T is nonexpansive if and only if S is 1/2-cocoercive. Hence it suffices to show that T is nonexpansive. From Assumption 3, B is $1/L$-cocoercive, which means that γB is $1/\gamma L$-cocoercive. It follows from [RL15, Lemma 5.1 (iv)] that $T_B = I - \gamma B$ is $\gamma L/2$-averaged. From [BC11, Proposition 4.25 (i)] it follows that T_B is nonexpansive provided that $\gamma < 2/L$. Finally, from Assumptions 2 and 3 we conclude that T is nonexpansive as the composition of nonexpansive operators.

Lemma 17. *If B is μ-strongly monotone, then the operator S is quasi-ν-strongly monotone, where $\nu = 1 - \sqrt{(1 - 2\gamma\mu + \mu\gamma^2 L)}$.*

Proof: The Lemma is proven in [PXYY16, Proposition 2] for the case of the proximal gradient method. The proof below is essentially the same generalized for an operator T. From [BC11, Example 22.5] we have that if T is c-Lipschitz continuous for some $c \in [0, 1)$ then $I - T$ is $(1-c)$-strongly monotone. Let us then prove that T is indeed Lipschitz continuous. For any $x \in \mathcal{H}$ and $x_* \in \operatorname{fix} T$ it holds that:

$$\begin{aligned}
\|T_B x - T_B x_*\|^2 &= \|x - x_*\|^2 - 2\gamma\langle x - x_*, Bx - Bx_*\rangle + \gamma^2\|Bx - Bx_*\|^2 \\
&\leq \|x - x_*\|^2 - \gamma(2 - \gamma L)\langle x - x_*, Bx - Bx_*\rangle \\
&\leq \|x - x_*\|^2 - \mu\gamma(2 - \gamma L)\|x - x_*\|^2 \\
&= (1 - 2\gamma\mu + \mu\gamma^2 L)\|x - x_*\|^2 \ ,
\end{aligned}$$

□

where the first inequality follows from the $1/L$-cocoercivity of B, while the second one from the μ-strong monotonicity of B.

Thus $\|T_B x - T_B x_*\| \leq \sqrt{(1 - 2\gamma\mu + \mu\gamma^2 L)}\|x - x_*\|$ and since T_A is nonexpansive, we have that $\|Tx - Tx_*\| \leq \sqrt{(1 - 2\gamma\mu + \mu\gamma^2 L)}\|x - x_*\|$. Finally, S is quasi-ν-strongly monotone with $\nu = 1 - \sqrt{(1 - 2\gamma\mu + \mu\gamma^2 L)}$ for $\gamma < 2/L$.

Chapter 5

Communication Reduction in Distributed Optimization via Estimation of the Proximal Operator

5.1 Introduction

The benefit of the proposed asynchronous optimization approach presented in Chapter 4 is the increased number of updates (communication rounds) per unit of time, hence the inherent assumption is that communication is cheap in comparison to computation. In this chapter we reverse the trade-off and favor computation over communication, assuming that a communication round is (much) costlier than solving the agent's local optimization problem.

We consider the same distributed optimization framework, *i.e.,* a population of independent agents with a global coordinator. In the majority of the distributed optimization schemes, including the instances of the FBS iteration of Chapter 4, the agents' subproblems that need to be solved are cast as proximal minimization problems (2.5). In the course of the execution, the agents need to communicate the solutions to the proximal minimization subproblems to the coordinator, who will, in turn, manipulate the agents' objectives by broadcasting incentives that skew their local policies toward the global target.

In such multi-agent frameworks, extensive communication might be undesirable for a variety of reasons. Such reasons involve the existence of delays due to a weak network, the fact that the agents might run on energy-limited resources which are drained rapidly with frequent activations, or plainly speeding up computation by skipping insignificant agent updates. It would, therefore, be useful if the coordinator could 'guess' the optimizers of its agents and base the selection of the agent to update on the satisfaction of some criterion.

We propose a reduced communication framework for distributed optimization problems. This is achieved by estimating a convex set containing the solution of a proximal minimization problem. Construction of the set is based on the theory of the Moreau envelope function (2.7) and its important connections with the proximal operator. The structural properties of the Moreau envelope allow the coordinator to restrict the area of all possible optimizers within a convex set, explicitly described as the intersection of ellipsoids and iteratively refined every time that a communication

round occurs. Subsequently, the coordinator can make a guess regarding the solution of the agent's optimizer by choosing a value from the constructed set and spare a communication round provided that the guess is adequately good.

Before devising a communication protocol, we analyze the properties of the constructed ellipsoidal sets that contain the unknown gradient. We show that the sets can be recovered as limiting cases of ϵ-*subdifferential* sets (see Chapter 2, Definition 6) associated to functions that upper bound the Moreau envelope function and that they are *optimal* with respect to the tightest upper bound of the (unknown) Moreau envelope. As a byproduct of the performed analysis, we prove a fundamental inequality for differentiable convex functions by means of ϵ-subdifferential sets. We subsequently devise *communication tests* that enable adoption of the proposed framework for the PGM (projected and proximal gradient methods), for a randomized coordinate descent version of PGM, and for ADMM. In addition, we model the effect incurred by the lack of communication as an error in the solution of the optimization problem. By analyzing this result in the context of fixed-point iterations with errors, we prove *asymptotic convergence* of the sequence to an optimizer.

The chapter is organized as follows: Section 5.3 defines the problems we want to solve and sketches the followed approach. Section 5.4 contains the main analysis, showing how a fundamental gradient inequality, that can be used to explicitly construct a set that contains an optimizer, is related to the ϵ-subdifferential set of a quadratic upper bound to the envelope function and prove that it is optimal with respect to some metric. In Section 5.5 we couple the proposed scheme to several decomposition methods and devise certification tests that decide when the gradient set should be updated, namely when a communication round should be triggered in each case. Subsequently, convergence results are proven in Section 5.6. Finally, in Section 5.7 we provide evidence about the performance of the proposed scheme by solving another load sharing problem for microgrids. We conclude with some comments in Section 5.8.

5.2 Related work

A comprehensive theory of the Moreau envelope function can be found in [RW98], while the algorithmic aspects of the proximal iteration and its gradient descent interpretation are analyzed in [LS97]. Recently the authors of [PSB14; STP17] introduced new envelope functions so as to analyze more complicated splitting algorithms as unconstrained gradient iterations. In our proposed scheme, the convex set within which the possible optimizers reside can be essentially constructed by making use of a fundamental gradient inequality as will be shown in Section 5.4. Interestingly, a similar line of thinking has been followed in [THG17] in order to compute the exact worst-case performance of fixed-step first-order methods for unconstrained optimization of smooth convex functions. We are not aware of other works that seek to 'learn' the solutions to proximal minimization problems, or, in fact to any other optimization problem. In the context of communication avoidance in distributed optimization, the work [SDDGD17] proposes a scheme for reducing communication rounds for a stochastic version of the iterative soft thresholding algorithm. In a similar but rather more generic manner, the authors in [SFMTJ16; MKJSJRT17] propose communication-efficient distributed-optimization frameworks for large-scale machine learning applications.

5.3 Problem description

Consider the proximal minimization problem

$$\mathbf{prox}_{\gamma f}(z) = \underset{x\in\mathbb{R}^n}{\arg\min} \left\{ f(x) + \frac{1}{2\gamma}\|x - z\|^2 \right\} \ . \tag{5.1}$$

We are interested in a distributed optimization setting where each of a population of agents solves (privately) a proximal minimization problem of this form, and a global coordinator intervenes to steer the agents toward the minimizer of some (global) objective. A sequence of points $\{z^k\}$ at which problem (5.1) is solved is generated by the coordinator. Focusing on just one agent for the purpose of the analysis, we want to estimate, to the best possible accuracy, the optimizer of (5.1) for a given (arbitrary) sequence $\{z^k\}$, without having to solve the optimization.

Recall from (2.8) that the unique solution to the proximal minimization $x^\gamma(z) = \mathbf{prox}_{\gamma f}(z)$ can be written as

$$x^\gamma(z) = z - \gamma\nabla f^\gamma(z) \ . \tag{5.2}$$

It is evident from (5.2) that, given an arbitrary point z, the gradient $\nabla f^\gamma(z)$ is all that is needed in order to reconstruct the optimizer of (5.1). Estimating the exact gradient $\nabla f^\gamma(z)$ might not be possible, so we opt for constructing *a set of possible gradients at z and a way to evaluate the worst-case gradient (how far away one can be from the actual gradient) contained in the set.*

Our approach is to construct such a set and to continually refine it (shrink it) every time that an exchange occurs between the coordinator and the agent, namely every time that (5.1) is solved for a point z. We are going to refer to points at which the problem is actually solved, as *query points*. The coordinator's goal is to estimate the solution of (5.1) at a point of interest by using the solution at previous query points.

Assume the existence of a number of generated query points $z_j,\ j \in \mathcal{J}$. We proceed to construct a set that contains the gradient $\nabla f^\gamma(v)$ at the point of interest v in five distinct steps described below.

1. We first devise a family of (local) nested sets that all contain $\nabla f^\gamma(v)$. Each family is centered around its corresponding query point z_j.

2. We show that the family of sets can be represented by a convex inequality.

3. We find the smallest set in the family for each query point z_j.

4. We devise a (global) set within which $\nabla f^\gamma(v)$ resides by taking the intersection of the smallest sets for all z_j's.

5. We derive the relation between the global set and the tightest known function that upper bounds $f^\gamma(v)$.

6. We derive an approximation of the global set that can be used for numerical computations.

5.4 Estimating the solution to a proximal minimization problem

Quadratic upper bound Since we want to locate the unknown gradient $\nabla f^\gamma(v)$ in a set, what comes to mind is a notion of *subdifferential* of some function at v. The function we are about to use is a quadratic upper bound on the Moreau envelope, implied by the Lipschitz gradient property of f^γ.

Proposition 1. *The Moreau envelope f^γ admits a quadratic upper bound in terms of the linear approximation of f^γ based on the gradient at any $v \in \mathbb{R}^n$, i.e.,*

$$\overline{f}^\gamma(z;v) = f^\gamma(v) + \langle \nabla f^\gamma(v), z - v\rangle + \frac{1}{2\gamma}\|z - v\|_2^2, \quad z, v \in \mathbb{R}^n \ , \tag{5.3}$$

(see, e.g., [Ber15, Section 6.1]).

The quadratic uppper bounding function (5.3) will enable us to compute the desired set. Consider a query point, namely z_1. This point is associated to the quadratic approximation $\overline{f}^\gamma(z;z_1)$ that upper bounds f^γ. Looking at Figure 5.2, we observe that the point $(v, f^\gamma(v))$ lies below the epigraph of $\overline{f}^\gamma(z;z_1)$. Since we only have knowledge about the location of v and the function $\overline{f}^\gamma(z;z_1)$, we cannot say anything about the slope $\nabla f^\gamma(v)$. We can, however, construct a family of slopes depicted by the blue lines and demarcated by *the extreme slopes of $\overline{f}^\gamma(z;z_1)$ at v*. This brings up the notion of the ϵ-subdifferential set, given in Definition 6.

A family of local sets The ϵ-subdifferential set provides us with a way to restrict the set within which the actual gradient of the Moreau envelope at v, $\nabla f^\gamma(v)$, resides.

In order to do so, we introduce the following parameterization:

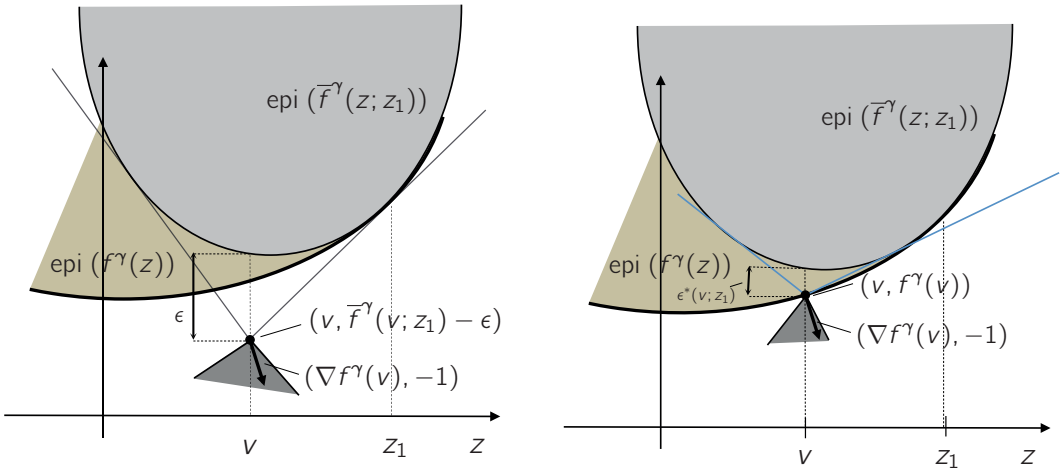
$$\epsilon^*(v;z_1) = \overline{f}^\gamma(v;z_1) - f^\gamma(v) \ . \tag{5.4}$$

Let us explain what equation (5.4) represents with the help of Figure 5.1. For any $\epsilon > 0$ denoting the vertical distance from $\overline{f}^\gamma(v;z_1)$ at v, we can draw the extreme slopes of all the hyperplanes that are tangent to the epigraph of $\overline{f}^\gamma(z;z_1)$ and generate an outer approximation of $\overline{f}^\gamma(z;z_1)$. The endpoints of these ϵ-subdifferential sets are depicted with grey lines. Equation (5.4) defines *the smallest ϵ-subdifferential set for which we can show that the gradient $\nabla f^\gamma(v)$ resides inside.* We will carry out the remaining analysis for the case that only one query point, namely z_1, has been generated, while the point of interest will be $z = v$.

Proposition 2. *The following statements characterize the set in which $\nabla f^\gamma(v)$ is located.*

1. *The gradient of the Moreau envelope at a point v is contained in the $\epsilon^*(v;z_1)$-subdifferential set of the upper-bounding quadratic function $\overline{f}^\gamma(z;z_1)$ evaluated at v and constructed around the query point z_1, i.e.,*
$$\nabla f^\gamma(v) \in \partial_{\epsilon^*(v;z_1)} \overline{f}^\gamma(v;z_1) \ .$$
2. *The $\epsilon^*(v;z_1)$-subdifferential set is nonempty, convex and compact.*

Proof: The proof is deferred to Appendix 5.9.1. □



(a) An ϵ-subdifferential set represented as a cone.

(b) The smallest ϵ-subdifferential set.

Figure 5.1: The epigraph of the envelope function $f^{\gamma}(z)$ and that of the quadratic upper bounding function $\overline{f}^{\gamma}(z; z_1)$ are illustrated. For any point centered at v that lies ϵ below the epigraph of $\overline{f}^{\gamma}(z; z_1)$, the ϵ-subdifferential set is depicted as the normal cone of the set constructed by the tangent hyperplanes of epi $(\overline{f}^{\gamma}(z; z_1))$, depicted with the grey lines, at $(v, \overline{f}^{\gamma}(v; z_1) - \epsilon)$. The actual gradient $\nabla f^{\gamma}(v)$ is contained in all $\partial_{\epsilon}\overline{f}^{\gamma}(v; z_1)$ for $\epsilon \geq \epsilon^*(v; z_1)$. As ϵ is getting smaller, the subdifferential sets shrink, and converge to $\partial_{\epsilon^*(v;z_1)}\overline{f}^{\gamma}(v; z_1)$, depicted by the normal cone to the set illustrated by the blue lines in the right figure.

The proposition suggests the existence of a set that contains the gradient $\nabla f^\gamma(v)$, namely the set $\partial_{\epsilon^*(v;z_1)}\overline{f}^\gamma(v;z_1)$ which describes a family of admissible slopes.

Explicit representation of the local sets We will, subsequently, construct $\partial_{\epsilon^*(v;z_1)}\overline{f}^\gamma(v;z_1)$ in an explicit way. To this end, we introduce the function

$$\overline{f}_z^\gamma(d;z_1) = \overline{f}^\gamma(z+d;z_1) - \overline{f}^\gamma(z;z_1), \quad \forall d \in \mathbb{R}^n$$

and its conjugate [Roc70, Chapter 23]

$$(\overline{f}_z^\gamma)^\star(g;z_1) = (\overline{f}^\gamma)^\star(g;z_1) + \overline{f}^\gamma(z;z_1) - \langle z, g\rangle \ . \tag{5.5}$$

The following Proposition holds.

Proposition 3. *The ϵ-subdifferential set of $\overline{f}^\gamma(z;z_1)$, evaluated at $z=v$, is given by*

$$\partial_\epsilon \overline{f}^\gamma(v;z_1) = \left\{ g \ \mid \ (\overline{f}_v^\gamma)^\star(g;z_1) \leq \epsilon \right\} \ , \tag{5.6}$$

for any $\epsilon > 0$.

Proposition 3 relates the ϵ-subdifferential of a function with the epigraph of its conjugate. More detailed discussions can be found in [Roc70, Chapter 23], [Ber15, Section 6.7.1]. This equivalence allows us to construct the $\epsilon^*(v;z_1)$-subdifferential set described in Proposition 2 provided that (5.5) is computable. We prove below that this is indeed the case.

Proposition 4. *The set $\partial_\epsilon \overline{f}^\gamma(v;z_1)$ is given by*

$$\partial_\epsilon \overline{f}^\gamma(v;z_1) = \left\{ g \ \mid \ (\gamma/2)\|g - \nabla f^\gamma(z_1)\|_2^2 - \langle g - \nabla f^\gamma(z_1), v - z_1\rangle + (1/2\gamma)\|z_1 - v\|_2^2 \leq \epsilon \right\} \ . \tag{5.7}$$

Proof: The conjugate of (5.3) can be easily computed using the definition since the function is a convex quadratic. The claim follows directly from computing it and substituting the result in (5.5). □

The smallest local set Propositions 2 and 4 tell us that the gradient $\nabla f^\gamma(v)$ is contained within any of the 2-norm balls associated to the query point z_1 and described by (5.7). The radius of the balls varies with ϵ. A question that naturally arises is related to the size of this set, namely, *whether we can find the smallest set in the family described by* (5.7).

Proposition 5. *Given a point $v \in \mathbb{R}^n$ and a distance $\epsilon < \epsilon^*(v;z_1)$, there exists a function $f(x)$ with an associated Moreau envelope $f^\gamma(z)$ and the corresponding local upper-bounding function $\overline{f}^\gamma(v;z_1)$ centered at some query point z_1, such that $\nabla f^\gamma(v) \notin \partial_\epsilon \overline{f}^\gamma(v;z_1)$.*

Proof: The proof can be found in Appendix 5.9.2. □

Proposition 5 states that plugging $\epsilon^*(v; z_1)$ in (5.7) results in the smallest set of the family containing $\nabla f^\gamma(v)$ for a general function f. The set can be written as

$$\begin{aligned}\mathcal{G}(v; z_1) :&= \partial_{\epsilon^*(v;z_1)} \overline{f}^\gamma(v; z_1) \\ &= \left\{ g \;\middle|\; (\gamma/2)\|g - \nabla f^\gamma(z_1)\|_2^2 - \langle g, v - z_1 \rangle \leq f^\gamma(z_1) - f^\gamma(v) \right\} \ . \end{aligned} \tag{5.8}$$

A global gradient set To recap, we showed that $\nabla f^\gamma(v)$ is contained in the family of ϵ-subdifferential sets $\partial_\epsilon \overline{f}^\gamma(v; z_1)$ constructed around v and associated to the query point z_1 (Proposition 2), that these sets can be explicitly described (Proposition 4) and that the smallest set of the family is recovered for the choice of $\epsilon = \epsilon^*(v; z_1)$ from (5.4) (Proposition 5).

Making use of (5.8), the set of possible gradients of f^γ at v can be described by

$$\begin{aligned}\mathcal{G}(v) :&= \bigcap_{j \in \mathcal{J}} \partial_{\epsilon^*(v;z_j)} \overline{f}^\gamma(v; z_j) \\ &= \bigcap_{j \in \mathcal{J}} \left\{ g \;\middle|\; \gamma\|g - \nabla f^\gamma(z_j)\|_2^2 - \langle g - \nabla f^\gamma(z_j), v - z_j \rangle \leq 0 \right\} \ , \end{aligned} \tag{5.9}$$

where $\mathcal{J}$ is a set of indices cooresponding to generated query points. As a consequence, the set of potential gradients can be described as an intersection of a finite number of 2-norm balls. It also trivially holds that $\bigcap_{j=1}^{\infty} \mathcal{G}(v; z_j) = \{\nabla f^\gamma(v)\}$, *i.e.,* upon complete construction of the envelope, the above intersection reduces to the singleton set $\{\nabla f^\gamma(v)\}$.

Relation to tightest upper bounding function As a last step, we will derive the relation between $\mathcal{G}(v)$ *and the ϵ-subdifferential set associated to the tightest constructed upper bound of f^γ at v.*

To this end we introduce the function corresponding to the convex hull of the generated upper bounding quadratics

$$\overline{f}^\gamma(z \mid \mathcal{J}) = \operatorname{conv}\{\overline{f}^\gamma(z; z_j) \mid j \in \mathcal{J}\} \ . \tag{5.10}$$

Note the abuse of notation in (5.10) since it actually describes a set and not a function. To define it properly, we should write

$$\overline{f}^\gamma(z \mid \mathcal{J}) = \inf_{\substack{\sum_{j\in\mathcal{J}} \theta_j z_j = z, \\ \theta_j \geq 0,\, \sum_{j\in\mathcal{J}} \theta_j = 1}} \sum_{j \in \mathcal{J}} \theta_j \overline{f}^\gamma(z; z_j) \ ,$$

according to [Roc70, Theorem 5.6], however we will keep using (5.10) for simplicity.

The function is illustrated in Figure 5.2 for two scalar functions. The dependence on $\mathcal{J}$ demonstrates that the function is refined (becomes tighter) with the generation of new query points. The following result holds:

Theorem 8. *Given a number of generated query points $\{z_j\}$, $j \in \mathcal{J}$ and $v \in \mathbb{R}^n$,*

$$\mathcal{G}(v) = \partial_{\epsilon^*(v|\mathcal{J})} \overline{f}^\gamma(v \mid \mathcal{J}) \ ,$$

where $\epsilon^*(v \mid \mathcal{J}) = \overline{f}^\gamma(v \mid \mathcal{J}) - f^\gamma(v)$.

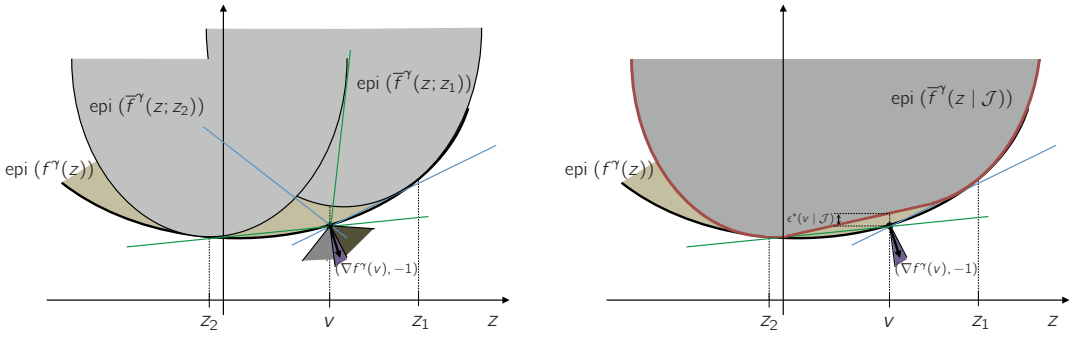


Figure 5.2: Assuming two query points z_1 and z_2, $\nabla f^{\gamma}(v)$ can be located in the intersection of the two normal cones, which is sketched in purple color. In addition, the function $\overline{f}^{\gamma}(z \mid \mathcal{J}) = \operatorname{conv}\{\overline{f}^{\gamma}(z; z_1), \overline{f}^{\gamma}(z; z_2)\}$ is drawn in deep red color. It is evident that the normal cone corresponding to the intersection of the two sets (left) is identical to the set $\partial_{\epsilon^*(v|\mathcal{J})}\overline{f}^{\gamma}(v \mid \mathcal{J})$ on the right.

Proof: The proof can be found in Appendix 5.9.3. □

An approximation of the global set A problematic aspect of the global set $\mathcal{G}(v)$ when it comes to computing it, is the dependence on the unknown quantity $f^{\gamma}(v)$ in (5.8). We want to drop this dependency by approximating $\mathcal{G}(v)$ with another set, namely $\hat{\mathcal{G}}(v)$.

To that end, we use the fundamental inequality (see, *e.g.,* [Ber15, Proposition 6.1.9])

$$f^{\gamma}(z_1) - f^{\gamma}(v) \leq -\langle \nabla f^{\gamma}(z_1), v - z_1 \rangle - \tfrac{\gamma}{2}\|\nabla f^{\gamma}(v) - \nabla f^{\gamma}(z_1)\|_2^2 \ .$$

Using the inequality above in the right-hand side of (5.8) leaves us, still, with another unknown quantity, *i.e.,* $\nabla f^{\gamma}(v)$. A reasonable thought would be to replace $\nabla f^{\gamma}(v)$ with g and look for a $g \approx \nabla f^{\gamma}(v)$ that satisfies the above inequality. By doing so, we end up with the inequality

$$\hat{\mathcal{G}}(v; z_1) := \{g \mid \gamma\|g - \nabla f^{\gamma}(z_1)\|_2^2 - \langle g - \nabla f^{\gamma}(z_1), v - z_1 \rangle \leq 0\} \ . \tag{5.11}$$

Interestingly, (5.11) is the necessary condition for convexity and Lipschitz continuity of the gradient of f^{γ}, the so-called *co-coercivity property* of the gradient, applied to the pair of points (z_1, v). As such, the set $\hat{\mathcal{G}}(v; z_1)$ is a *valid approximation of* (5.8) in the sense that it contains the unknown gradient $\nabla f^{\gamma}(v)$. In addition, it does not depend on any unknown quantities. The (approximate) global set is given by

$$\hat{\mathcal{G}}(v) = \bigcap_{j \in \mathcal{J}} \hat{\mathcal{G}}(v; z_j) \ .$$

Remark 11. *We cannot say anything about the relative size of the sets $\hat{\mathcal{G}}(v)$ and $\mathcal{G}(v)$. As a matter of fact, we can devise examples that show that no set is strictly contained in the other. Therefore, we simply use $\hat{\mathcal{G}}(v)$ in the computations since we can explicitly compute it.*

5.5 Communication

Let us now move back to the original distributed setting. Consider an optimization problem of the form

$$\text{minimize} \quad h(x) + \sum_{i=1}^{N} f_i(x_i) \ , \tag{5.12}$$

where $f_i : \mathbb{R}^{n_i} \mapsto \mathbb{R} \cup \{+\infty\},\ i = 1, \ldots, N$ are convex functions private to the corresponding agents and $h : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$, $n = \sum_{i=1}^{N} n_i$, is the convex objective of the coordinator.

The aim is to devise communication tests based on which the coordinator will decide whether to trust its approximate optimizer or to query the agent. Let us denote with $\mathcal{G}_i(v_i; z_{i,j}),\ i = 1, \ldots, N$ the set given by (5.11), associated to agent i. The coordinator can keep in its memory a set $\mathcal{G}_i(v_i) := \bigcap_{j \in \mathcal{J}_i} \mathcal{G}_i(v_i; z_{i,j})$, $\mathcal{J}_i$ being the number of queries generated up to the current algorithmic iteration, for every agent $i = 1, \ldots, N$. The set is updated whenever a communication round occurs. With every query point, $\mathcal{G}_i(v_i)$ is augmented by one more inequality. The challenge that needs to be addressed is twofold:

1. Compute or locate an approximate minimizer of the proximal minimization problem of agent i.

2. Verify the suitability of the computed solution, *i.e.,* design a certification test based on which the decision to communicate or not is taken.

We are going to discuss tests for three algorithmic approaches that solve (5.12), namely the *projected gradient method, the proximal gradient method and the alternating direction method of multipliers (Section 2.2).*

Projected Gradient Method (ProjGM) Consider the case when $f_i(x_i) = \delta(x_i \mid \mathcal{X}_i)$, and $\mathcal{X}_i,\ i = 1, \ldots, N$ are convex compact sets, h is convex differentiable with L-Lipschitz continuous gradient, and a relaxed version of the Projected Gradient Method is used in order to solve (5.12), namely the iteration becomes

$$x_i^{k+1} = \mathcal{P}_{\mathcal{X}_i}(x_i^k - c\nabla_i h(x^k)) \ , \tag{5.13}$$

with a stepsize $c \in (0, 2/L)$.

Observe that

$$\mathcal{P}_{\mathcal{X}_i}(v_i^k) = v_i^k - \gamma \nabla f_i^{\gamma}(v_i^k) \ ,$$

where $v_i^k = x_i^k - c\nabla_i h(x^k)$ and $\gamma > 0$. The aim is to choose a point $g_i^k \in \mathcal{G}_i(v_i^k)$ such that $g_i^k \approx \nabla f_i^{\gamma}(v_i^k)$.

It follows from the descent property of the projected gradient method that $\langle \nabla h(x^k), x^{k+1} - x^k \rangle < 0$ (see, *e.g.,* [Ber15, Proposition 6.1.1]). Consequently, the worst-case optimizer can be found be solving the following Quadratically Constrained Quadratic Program (QCQP):

$$\begin{array}{ll} \text{maximize} & \langle \nabla h(x^k), v^k - x^k - \gamma g \rangle =: \mathcal{T}(g, v^k) \\ \text{subject to} & g \in \mathcal{G}(v^k) \ , \end{array} \tag{5.14}$$

with variable $g \in \mathbb{R}^n$. Note that the underlying assumption behind solving (5.14) is that the point x^k is feasible, which is not necessarily the case if (5.13) has been generated by the coordinator.

After the test $\mathcal{T}(g, v^k) \leq 0$ has been passed, *any* point $g \in \mathcal{G}(v^k)$ can be chosen.

Proximal Gradient Method (PGM) In the more general case that f_i encapsulates a convex objective restricted within some convex set and h is convex differentiable with L-Lipschitz continuous gradient, iteration (5.13) becomes the relaxed proximal iteration

$$x_i^{k+1} = \mathbf{prox}_{\gamma f_i}(x_i^k - \gamma \nabla_i h(x^k)) \ . \tag{5.15}$$

Similar to the projected gradient method, an inherent property of the proximal gradient scheme is the objective value decrease per iteration (see, *e.g.,* [Ber15, Proposition 6.3.2]).

$$h(x^{k+1}) + \sum_{i=1}^{N} f_i(x_i^{k+1}) \leq h(x^k) + \sum_{i=1}^{N} f_i(x_i^k) - \frac{1}{2\gamma}\|x^{k+1} - x^k\|^2, \quad \forall \gamma \in (0, 1/L] \ . \tag{5.16}$$

Since the exact value of $f_i(x_i^{k+1})$ cannot be known, it is impossible to directly apply (5.16) in order to evaluate a worst case gradient. We can, however, devise a more conservative test that results in a sufficient condition for communication.

Proposition 6. *The optimization problem*

$$\begin{array}{ll} \textit{maximize} & h(v^k - \gamma g) - \langle g, v^k - x^k \rangle =: \mathcal{T}(g, v^k) \\ \textit{subject to} & g \in \mathcal{G}(v^k) \end{array} \tag{5.17}$$

results in a sufficient decrease condition if

$$\mathcal{T}(g, v^k) \leq h(x^k) + \sum_{i=1}^{N} f_i(x_i^k) - \sum_{i=1}^{N} \min_{j \in \mathcal{J}} \{\overline{f}_i^{\gamma}(v_i^k; z_{i,j})\} - \frac{1}{2\gamma}\|v^k - x^k\|^2 \ . \tag{5.18}$$

Proof: The proof can be found in Appendix 5.9.4. □

Except for the case that h is an affine function, problem (5.17) is nonconvex. In order to solve it we can resort either to some heuristic or an algorithm that will result in a local optimizer. In the case that h is quadratic, a solution to a (conservative) convex approximation of (5.17) can be recovered by solving two Semidefinite Programs (SDP's). We first compute the maximum volume ellipsoid that lies in $\mathcal{G}(v^k)$, which can be cast as an SDP. Enlarging this by a factor of n about its center results in an ellipsoid that covers $\mathcal{G}(v^k)$ (see [BV04, Section 8.4]). Subsequently, we solve the convex maximization problem of maximizing $\mathcal{T}(g, v^k)$ over the covering ellipsoid, which can be exactly relaxed into another SDP [BV04, Appendix B.1]. It should be noted, of course, that the additional layers of conservativeness that have been added might result in the test not being passed, rendering it practically useless for communication savings.

Alternating Direction Method of Multipliers (ADMM) Let us finally consider the most general case of solving (5.12) when h is not necessarily differentiable and f_i encapsulates a convex

objective restricted within some convex set. ADMM comprises the iterations:

$$\begin{aligned} x^{k+1} &= \arg\min_{x} \left\{ h(x) + (\rho/2)\|x - y^k + (1/\rho)\lambda^k\|^2 \right\} \\ y_i^{k+1} &= \arg\min_{y_i} \left\{ f_i(y_i) + (\rho/2)\|y_i - x_i^{k+1} - (1/\rho)\lambda_i^k\|^2 \right\} \\ \lambda^{k+1} &= \lambda^k + \rho(x^{k+1} - y^{k+1}) \ , \end{aligned} \tag{5.19}$$

$\rho > 0$. The splitting described in (5.19) is achieved by introducing a copy $y = x$ and the associated dual variables λ. Observe that the y-minimization step constitutes local proximal minimization problems of the form

$$\begin{aligned} y_i^{k+1} &= \mathbf{prox}_{\frac{1}{\rho} f_i}(\underbrace{x_i^{k+1} + (1/\rho)\lambda_i^k}_{v_i^k}) \\ &= v_i^k - \gamma \nabla f_i^{\gamma}(v_i^k), \quad \gamma = 1/\rho \ . \end{aligned} \tag{5.20}$$

For agent i, we know that any point lying in the interior of the set $\mathcal{G}_i(v_i^k)$ is a potential candidate. Assuming that we have such a point, we need to decide whether the resulting g_i^k checks out as a valid gradient estimate. In order to do so, we propose a heuristic test based on the progress of the ADMM residual. The residual is defined with respect to the consensus condition $x = y$, *i.e.,* $s^k = \|x^k - y^k\|$, and the communication test reads:

$$\mathcal{T}(s^k, s^{k+1}) := s^k - s^{k+1} < 0 \ . \tag{5.21}$$

Monotonic decrease of the residual is neither a necessary nor a sufficient condition for the convergence of ADMM. It can be used, nevertheless, as a certification test for assessing the validity of the approximate proximal minimizer.

Sparing further details, if we were to use (5.21) so as to follow a worst-case approach, as we did with ProjGM and PGM, we would end up with a nonconvex problem. We will, instead, use a heuristic approach to recover *any* $g \in \mathcal{G}$.

More specifically, any guess for $\nabla f_i(v_i^k)$ denoted by g_i^k will give rise to an approximate minimizer $\hat{y}_i^{k+1} = v_i^k - \gamma g_i^k$. This minimizer is used, in turn, to update the dual variable $\hat{\lambda}_i^{k+1}$. After performing these updates for all the agents, inequality (5.21) is checked for $x = x^k$. If it holds, no communication is necessary, while in the opposite case all the agents need to communicate.

The rational behind checking inequality (5.21) for $x = x^k$ is to assess whether any progress has been made with regard to reaching optimality and in comparison to the previous pair (x^k, λ^k). If not, there is an indication that (some of) the approximate optimizers $\hat{y}_i^{k+1}$ were not 'in the right direction', thereby it is better to correct via communication. In addition, the test only involves h in its evaluation, and thus can be performed at the coordinator's level with no extra knowledge about the agents.

Let us summarize what we have so far: We can either compute the worst-case optimizer g for an agent and check whether it passes a test, or make a guess for an optimizer and perform a posteriori validation through a test. The choice depends mainly on how easy it is to compute the worst-case gradient. As we have indicated, this is possible for ProjGM by solving (5.14), and potentially too

conservative (and too expensive) for both PGM (5.17) and ADMM.

5.5.1 Non worst-case candidates

Solving the convex programs (5.14) and (5.17) results in the acquisition of worst-case optimizers for the projected (and proximal) gradient methods. We present below two methods that choose a possible gradient $g^k \in \mathcal{G}(v^k)$ in case that we cannot, or do not want to adopt such conservative solutions.

Projection of the 'closest' gradient onto the feasible set. A reasonable guess for a 'good' g_i^k would be to identify the hyperplane that is closest to f_i^γ at v_i^k, namely to find the index

$$j^* = \arg\max_{j \in \mathcal{J}_i} \left\{ f_i^\gamma(z_{i,j}) + \langle \nabla f_i^\gamma(z_{i,j}), v_i^k - z_{i,j} \rangle \right\} \ , \tag{5.22}$$

expecting that the gradient of the envelope won't have changed much if the points z_{i,j^*} and v_i^k are close to each other. If $\nabla f_i^\gamma(z_{i,j^*}) \in \mathcal{G}_i(v_i^k)$, then $g_i^k = \nabla f_i^\gamma(z_{i,j^*})$, otherwise the projection of $\nabla f_i^\gamma(z_{i,j^*})$ onto $\mathcal{G}_i(v_i^k)$ is taken. The latter is achieved by solving the QCQP

$$\begin{array}{ll} \text{minimize} & \|g_i - \nabla f_i^\gamma(z_{i,j^*})\|_2 \\ \text{subject to} & g_i \in \mathcal{G}_i(v_i^k; z_{i,j}),\ j = 1, \ldots, \mathcal{J}_i \ , \end{array} \tag{5.23}$$

with variable $g_i \in \mathbb{R}^{n_i}$.

Chebyshev center. Another approach would be to update x_i^{k+1} based on the g_i^k that resides in the *Chebyshev center* of the set $\mathcal{G}_i(v_i^k)$, *i.e.,* g_i^k is the center of the largest ball inscribed in the set $\mathcal{G}_i(v_i^k)$. The center can be computed by solving the SDP [BV04, Section 8.5.1]:

$$\begin{array}{ll} \text{max.} & R_i \\ \text{s.t.} & \begin{bmatrix} -\lambda_{i,j} - c_{i,j} + b_{i,j}^\top A_i^{-1} b_{i,j} & 0 & (g_i + A_i^{-1} b_{i,j})^\top \\ 0 & \lambda_{i,j} I & R_i I \\ g_i + A_i^{-1} b_{i,j} & R_i I & A_i^{-1} \end{bmatrix} \succeq 0, \\ & j = 1, \ldots, \mathcal{J}_i \ , \end{array} \tag{5.24}$$

with variables $R_i \in \mathbb{R}$, $\lambda_{i,j} \in \mathbb{R}$, $g_i \in \mathbb{R}^{n_i}$, while

$$\begin{aligned} A_i &= \gamma I,\ i = 1, \ldots, N \\ b_{i,j} &= -\gamma \nabla f_i^\gamma(z_{i,j}) - (1/2)(v_i^k - z_{i,j}), \\ c_{i,j} &= \nabla f_i^\gamma(z_{i,j})^\top (\gamma \nabla f_i^\gamma(z_{i,j}) + v_i^k - z_{i,j}) \ . \end{aligned}$$

5.5.2 The algorithmic schemes

Putting together the results of the preceding sections, we describe below reduced communication schemes that solve problem (5.12) using the discussed algorithms.

Algorithm 9 Distributed Optimization with Estimated Proximal Operator for PGM (ProjGM)

Require: $x_i^0 \in \mathbb{R}^{n_i}$, $\mathcal{G}_i = \varnothing$, suitable c and γ. Iteration counter is set to $k = 0$, $k_{\text{stop}} > 0$.

while $k < k_{\text{stop}}$ **do**
  Compute $v^k = x^k - c\nabla h(x^k)$ // Coordinator
  Solve (5.17) (or (5.14)) // Coordinator
  if $\mathcal{T}(g, v^k) < 0$ **then**
    Choose any $g \in \mathcal{G}(v^k)$, set $g^k = g$ // Coordinator
    Compute $x^{k+1} = v^k - \gamma g^k$ // Coordinator
  else
    Transmit v_i^k to Agent i, $i = 1, \ldots, N$ // Coordinator
    Solve (5.15) (or (5.13)) // Agent i
    Transmit x_i^{k+1} to the Coordinator // Agent i
    Update set $\mathcal{G}_i$ with x_i^{k+1} // Coordinator
  end if
end while
$k \leftarrow k + 1$

Algorithm 10 Distributed Optimization with Estimated Proximal Operator for ADMM

Require: $x_i^0 \in \mathbb{R}^{n_i}$, $\mathcal{G}_i = \varnothing$, $i = 1, \ldots, N$, constant $\rho > 0$. Iteration counter is set to $k = 0$, $k_{\text{stop}} > 0$.

while $k < k_{\text{stop}}$ **do**
  Compute x^{k+1} from (5.19) // Coordinator
  Compute $v^k = \tilde{x}^{k+1} + (1/\rho)\lambda^k$ // Coordinator
  Solve (5.23) $i = 1, \ldots, N$ to get $g^k = g$ // Coordinator
  Compute $y^{k+1} = v^k - (1/\rho)g^k$ // Coordinator
  Compute λ^{k+1} from (5.19) // Coordinator
  Compute $s^{k+1} = \|x^{k+1} - y^{k+1}\|$ // Coordinator
  if $\mathcal{T}(s^k, s^{k+1}) < 0$ **then**
    Transmit v_i^k to Agent i, $i = 1, \ldots, N$ // Coordinator
    Compute y_i^{k+1} from (5.19) // Agent i
    Compute λ_i^{k+1} from (5.19) // Agent i
    Transmit $(y_i^{k+1}, \lambda_i^{k+1})$ to the Coordinator // Agent i
    Update sets of query points with $z_{i,j} = v_i^k$ and $\mathcal{G}_i(v_i) = \mathcal{G}_i(v_i) \cap \mathcal{G}_i(v_i; z_{i,j})$ with $\nabla f_i^\gamma(v_i^k) = \rho(v_i^k - y_i^{k+1})$, $i = 1, \ldots, N$ // Coordinator
  end if
end while
$k \leftarrow k + 1$

In the way that both Algorithms 9 and 10 are written, if the test is not passed, all the agents will need to communicate. One could, however, design different communication schemes. For example, the coordinator could start transmitting to the agents sequentially until the test is passed. Indeed, any subset of agents could be selected based on some specified criterion, as long as the test is passed. To this end, we present in Algorithm 11 below a *randomized coordinate descent* version of the proposed scheme for P(roj)GM.

Algorithm 11 Randomized Coordinate Descent with Estimated Proximal Operator for PGM (ProjGM)

Require: $x_i^0 \in \mathbb{R}^{n_i}$, $\mathcal{G}_i = \varnothing$, suitable c and γ, discrete probability distribution $(p_1, \ldots, p_N)$, $\sum_{i=1}^N p_i = 1$, $p_i > 0$. Iteration counter is set to $k = 0$, $k_{\text{stop}} > 0$.

while $k < k_{\text{stop}}$, choose agent i_k with probability $p_{i_k} := \mathbb{P}[i_k = i] = p_i$, **do**
  Compute $v_{i_k}^k = x_{i_k}^k - c\nabla_{i_k} h(x^k)$ // Coordinator
  Solve (5.14) (or (5.17)) // Coordinator
  if $\mathcal{T}(g, v^k) < 0$ **then**
    Choose any $g_{i_k} \in \mathcal{G}_{i_k}$, set $g_{i_k}^k = g_{i_k}$ // Coordinator
    Compute $x_{i_k}^{k+1} = v_{i_k}^k - \gamma g_{i_k}^k$ // Coordinator
  else
    Transmit $v_{i_k}^k$ to Agent i // Coordinator
    Solve (5.15) (or (5.13)) // Agent i_k
    Transmit $x_{i_k}^{k+1}$ to the Coordinator // Agent i_k
    Update set $\mathcal{G}_{i_k}$ with $x_{i_k}^{k+1}$ // Coordinator
  end if
end while
$k \leftarrow k + 1$

5.6 Convergence

In this section we give conditions that ensure convergence of the scheme described by Algorithms 10 and 11. The key point is to write the approximate proximal minimization step as a regular proximal iteration with additive error. Consequently, the analysis follows by employing results from inexact fixed point iteration theory.

In order to have a unified convergence framework, all three splitting methods presented above (ProjGM, PGM and ADMM) are cast as KM iterations described in Section 2.15 with additive errors. Consequently, and to be able to prove any convergence result, the iterations should be *under-relaxed* with some parameter $\eta^k \in [0, 1]$. Let us summarize the relaxed versions of the three schemes, (5.13), (5.15) and (5.19), below.

1. Projected Gradient Method

$$x_i^{k+1} = (1 - \eta^k)x_i^k + \eta^k \mathcal{P}_{\mathcal{X}_i}(x_i^k - c\nabla_i h(x^k)) \ . \tag{5.25}$$

2. Proximal Gradient Method

$$x_i^{k+1} = (1-\eta^k)x_i^k + \eta^k \, \mathbf{prox}_{\gamma f_i}(x_i^k - \gamma \nabla_i h(x^k)) \ . \tag{5.26}$$

3. Alternating Direction Method of Multipliers

$$\begin{aligned} x^{k+1} &= \mathbf{prox}_{\frac{1}{\rho}h}(y^k - (1/\rho)\lambda^k) \\ \tilde{x}^{k+1} &= \eta^k x^{k+1} + (1-\eta^k)y^k \\ y_i^{k+1} &= \mathbf{prox}_{\frac{1}{\rho}f_i}(\tilde{x}_i^{k+1} + (1/\rho)\lambda_i^k) \\ \lambda^{k+1} &= \lambda^k + \rho(\tilde{x}^{k+1} - y^{k+1}) \ . \end{aligned} \tag{5.27}$$

We start by analyzing ADMM and subsequently move to the coordinate descent PGM iteration.

5.6.1 Convergence of ADMM

For analysis purposes, we are going to express ADMM as a *Douglas-Rachford Splitting (DRS) iteration of the dual variables* λ. The equivalence of the two algorithms can be found in, *e.g.,* [EB92], while a detailed derivation of ADMM from DRS is given in Appendix A. By using standard conjugate duality (Section 2.1), we can rewrite (5.12) as

$$\text{minimize} \quad H(\lambda) + F(\lambda) \ , \tag{5.28}$$

where λ is the dual variable in (5.19), $H(\lambda) := h^\star(-\lambda)$ and $F(\lambda) := \sum_{i=1}^N f_i^\star(\lambda_i)$. Applying the ADMM iteration (5.27) to solve (5.12) is equivalent to applying the following DRS iteration to problem (5.28):

$$\begin{aligned} r^{k+1} &= \mathbf{prox}_{\rho H}(2\lambda^k - z^k) \\ z^{k+1} &= z^k + \eta^k(r^{k+1} - \lambda^k) \\ \lambda^{k+1} &= \mathbf{prox}_{\rho F}(z^{k+1}) \ , \end{aligned} \tag{5.29}$$

where

$$\mathbf{prox}_{\rho F}(z^{k+1}) := (\mathbf{prox}_{\rho f_1^\star}(z_1^{k+1}), \ldots, \mathbf{prox}_{\rho f_N^\star}(z_N^{k+1})).$$

We will now illustrate that the approximate minimizer y_i^{k+1} of (5.27) can be expressed as the solution of a proximal minimization problem with some additive error. Let us first express the approximate iteration $\hat{y}_i^{k+1} = v_i^k - \gamma g_i^k$ as *an inexact proximal minimization solve*. To this end, we introduce the error sequence

$$\{e^k\} = \{g^k - \nabla f^\gamma(v^k)\} \ , \tag{5.30}$$

with $\nabla f^\gamma(v^k) = (\nabla f_1^\gamma(v_i^k), \ldots, \nabla f_N^\gamma(v_N^k))$. Using (5.30), the following result holds:

Proposition 7. *Algorithm 10 can be expressed as the inexact DRS iteration*

$$\begin{aligned} r^{k+1} &= \mathbf{prox}_{\rho H}(2\lambda^k - z^k) \\ z^{k+1} &= z^k + \eta^k(r^{k+1} - \lambda^k) \\ \lambda^{k+1} &= \mathbf{prox}_{\rho F}(z^{k+1}) + e^k \ , \end{aligned} \tag{5.31}$$

with e^k defined in (5.30).

Proof: The proof is deferred to Appendix 5.9.5. □

Capitalizing on this result, the inexact DRS iteration (5.31) can be analyzed in the more general context of inexact fixed point iterations, analyzed in detail in [LFP16, Section 5.2]. To this end, the iteration can be written in the more compact notation

$$z^{k+1} = z^k + \eta^k(Tz^k + \zeta^k - z^k) \ , \tag{5.32}$$

by introducing the operator

$$T = \frac{1}{2}\left(\mathbf{refl}_{\rho H}\,\mathbf{refl}_{\rho F} + I\right),$$

with $T : \mathbb{R}^n \mapsto \mathbb{R}^n$ and

$$\zeta^k = \frac{1}{2}\left(\mathbf{refl}_{\rho H}\left(\mathbf{refl}_{\rho F}(z^k) + 2e^k\right) + z^k\right) - Tz^k \ . \tag{5.33}$$

With iteration (5.32) in place, we are ready to give conditions for the convergence of the DRS iteration (5.31) and, consequently, of (the under-relaxed variant of) Algorithm 10.

Theorem 9. *[CP07, Theorem 2] Consider Algorithm 10 with $x = \tilde{x}$, where $\tilde{x}$ is defined in* (5.27). *The multiplier sequence $\{\lambda^k\}$ generated by the algorithm will converge to an optimizer of* (5.28) *provided that $\sum_{k=1}^{\infty} \eta^k \|e^k\| < \infty$ and $\sum_{k=1}^{\infty} \eta^k(2 - \eta^k) = \infty$.*

Remark 12. *The sequence $\{\|e^k\|\}$ is bounded since $\|e^k\| \leq N \max_{1\leq i\leq N} \|g_i^k - \nabla f_i^{\gamma}(v_i^k)\|$. Since condition* (5.21) *will be enforcing communication until feasibility is achieved, more query points will be generated and $\lim_{j\to\infty} \bigcap_{j\in\mathcal{J}_i} \hat{\mathcal{G}}_i(v_i; z_{i,j}) = \{\nabla f_i^{\gamma}(v_i)\}$. Since we do not know anything about the rate of convergence, we need to resort to a generic sequence $\{\eta^k\}$ that would enforce convergence, e.g., $\eta^k = 1/(k^2\|e^k\|)$. The quantity $\|e^k\|$ can be (over-)approximated by taking, e.g., the smallest ellipsoid of the ones that are intersected and compute its diameter.*

Remark 13. *The convergence result presented above is only useful for analysis purposes, since in practice we would not use a diminishing stepsize which typically results in very slow convergence.*

5.6.2 Convergence of randomized coordinate descent PGM

In the case of Algorithm 11 we are going to use a rather different approach to prove convergence. Since the coordinate descent scheme assumes one agent update per iteration and the probability

of an update complies to some prior distribution, we will prove convergence of the iterates $\{x^k\}$ to some optimizer x^* *in expectation*. To this end, we make the assumption that h is *strongly convex* with strong convexity constant μ. Moreover, we assume that every $K > 0$ iterations a full correction takes place, namely all the agents communicate their actual optimizers to the coordinator, and that K is bounded.

Theorem 10. *Let N agents update with probabilities p_i and $p_{\min} = \min_{i\in\{1,\dots,N\}} p_i$. Let $\nu = 1 - \sqrt{(1 - 2\gamma\mu + \mu\gamma^2 L)}$, where L is the Lipschitz constant of ∇h and μ the strong convexity constant of h, while $\gamma < 2/L$. If x^* is the unique optimizer of Problem (5.12), for any time instant $k > K$, the sequence $\{x^k\}$ generated by Algorithm 11 satisfies*

$$\begin{aligned}\mathrm{E}[\|x^{k+K} - x^*\|^2] \leq \left(1 - \frac{\rho(\mu-\epsilon)}{N}\right)^k \mathrm{E}[\|x^K - x^*\|^2] \\ + \frac{\rho}{N}\left(\frac{1}{\epsilon} + \frac{\rho(1+\delta)}{Np_{\min}\delta}\right)\sum_{j=1}^{k}\left(1 - \frac{\rho(\mu-\epsilon)}{N}\right)^{k-j}\mathrm{E}[\|e^{K-1+j}\|^2] \ ,\end{aligned} \tag{5.34}$$

for $\rho \in (0, Np_{\min}/(2(1+\delta)))$, $\eta^k = \frac{\rho}{Np_{i_k}}$, $\delta > 0$, $\nu > \epsilon > 0$, and $e^k = (e_1^k, \dots, e_N^k) \in \mathbb{R}^{Nn}$ the vector that is constituted of the components $e_i^k = \gamma(\nabla f_i^\gamma(v^k) - g_i^k)$, $i = 1, \dots, N$, while $e^K = 0$.

Proof: The proof is deferred to Appendix 5.9.6. □

Theorem 10 states that if the algorithm is terminated earlier, the sequence $\{x^k\}$ will converge (in expectation) to a ball that is centered at the optimum and has a radius that increases with the expected error that has been accumulated since the last correction occured.

Remark 14. *The absence of any assumptions on h results in a diminishing stepsize rule in the case of ADMM (Theorem 9), in contrast to the randomized coordinate descent PGM where η^k need not be vanishing. On the other hand, only convergence within a ball centered at the optimum can be shown in the latter case, while the diminishing sequence $\{\eta^k\}$ will result in asymptotic convergence.*

5.7 Applications

5.7.1 Distributed Load Sharing

The sharing problem takes the form

$$\text{minimize} \quad \sum_{t=0}^{T-1}(\sum_{i=1}^{N} p_i^{\mathrm{cb}}(t) - r(t))^2 + \sum_{i=1}^{N} f_i(p_i^{\mathrm{cb}}, u_i, x_i) \ , \tag{5.35}$$

which is identical to the one given in (2.23), with variables $p_i^{\mathrm{cb}}, u_i, x_i$. The variable $p_i^{\mathrm{cb}}(t)$ refers to the total consumption of the i^{th} building at time instant t, where $i = 1, \dots, N$, and $p_i^{\mathrm{cb}} = (p_i^{\mathrm{cb}}(0), \dots, p_i^{\mathrm{cb}}(T-1)) \in \mathbb{R}^T$, $p^{\mathrm{cb}} = (p_1^{\mathrm{cb}}, \dots, p_N^{\mathrm{cb}}) \in \mathbb{R}^{NT}$. The reference power profile is denoted by $r(t)$, while time spans from $t = 0, \dots, T-1$, *i.e.,* we have a T-timesteps ahead prediction of the

power profile. The first term, $h(p^{\text{cb}}) = \sum_{t=0}^{T-1}(\sum_{i=1}^{N} p_i^{\text{cb}}(t) - r(t))^2$, penalizes the deviation of the total power contribution to the reference power profile. The individual components $f_i(p_i^{\text{cb}}, u_i, x_i)$ are local performance criteria coupled with implicit descriptions of convex sets, constructed by the intersection of linear equations (agent dynamics) and constraints, details that are hidden from the global node. More details are given below.

Modeling of the agents

The microgrid comprises small and medium office buildings, generated by the OpenBuild software [GQJ15]. An instance of a local optimization problem for building i is

$$\begin{aligned} f_i(p_i^{\text{cb}}, u_i, x_i) = w \sum_{t=0}^{T-1} \max \Big\{ & c_i(t)(p_i^{\text{cb}}(t) - p_i^{\text{base}}(t)), \\ & 0.5c_i(t)(p_i^{\text{cb}}(t) - p_i^{\text{base}}(t)), \\ & -0.5c_i(t)(p_i^{\text{cb}}(t) + 3p_i^{\text{base}}(t)) \Big\} \\ & + \delta((p_i^{\text{cb}}, u_i, x_i) \mid \mathcal{C}_{\text{build}}) \ , \end{aligned} \tag{5.36}$$

$$\mathcal{C}_{\text{build}} = \left\{ \begin{array}{l} x_i(t+1) = A_i x_i(t) + B_i u_i(t) \\ x_i(0) = x_i^{\text{init}} \\ p_i^{\text{cb}}(t) = \sum_{j=1}^{M_i} u_{ij}(t) \\ C_i x_i(t) \in X_i(t) \\ \|u_i(t)\|_\infty \leq u_i^{\max} \end{array} \right\} \ .$$

Note that this is identical to the description in (2.25), where in addition to feasibility, the buildings have certain preferences when it comes to tracking a given power profile. The local objective is weighted versus the global tracking cost by means of a constant $w > 0$. When the profile causes the total power consumption to exceed some threshold, the cost of electricity increases, while when the total power consumption drops below some threshold the ambient temperature might become discomforting (although tolerable) for the residents. This discontent is expressed in (5.36) by the piecewise affine function expressed by the max term in the objective. The data $c_i(t)$ is the associated electricity cost for bulding i at time t and p^{base} is the expected baseline consumption, the deviation from which is penalized or subsidized.

Instead of carrying the full building description as in $\mathcal{C}_{\text{build}}$, which would result in a high-dimensional optimization problem, the authors in [GBSJ15] propose a low-dimensional modeling abstraction of the building as a 'thermal battery'. A robust optimization problem is solved to ensure that the trackable power profiles $p_i^{\text{cb}}(t) = \sum_{j=1}^{M_i} u_{ij}(t) \in \mathbb{R}^T$ for building i reside inside a convex set, namely $\mathcal{P}_i^{\text{cb}}$, and satisfy the constraints imposed in (5.36).

Using the aforementioned abstraction, the optimization problem associated to building i be-

Simulation characteristics			
Data	1st January 2013		
Location	Lausanne		
Time	00:00 - 23:00		
Sampling time	60		min
Horizon	24		–
Dimension $\mathcal{P}_i^{\text{cb}}$	96		–
Buildings			
Desired temperature	20		°C
Minimum temperature (day/night)	18/15		°C
Maximum temperature (day/night)	22/25		°C
Tariff (day/night)	21.6/12.7	13.15/8.3	ct./kWh
Heat pump COP_{hot}	3.0		–
Heat pump COP_{cold}	3.0		–
	Small	**Medium**	
Number of systems	28	22	#
Area	511	4982	m^2
Number of states	15	54	–
Number of inputs	5	18	–
Average thermal consumption	4	40	W/m^2

Table 5.1: Micro-grid case study overview

comes

$$f_i(p_i^{\text{cb}}) = w \sum_{t=0}^{T-1} \max \Big\{ c_i(t)(p_i^{\text{cb}}(t) - p_i^{\text{base}}(t)), \\ 0.5c_i(t)(p_i^{\text{cb}}(t) - p_i^{\text{base}}(t)), \\ -0.5c_i(t)(p_i^{\text{cb}}(t) + 3p_i^{\text{base}}(t)) \Big\} \\ + \delta(p_i^{\text{cb}} \mid \mathcal{P}_i^{\text{cb}}) \ , \tag{5.37}$$

We can now solve (5.35) with f_i given by (5.37) using Algorithm 10.

Simulation setup

Our purpose is to assign one-hour reference tracking to a population that consists of agents described in the previous section. We consider $N = 50$ buildings, different electricity tariffs that vary according to the time of the day as well as the size of the load. The tracking term w in (5.37) is set to 0.05 in order to prioritize tracking over the agents' local objectives. Other details associated to the simulation are given in Table 5.1.

We solve problem (5.37) using ADMM (5.19) with $\rho = 20$. A comparison is performed between the exact solution of the problem instance, and three schemes that allow for reduced communication. We consider:

1. The proposed reduced communication approach presented in Algorithm 10.
2. Directly using the tightest hyperplane as given in (5.22) (without evaluating the communication test (5.21)).
3. Using the previous guess for the pair (y_i, λ_i) whenever there is no communication, *i.e.,* $y_i^{k+1} = y_i^k$ and $\lambda_i^{k+1} = \lambda_i^k$.

In addition, we trigger a compulsory communication round to and from all the agents every 10 iterations, *i.e.,* whenever $\mod(k, 10) = 0$. The proximal minimization problems are solved in MATLAB using the YALMIP optimizer [LÖ4] with the Gurobi solver. The termination criterion is a combination of feasibility and optimality, namely we iterate until inclusion of p_i^{cb} in $\mathcal{P}_i^{\text{cb}}$ is satisfied for all agents with accuracy 10^{-4}, while the distance between two consequtive iterates satisfies $\|(p^{\text{cb}})^k - (p^{\text{cb}})^{k-1}\|/\|(p^{\text{cb}})^{k-1}\| \leq 10^{-4}$.

Figure 5.3 depicts the number of iterations versus the number of communication rounds using the four approaches discussed above. We observe from Figure 5.3a that Algorithm 10 achieves approximately 32% reduction in communication in comparison to the regular ADMM implementation. The number of iterations is, as expected, increased (from 19 to 52). When the tightest hyperplane identified by (5.22) is used, the algorithm still converges, the communication rounds increase by 16% and the number of iterations jumps up to 72. Figure 5.3b indicates that when the agents' updated values are set to the previous ones, the algorithm does not converge.

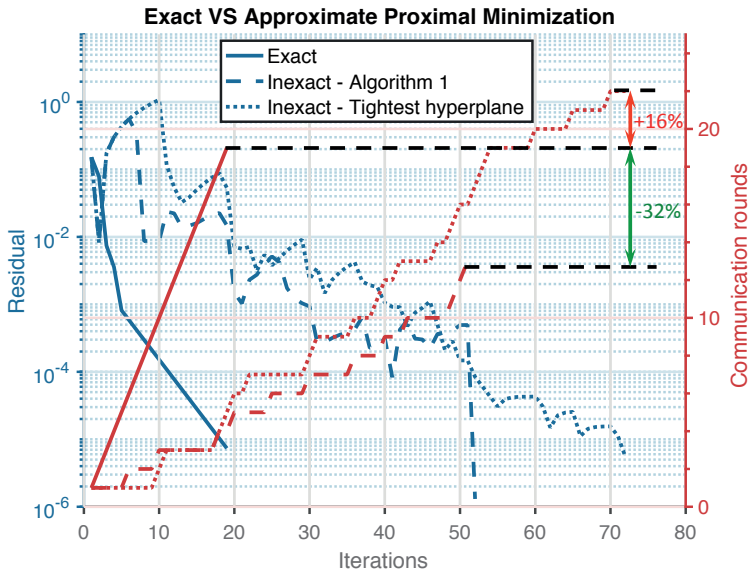
The tracking quality of the mix is depicted in Figure 5.4. The small buildings' constribution is colored in pink, while that of the medium scale buildings in blue. The reference signal in red has been normalized with respect to the baseline. When the buildings are asked to decrease their consumption (negative contribution), participation is more vigorous, while in the opposite case (increase consumption, positive contribution) the asymmetry caused by the objective (5.36) renders the mix 'lazier' to react.

To sum up, it is observed that when the proximal solution is estimated, the communication rounds are significantly reduced, by more than 30%. This reduction is accompanied by a significant increase in the number of iterations needed for convergence. The cause for this increase can be attributed to the excursions that the trajectory takes, most probably caused by the erroneous optimizers that are selected in the early iterations. We see that confining the potential optimizers within the set has a beneficial effect in comparison to just guessing the solution, although it comes at the cost of solving a QCQP of complexity that is growing with the number of generated query points.

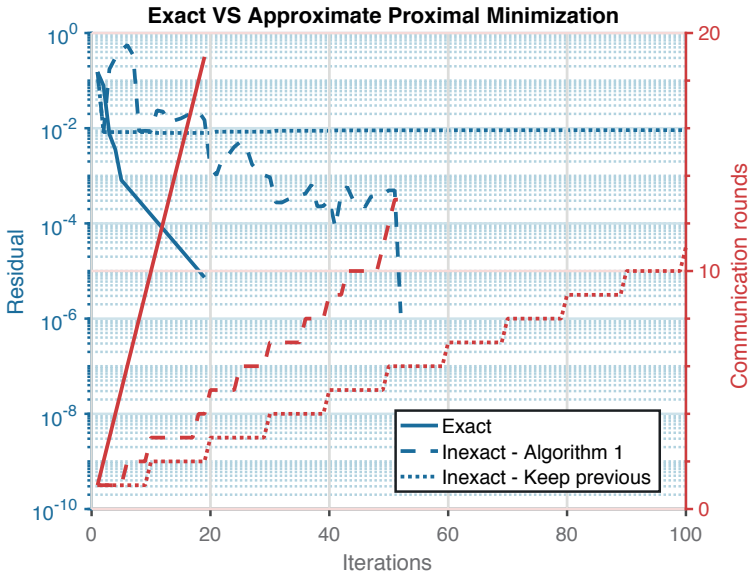
5.7.2 Optimal Price Adjustment

We consider a setup identical to the one presented above, with the difference being that we now solve the dispatchability problem (2.24). Problem (5.35) takes the form

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} f_i^{\text{cb}}(p_i^{\text{cb}}, u_i, x_i) + f^{\text{bess}}(p^{\text{bess}}) \\ \text{subject to} \quad & \sum_{t=1}^{T} p^{\text{bess}}(t) + \sum_{i=1}^{N} (p_i^{\text{cb}}(t) - \hat{p}_i^{\text{cb}}(t)) = r(t) \ , \end{aligned} \tag{5.38}$$



(a) Exact ADMM versus Algorithm 10 versus Tightest hyperplane approach.



(b) Exact ADMM versus Algorithm 10 versus Keeping-value-constant approach.

Figure 5.3: Performance in terms of communication savings.

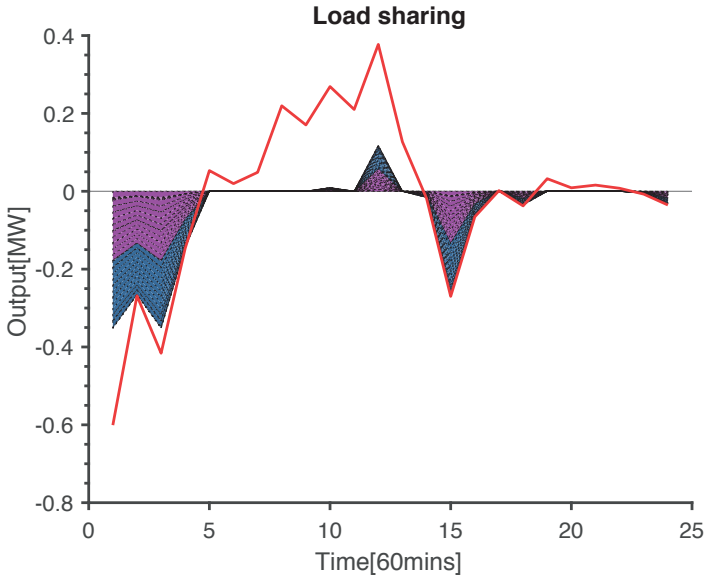


Figure 5.4: Tracking of a reference by a mix of small (pink) and medium scale (blue) buildings.

where

$$f^{\text{bess}}(p^{\text{bess}}) := \left\{ \frac{1}{2} \sum_{t=1}^{T} \|SOC(t) - SOC^{\text{ref}}(t)\|_2^2 \;\mid\; p^{\text{bess}} \in \mathcal{C}^{\text{bess}} \right\} , \tag{5.39}$$

$$\mathcal{C}^{\text{bess}} = \left\{ \begin{array}{l} SOC(t+1) = aSOC(t) + bp^{\text{bess}}(t) \\ SOC(0) = SOC^{\text{init}} \\ SOC_{\min} \leq SOC(t) \leq SOC_{\max} \\ p^{\text{bess}}_{\min} \leq p^{\text{bess}}(t) \leq p^{\text{bess}}_{\max} \end{array} \right\} ,$$

while f_i^{cb} is given by (5.37).

After performing the simple reformulation suggested in [BPCPE11, Section 7.3.2], the dispatchability problem (5.38) reduces to the *optimal exchange* problem given below.

$$\begin{array}{ll} z_i^{k+1} & = \arg\min_{z_i} \left\{ f_i(z_i) - \langle z_i, \lambda^k \rangle + (\rho/2)\|z_i - (z_i^k - \bar{z}^k)\|_2^2 \right\}, \quad i = 1, \ldots, N+1 \\ \lambda^{k+1} & = \lambda^k + \rho(\bar{z}^{k+1} - r/N) \ , \end{array} \tag{5.40}$$

where we have denoted $z_i = p_i^{\text{cb}}$, $i = 1, \ldots, N$, $z_{N+1} = p^{\text{bess}}$, $f_i = f_i^{\text{cb}}$, $i = 1, \ldots, N$, $f_{N+1} = f^{\text{bess}}$ while $\bar{z} = \frac{1}{N+1} \sum_{i=1}^{N+1} z_i$.

Except for the fact that (5.40) gives a simple way to solve (5.38), there is another interesting observation to make. Examined through the lens of *Walras' (simultaneous) tâtonnement* [Wal96] or *price adjustment* process, we can give the following intrepretation to (5.40) [Uza60]:

1. At each iteration, the coordinator (Secretary of Market) announces a price vector λ.

2. Each agents submits to the coordinator a vector z_i with its quantities of demand and supply

of electrical power based on the announced price vector.

3. The coordinator adapts the price vector to the received supply/demand profiles and the process repeats until the total supply equals the reference.

If the quadratic term that appears in the proximal minimization agent subproblems is removed, we end up with *dual decomposition*, the most basic algorithmic form of tâtonnement. As mentioned in [BPCPE11, Section 7.3.2], the term added by ADMM indicates each agent's commitment to help clear the market.

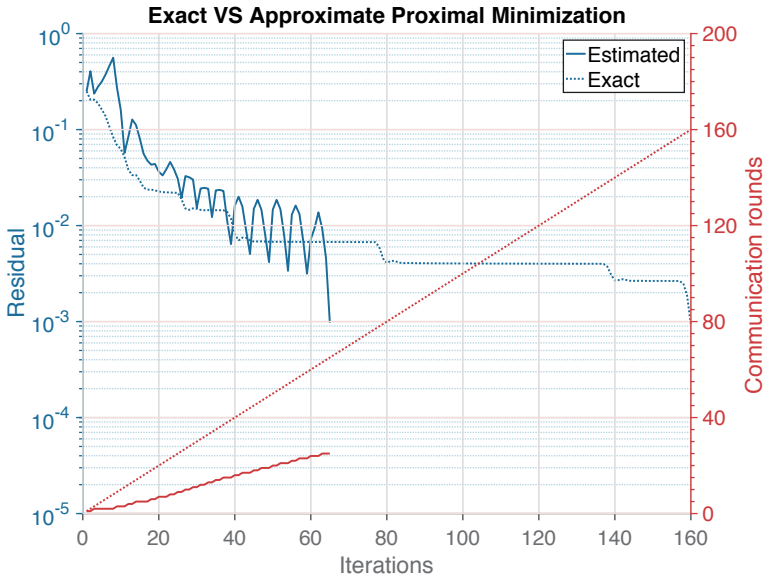
By 'learning' the envelope function of each agent, the coordinator is capable of bounding the predicted individual demands inside some set for a given price vector. With this in mind, one can think of Algorithm 10 applied to (5.40) as a *mechanism design* protocol, where the goal is to elicit private information from each of multiple agents in order to select a desirable system wide outcome.

In addition, the process always operates as a communication reduction scheme, the performance of which is illustrated in Figure 5.5. The residual is formed as the relative distance of the current power consumption vector to the optimizer. When terminated early (Figure 5.5a), the scheme reduces considerably the number of both iterations for convergence and communication rounds. Since this measurement can be considered unreliable due to the oscillatory nature of the residual trajectory, in Figure 5.5b we illustrate a running average of the last 100 iterations of the residual. We observe a (much) larger number of iterations to reach the same accuracy, followed by a modest (27%) decrease in communication.

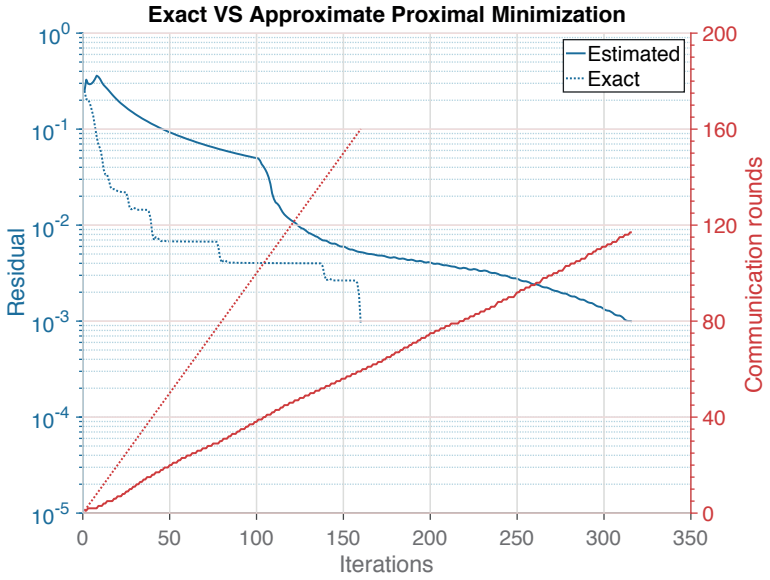
5.7.3 Randomized Coordinate Load Sharing

We finally consider once more the distributed load sharing problem in the beginning of the section, this time, though, the agents do not pursue any individual objectives, namely only the indicator function of the composite cost (5.37) is considered. We solve problem (5.35) using the projected gradient method with randomized coordinate descent. Uniform updates are considered, *i.e.,* $p_i = 1/N$ while we set $\eta^k = \eta = 0.9\ \forall k$ (its value should be smaller according to Theorem 10, but the formula is rather conservative, while the aforementioned choice performed well in practice). A comparison is performed between the exact solution of the problem and Algorithm 11. The latter is implemented in two different ways: We first consider the worst-case approach (5.14) for locating each agent's optimizer. Afterwards, we compare to the centering approach (5.24). The proximal minimization problems and the QCQPs of the worst-case scenario are solved in MATLAB using the YALMIP optimizer [Löf04] with the Gurobi solver, while the SDPs are solved using CVX [CR12] with SEDUMI [Stu98].

Figure 5.6 depicts the number of iterations versus the number of communication rounds in all three approaches, namely exact, inexact worst-case and inexact with centering. The termination criterion is a combination of feasibility and optimality, namely we iterate until inclusion of p_i^{cb} in $\mathcal{P}_i^{\text{cb}}$ is satisfied for all agents with accuracy 10^{-3}, while the distance to the optimizer $\|(p^{\text{cb}})^k - (p^{\text{cb}})^*\|/\|(p^{\text{cb}})^*\| \leq 10^{-1}$. It is observed that in both cases where the proximal solution is estimated, the communication rounds are significantly reduced, *i.e.,* by more than 70% in the worst-case approach and by more than 30% in the centering approach. In addition, with the centering approach the number of iterations for convergence is also reduced by about 25%. Surprising though it might

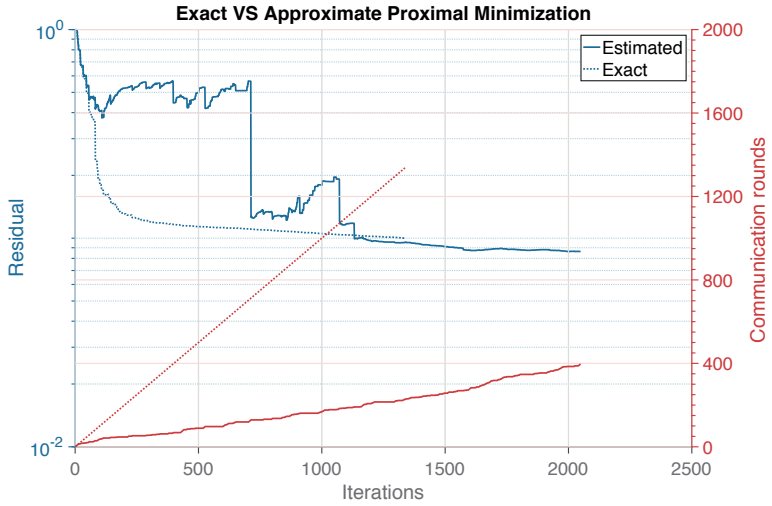


(a) Exact ADMM versus Algorithm 10: Convergence.

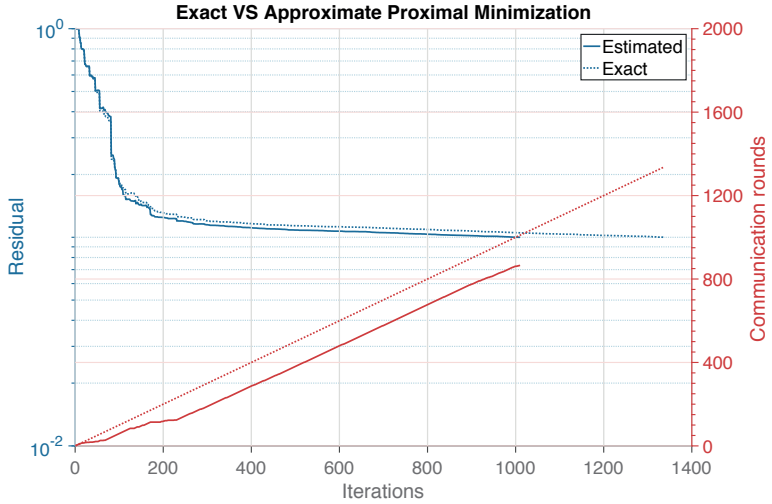


(b) Exact ADMM versus Algorithm 10: Averaged residual convergence.

Figure 5.5: Performance in terms of communication savings for the price adjustment (dispatch) problem.



(a) Exact ADMM versus Algorithm 11: Worst-case gradient estimate (5.14).



(b) Exact ADMM versus Algorithm 11: Gradient estimate is the Chebychev center (5.24).

Figure 5.6: Performance in terms of communication savings for the randomized coordinate descent implementation of the load sharing problem.

seem, this is possible since the estimated gradient g_i^k might result in an infeasible approximate optimizer $(p_i^{\text{cb}})^{k+1}$ for some i, such that $(p_i^{\text{cb}})^{k+1} \notin \mathcal{P}_i^{\text{cb}}$, thus giving rise to a more aggressive step.

In conclusion, we can say that estimating the solution to the proximal minimization problem seems to be more beneficial in the case where coordinate descent is applied instead of the regular algorithmic iteration. Our explanation for this observation has as follows. Since in the coordinate descent case there is only one agent update per iteration, there is no significant change between two consecutive optimizers. This, in turn, renders the non-communication test easier to pass, resulting in fewer communication rounds. Moreover, the typically large number of iterations necessary for convergence gives the coordinator enough time to 'learn' the agents.

5.8 Conclusion

Modern multi-agent setups consist of several heterogeneous components equipped with prediction and control algorithms, that attribute to them some decision-making capacity. These attributes ask for distributed solutions which come with an inherent communication overhead. We propose a framework where the communication requests are reduced by enabling the central coordinator to gradually 'learn' the optimization model of the agents and to trigger them based on the result of a predesigned certification test.

A more interesting goal would be to generalize the scheme to decentralized settings, for instance, to wireless sensor networks where only one-hop communication is allowable. Such networks are characterized by high energy consumption related to communication, often several orders of magnitude greater than the energy required for local computations [RN04]. As a result, our proposed approach could become of practical relevance.

5.9 Appendices

5.9.1 Proof of Proposition 2

Without loss of generality we derive the analysis for the query point z_1 to be consistent with Figure 5.1.

1. We need to show that $\nabla f^\gamma(v) \in \partial_{\epsilon^*(v;z_1)} \overline{f}^\gamma(v; z_1)$. Employing Definition 6, we start by showing that
$$\overline{f}^\gamma(z; z_1) \geq \overline{f}^\gamma(v; z_1) + \langle \nabla f^\gamma(v), z - v \rangle - \epsilon^*(v; z_1),\ \forall z \in \mathbb{R}^n \ .$$
We will proceed and prove the statement by contradiction. Suppose that $\exists\ z \in \mathbb{R}^n$, such that
$$\overline{f}^\gamma(z; z_1) < \overline{f}^\gamma(v; z_1) + \langle \nabla f^\gamma(v), z - v \rangle - \epsilon^*(v; z_1)$$
holds. From the definition of $\epsilon^*(v; z_1)$ (equation (5.4)), the above inequality is equivalent to:
$$\overline{f}^\gamma(z; z_1) < \langle \nabla f^\gamma(v), z - v \rangle + f^\gamma(v) \ .$$
From convexity of f^γ, it holds that $f^\gamma(v) + \langle \nabla f^\gamma(v), z - v \rangle \leq f^\gamma(z),\ \forall z \in \mathbb{R}^n$, therefore we

conclude that $\exists\, z \in \mathbb{R}^n$ such that

$$\overline{f}^{\gamma}(z; z_1) < f^{\gamma}(z) \ ,$$

which leads to a contradiction. Consequently, $\nabla f^{\gamma}(v) \in \partial_{\epsilon^*(v;z_1)} \overline{f}^{\gamma}(v; z_1)$.

2. The property follows directly from the fact that $\overline{f}^{\gamma}(z; z_1)$ is real-valued, along with [Ber15, Proposition 6.7.1].

5.9.2 Proof of Proposition 5

All we need in order to prove the statement is a function f for which $\epsilon^*(v; z_1)$ results in a tight bound. Let us assume that $f(x)$ is an indicator function of some convex set $\mathcal{X}$ and thus given by $f(x) = \delta(x \mid \mathcal{X})$. Let us also assume that the set (5.7) associated to $\epsilon^*(v; z_1)$ is not the smallest, *i.e.,* $\exists\, \epsilon > 0$ such that $\epsilon = \overline{f}^{\gamma}(v; z_1) - f^{\gamma}(v) - \alpha$ for some $\alpha > 0$, and that $\nabla f^{\gamma}(v) \in \partial_{\epsilon} \overline{f}^{\gamma}(v; z_1)$, or

$$\overline{f}^{\gamma}(z; z_1) \geq \overline{f}^{\gamma}(v; z_1) + \langle \nabla f^{\gamma}(v), z - v \rangle - \epsilon, \ \forall z \in \mathbb{R}^n \ .$$

Substituting ϵ from above, we have that

$$\overline{f}^{\gamma}(z; z_1) \geq f^{\gamma}(v) + \langle \nabla f^{\gamma}(v), z - v \rangle + \alpha, \ \forall z \in \mathbb{R}^n \ . \tag{5.41}$$

For $z = z_1$, equation (5.41) reads

$$f^{\gamma}(z_1) \geq f^{\gamma}(v) + \langle \nabla f^{\gamma}(v), z_1 - v \rangle + \alpha \ .$$

If $z_1 \in \mathcal{X}$, *i.e.,* $z_1 \in \mathrm{dom}(f)$, it holds $\forall v \in \mathcal{X}$ that $f^{\gamma}(z_1) = f^{\gamma}(v) = 0$ and $\nabla f^{\gamma}(v) = 0$. Consequently, the previous inequality results in $\alpha \leq 0$, which leads to a contradiction. Therefore, there exists no set $\partial_{\epsilon} \overline{f}^{\gamma}(v; z_1)$ smaller than $\partial_{\epsilon^*} \overline{f}^{\gamma}(v; z_1)$ that contains $\nabla f^{\gamma}(v)$.

5.9.3 Proof of Theorem 8

We need to compute explicitly the ϵ-subdifferential sets of both $\overline{f}^{\gamma}(v)$ and $\overline{f}^{\gamma}(v; z_1)$ and subsequently perform the comparison. We will need the following result from [Roc70]:

Lemma 18. *[Roc70, Theorem 16.5] The conjugate of $\overline{f}^{\gamma}(z \mid \mathcal{J})$ is given by*

$$(\overline{f}^{\gamma})^{\star}(g \mid \mathcal{J}) = (\mathrm{conv}\{\overline{f}^{\gamma}(z; z_j) \mid j \in \mathcal{J}\})^{\star} = \sup_{j \in \mathcal{J}} \{(\overline{f}^{\gamma})^{\star}(g; z_j)\}.$$

- Let us fix the distance of $f^{\gamma}(v)$ from the quadratic upper bounds as demonstrated in Figure 5.2. The $\epsilon^*(v; z_j)$-subdifferential sets for $\overline{f}^{\gamma}(v; z_j), \ j \in \mathcal{J}$ are:

$$\begin{aligned} \partial_{\epsilon^*(v;z_j)} \overline{f}^{\gamma}(v; z_j) =& \Big\{ g \mid (\overline{f}_v^{\gamma})^{\star}(g; z_j) \leq \epsilon^*(v; z_j) \Big\} \\ =& \Big\{ g \mid (\overline{f}^{\gamma})^{\star}(g; z_j) + \overline{f}^{\gamma}(v; z_j) - \langle v, g \rangle \leq \epsilon^*(v; z_j) \Big\} \ , \end{aligned} \tag{5.42}$$

where we used relation (5.6) in the first equality and (5.5) in the second.

- For $\overline{f}^{\gamma}(v)$:

$$\begin{aligned}\partial_{\epsilon^*(v|\mathcal{J})}\overline{f}^{\gamma}(v \mid \mathcal{J}) &= \left\{g \mid (\overline{f}_v^{\gamma})^{\star}(g \mid \mathcal{J}) \leq \epsilon^*(v \mid \mathcal{J})\right\} \\ &= \left\{g \mid (\overline{f}^{\gamma})^{\star}(g \mid \mathcal{J}) + \overline{f}^{\gamma}(v \mid \mathcal{J}) - \langle v, g\rangle \leq \epsilon^*(v \mid \mathcal{J})\right\} \\ &= \left\{g \mid (\sup_{j\in\mathcal{J}}\{(\overline{f}^{\gamma})^{\star}(g; z_j)\} + \overline{f}^{\gamma}(v \mid \mathcal{J}) - \langle v, g\rangle \leq \epsilon^*(v \mid \mathcal{J})\right\} ,\end{aligned} \tag{5.43}$$

  where we used relation (5.6) in the first equality, (5.5) in the second, and Lemma 18 in the third.

Equation (5.43) holds for each $(\overline{f}^{\gamma})^{\star}(g; z_j)$ individually since it holds for the pointwise supremum. In addition, it is evident from Figure 5.2 that $\epsilon^*(v \mid \mathcal{J})$, expressing the distance of $\overline{f}^{\gamma}(v \mid \mathcal{J})$ from the Moreau envelope $f^{\gamma}(v)$, can be expressed with respect to any of the distances $\epsilon^*(v; z_j)$, reduced by a factor of $\overline{f}^{\gamma}(v; z_j) - \overline{f}^{\gamma}(v \mid \mathcal{J})$. Equation (5.43) thus ends up reading

$$\partial_{\epsilon^*(v|\mathcal{J})}\overline{f}^{\gamma}(v \mid \mathcal{J}) = \left\{g \mid (\overline{f}^{\gamma})^{\star}(g; z_j) + \overline{f}^{\gamma}(v; z_j) - \langle v, g\rangle \leq \epsilon^*(v; z_j),\ \forall j \in \mathcal{J}\right\} ,$$

so $\partial_{\epsilon^*(v|\mathcal{J})}\overline{f}^{\gamma}(v \mid \mathcal{J}) = \bigcap_{j\in\mathcal{J}} \partial_{\epsilon^*(v;z_j)}\overline{f}^{\gamma}(v; z_j)$ from (5.42).

5.9.4 Proof of Proposition 6

Starting from the fact that $x^{k+1} = v^k - \gamma g^k$, we want to find from (5.16) the worst-case point g such that

$$h(v^k - \gamma g) + \sum_{i=1}^{N} f_i(v_i^k - \gamma g_i) \leq h(x^k) + \sum_{i=1}^{N} f_i(x_i^k) - \frac{1}{2\gamma}\|v^k - \gamma g - x^k\|^2 ,$$

or

$$\max_{g} \quad \left\{h(v^k - \gamma g) + \sum_{i=1}^{N} f_i(v_i^k - \gamma g_i) + \frac{1}{2\gamma}\|v^k - \gamma g - x^k\|^2\right\} \leq h(x^k) + \sum_{i=1}^{N} f_i(x_i^k) .$$

It holds that

$$\begin{aligned}&h(v^k - \gamma g) + \sum_{i=1}^{N} f_i(v_i^k - \gamma g_i) + \frac{1}{2\gamma}\|v^k - \gamma g - x^k\|^2 = \\ &h(v^k - \gamma g) + \sum_{i=1}^{N} f_i^{\gamma}(v_i^k) - \frac{1}{2\gamma}\|\gamma g\|^2 + \frac{1}{2\gamma}\|\gamma g - v^k + x^k\|^2 = \\ &h(v^k - \gamma g) + \sum_{i=1}^{N} f_i^{\gamma}(v_i^k) - \langle g, v^k - x^k\rangle + \frac{1}{2\gamma}\|v^k - x^k\|^2 \leq \\ &h(v^k - \gamma g) + \sum_{i=1}^{N} \min_{j\in\mathcal{J}}\{\overline{f}_i^{\gamma}(v_i^k; z_{i,j})\} - \langle g, v^k - x^k\rangle + \frac{1}{2\gamma}\|v^k - x^k\|^2 ,\end{aligned}$$

where the first equality follows from (2.7), the second from the development of the quadratic form and the inequality from the fact that $f_i^\gamma(v_i^k) \leq \overline{f}_i^\gamma(v_i^k; z_{i,j}),\ \forall j \in \mathcal{J}$.

5.9.5 Proof of Proposition 7

We start by analyzing the ADMM iterations in (5.19). The proximal and dual iterations can be, respectively, expressed as

$$\begin{aligned}
\hat{y}_i^{k+1} &= v_i^k - \gamma g_i^k \\
&= v_i^k - \gamma \nabla f_i^\gamma(v_i^k) - \gamma e_i^k \\
&= \underbrace{\mathbf{prox}_{\frac{1}{\rho} f_i}(v_i^k)}_{y_i^{k+1}} - (1/\rho) e_i^k \\
\hat{\lambda}_i^{k+1} &= \lambda_i^k + \rho(\tilde{x}_i^{k+1} - \hat{y}_i^{k+1}) \\
&= \underbrace{\lambda_i^k + \rho(\tilde{x}_i^{k+1} - y_i^{k+1})}_{\lambda_i^{k+1}} + e_i^k \ .
\end{aligned} \tag{5.44}$$

Let us introduce the variable $\mu_i^k := \lambda_i^k + \rho \tilde{x}_i^{k+1}$ and write the dual update as

$$\begin{aligned}
\lambda_i^{k+1} &= \mu_i^k - \rho\, \mathbf{prox}_{\frac{1}{\rho} f_i}(v_i^k) \\
&= \mu_i^k - \rho\, \mathbf{prox}_{\frac{1}{\rho} f_i}(\mu_i^k/\rho) \ ,
\end{aligned} \tag{5.45}$$

where the second equality follows from (5.20).

We still need to express the error in (5.44) with respect to the functions appearing in (5.29). The Moreau identity is a useful Lemma that associates a convex function with its conjugate and is given below.

Lemma 19. *Let $f : \mathbb{R}^n \mapsto \mathbb{R} \cup \{+\infty\}$ be a closed proper convex function. Then for any $x \in \mathbb{R}^n$*

$$\mathbf{prox}_{\rho f^*}(x) + \rho\, \mathbf{prox}_{f/\rho}(x/\rho) = x, \ \ \forall\, 0 < \rho < +\infty \ .$$

Using Lemma 19 in (5.45), it directly follows that that $\lambda_i^{k+1} = \mathbf{prox}_{\rho f_i^\star}(\lambda_i^k + \rho \tilde{x}_i^{k+1})$. Substituting in (5.44), we finally get that

$$\hat{\lambda}_i^{k+1} = \mathbf{prox}_{\rho f_i^\star}(\lambda_i^k + \rho \tilde{x}_i^{k+1}) + e_i^k \ . \tag{5.46}$$

It follows from the definition of $F(\lambda)$ and (5.46) that Algorithm 10 can be equivalently cast as the inexact DR iteration provided that $z^{k+1} = \lambda^k + \rho \tilde{x}^{k+1}$. The proof of this last point follows directly from the slides [Van10, Lecture 13].

By switching the second and third updates, the DR iteration (5.29) can be written as

$$\begin{aligned} r^{k+1} &= \mathbf{prox}_{\rho H}(2\lambda^k - z^k) \\ \lambda^{k+1} &= \mathbf{prox}_{\rho F}(z^k + \eta(r^{k+1} - \lambda^k)) \\ z^{k+1} &= z^k + \eta(r^{k+1} - \lambda^k) \ . \end{aligned} \tag{5.47}$$

By introducing the variable $w = z - \lambda$, the iteration above can be cast as

$$\begin{aligned} r^{k+1} &= \mathbf{prox}_{\rho H}(\lambda^k - w^k) \\ \lambda^{k+1} &= \mathbf{prox}_{\rho F}(\lambda^k + w^k + \eta(r^{k+1} - \lambda^k)) \\ w^{k+1} &= w^k + \lambda^k + \eta(r^{k+1} - \lambda^k) \ . \end{aligned}$$

It is shown in the slides that the x-update of the ADMM algorithm (5.19) can be expressed as $x^{k+1} = \frac{1}{\rho}(r^{k+1} + w^k - \lambda^k)$. Similarly, the y-update of the algorithm can be written as $y^{k+1} = \frac{1}{\rho}w^k$. Putting these two together, we have that

$$\begin{aligned} \tilde{x}^{k+1} &= \eta x^{k+1} + (1-\eta)y^k \\ &= \eta\frac{1}{\rho}(r^{k+1} + w^k - \lambda^k) + (1-\eta)\frac{1}{\rho}w^k \ . \end{aligned}$$

Then $\lambda^k + \rho\tilde{x}^{k+1} = \lambda^k + w^k + \eta(r^{k+1} - \lambda^k)$, which is the z-update given in (5.47) after substituting $z = w + \lambda$.

5.9.6 Proof of Theorem 10

The key point is to observe that the approximate iteration $x_i^{k+1} = v_i^k - \gamma g_i^k$ can be expressed as an inexact proximal gradient iteration. To this end, we introduce the error sequence $\{e^k\}$ so as to write

$$e_i^k + \mathbf{prox}_{\gamma f_i}(v_i^k) = v_i^k - \gamma g_i^k \ , \tag{5.48}$$

while

$$\mathbf{prox}_{\gamma f_i}(v_i^k) = v_i^k - \gamma\nabla f_i^\gamma(v_i^k) \ . \tag{5.49}$$

Substituting (5.49) in (5.48) we have that

$$e_i^k = \gamma(\nabla f_i^\gamma(v_{i,k}) - g_i^k) \ . \tag{5.50}$$

Using the error (5.50), the randomized coordinate descent iteration can be expressed as

$$\begin{cases} x_{i_k}^{k+1} &= x_{i_k}^k + \eta^k\Big(e_{i_k}^k + \mathbf{prox}_{\gamma f_{i_k}}\left(x_{i_k}^k - \gamma\nabla_{i_k}h(x^k)\right) - x_{i_k}^k\Big) \\ x_{i\neq i_k}^{k+1} &= x_{i\neq i_k}^k \ , \end{cases}$$

or, more compactly, as

$$x^{k+1} = x^k + \eta^k U_{i_k}\Big(\mathbf{prox}_{\gamma F}\left(x^k - \gamma\nabla h(x^k)\right) - x^k + e^k\Big) \ . \tag{5.51}$$

The matrix $U_{i_k} : \mathbb{R}^{Nn} \mapsto \mathbb{R}^{Nn}$ is drawn from a set of orthogonal projection matrices $\{U_i\}_{i=1}^N$ such that $U_i : x \mapsto (0, \ldots, 0, x_i, 0, \ldots, 0),\ i = 1, \ldots, N$ and $\sum_{i=1}^N U_i = I$. Consequently, U_{i_k} isolates the i_k^{th} component of its argument, thus it updates the corresponding component of x, while the other components (agents) are set to their previous values. The proximal operator $\mathbf{prox}_{\gamma F}$ is defined as

$$\mathbf{prox}_{\gamma F}(x^{k+1}) := (\mathbf{prox}_{\gamma f_1}(x_1^{k+1}), \ldots, \mathbf{prox}_{\gamma f_N}(x_N^{k+1})).$$

Equation (5.51) is an instance of a more general *inexact fixed-point iteration*. Such iterations have been heavily studied in the literature. The works [LFP16; RFP13; SRB11] consider the case of deterministic fixed-point iterations with errors both in the gradient and in the proximal operator, where the errors are summable, while [PXYY16] analyze asynchronous fixed-point iterations, where the errors appear due to outdated samples in the update. It turns out that the asynchronous iteration in the latter works takes the form (5.51), the only difference being the expression for the error (5.50). We can thus employ similar arguments for proving convergence.

By introducing the operator

$$T : \mathbb{R}^n \mapsto \mathbb{R}^n, \quad T := \mathbf{prox}_{\gamma F}\,(I - \gamma \nabla h)$$

and

$$S : \mathbb{R}^n \mapsto \mathbb{R}^n, \quad S = I - T\ .$$

equation (5.51) can be written as

$$x^{k+1} = x^k + \eta^k U_{i_k}(Tx^k - x^k + e^k) = x^k - \eta^k U_{i_k} s^k\ , \tag{5.52}$$

where $s^k = Sx^k - e^k$, and e^k is given by (5.50). We set the relaxation parameter to $\eta^k = \frac{\rho}{N p_{i_k}}$, where $\rho > 0$ will be bounded from above later on.

Our purpose is to bound the distance of x^{k+1} to the fixed point x^* as a function of $\|x^k - x^*\|$ and $\|e^k\|$, always in expectation. We thus introduce $X^k = \left\{x^0, x^1, \ldots, x^k\right\}$, and by taking the conditional expectation and squaring (5.52), we get

$$\begin{aligned}
\mathrm{E}[\|x^{k+1} - x^*\|^2 \mid X^k] &= \|x^k - x^*\|^2 - 2\frac{\rho}{N}\,\mathrm{E}[\langle x^k - x^*, \frac{1}{p_{i_k}} U_{i_k} s^k\rangle \mid X^k] + \frac{\rho^2}{N^2}\,\mathrm{E}[\|\frac{1}{p_{i_k}} U_{i_k} s^k\|^2 \mid X^k] \\
&= \|x^k - x^*\|^2 - 2\frac{\rho}{N}\sum_{i=1}^N p_i \langle x^k - x^*, \frac{1}{p_i} U_i s^k\rangle + \frac{\rho^2}{N^2}\sum_{i=1}^N p_i \langle \frac{1}{p_i} U_i s^k, \frac{1}{p_i} U_i s^k\rangle \\
&\leq \|x^k - x^*\|^2 - 2\frac{\rho}{N}\langle x^k - x^*, s^k\rangle + \frac{\rho^2}{N^2 p_{\min}}\|s^k\|^2\ ,
\end{aligned} \tag{5.53}$$

where the second equality follows from the definition of the expectation and the third one from the fact that U_i is an orthogonal projection operator.

Let us now analyze the second and third term in (5.53).

- Bound $-2\frac{\rho}{N}\langle x^k - x^*, s^k\rangle$: From the definition of $s^k = Sx^k - e^k$, it holds that

$$\langle x^k - x^*, s^k\rangle = \langle x^k - x^*, Sx^k\rangle - \langle x^k - x^*, e^k\rangle\ . \tag{5.54}$$

We will now upper-bound the resulting inner product terms. In order to do so, we must use both the Lipschitz continuity of ∇h and the strong convexity of h.

Lemma 20. *Let $S = I - \mathbf{prox}_{\gamma F}(I - \gamma\nabla h)$ as defined above. Then*

$$\langle x^k - x^*, Sx^k\rangle \geq \frac{1}{2}\|Sx^k\|^2 \ .$$

Proof: If $T = \mathbf{prox}_{\gamma F}(I - \gamma\nabla h)$ is a nonexpansive operator, then the property holds for $S = I - T$ from [BC11, Proposition 4.33]. Nonexpansivity of T can be easily shown (see, *e.g.,* [PXYY16, Proposition 2.2]), from where the result follows. □

Lemma 21. *Denoting as L be the Lipschitz continuous gradient constant of h and μ its strong convexity modulus, it holds that*

$$\langle x^k - x^*, Sx^k\rangle \geq \nu\|x^k - x^*\|^2 \ ,$$

where $\nu = 1 - \sqrt{(1 - 2\gamma\mu + \mu\gamma^2 L)}$ for $\gamma < 2/L$.

Proof: From [BC11, Example 22.5] we have that if T is β-Lipschitz continuous for some $\beta \in [0, 1)$ then $I - T$ is $(1 - \beta)$-strongly monotone. It is proven in [PXYY16, Proposition 2.2] that $\|Tx - Tx^*\| \leq \sqrt{(1 - 2\gamma\mu + \mu\gamma^2 L)}\|x - x^*\|$ for $\gamma < 2/L$, so T is β-Lipschitz continuous with $\beta = \sqrt{(1 - 2\gamma\mu + \mu\gamma^2 L)}$, which concludes the proof. □

Using Lemmata 20 and 21 we get

$$-2\frac{\rho}{N}\langle x^k - x^*, Sx^k\rangle \leq -\frac{\rho\nu}{N}\|x^k - x^*\|^2 - \frac{\rho}{2N}\|Sx^k\|^2 \ . \tag{5.55}$$

For the second inner product term in (5.54) we can easily derive the bound

$$2\frac{\rho}{N}\langle x^k - x^*, e^k\rangle \leq 2\frac{\rho}{N}\|x^k - x^*\|\|e^k\| \ . \tag{5.56}$$

Equations (5.55) and (5.56) result in the bound

$$-2\frac{\rho}{N}\langle x^k - x^*, s^k\rangle \leq -\frac{\rho\nu}{N}\|x^k - x^*\|^2 - \frac{\rho}{2N}\|Sx^k\|^2 + 2\frac{\rho}{N}\|x^k - x^*\|\|e^k\| \ . \tag{5.57}$$

- Bound $\frac{\rho^2}{N^2 p_{\min}}\|s^k\|^2$: Using again the definition of s^k, we have that

$$\begin{aligned}\|s^k\|^2 &= \|Sx^k\|^2 + \|e^k\|^2 - 2\langle Sx^k, e^k\rangle \\ &\leq \|Sx^k\|^2 + \|e^k\|^2 + \frac{\delta}{p_{\min}}\|Sx^k\|^2 + \frac{1}{\delta p_{\min}}\|e^k\|^2 \ ,\end{aligned} \tag{5.58}$$

where the inner product term was bounded by employing Young's inequality.[1] We finally get the bound:

$$\frac{\rho^2}{N^2 p_{\min}}\|s^k\|^2 \leq \frac{\rho^2}{N^2 p_{\min}}(1+\delta)\|Sx^k\|^2 + \frac{\rho^2}{N^2 p_{\min}\delta}(1+\delta)\|e^k\|^2 \ . \tag{5.59}$$

Using (5.57) and (5.59), inequality (5.53) can be written as

$$\begin{aligned}\mathrm{E}[\|x^{k+1}-x^*\|^2 \mid X^k] \leq \|x^k - x^*\|^2 &- \frac{\rho\nu}{N}\|x^k - x^*\|^2 \\ &+ \frac{\rho}{N}\left(\frac{\rho(1+\delta)}{Np_{\min}} - \frac{1}{2}\right)\|Sx^k\|^2 \\ &+ 2\frac{\rho}{N}\|x^k - x^*\|\|e^k\| + \frac{\rho^2}{N^2 p_{\min}\delta}(1+\delta)\|e^k\|^2 \ . \end{aligned} \tag{5.60}$$

The third term in the sum can be eliminated by asumming that

$$\frac{\rho(1+\delta)}{Np_{\min}} - \frac{1}{2} < 0 \Rightarrow \rho < \frac{Np_{\min}}{2(1+\delta)} \ , \tag{5.61}$$

which gives rise to the inequality

$$\mathrm{E}[\|x^{k+1}-x^*\|^2 \mid X^k] \leq \|x^k - x^*\|^2 - \frac{\rho\nu}{N}\|x^k - x^*\|^2 + 2\frac{\rho}{N}\|x^k - x^*\|\|e^k\| + \frac{\rho^2}{N^2 p_{\min}\delta}(1+\delta)\|e^k\|^2 \ . \tag{5.62}$$

The complicating term on the right hand side can be eliminated by using once more Young's inequality, *i.e.,*

$$\begin{aligned} 2\frac{\rho}{N}\|x^k - x^*\|\|e^k\| &\leq 2\frac{\rho}{N}\left(\frac{\epsilon}{2}\|x^k - x^*\|^2 + \frac{1}{2\epsilon}\|e^k\|^2\right) \\ &= \frac{\rho\epsilon}{N}\|x^k - x^*\|^2 + \frac{\rho}{N\epsilon}\|e^k\|^2 \ . \end{aligned}$$

Using the above in (5.62) and taking the expectation in both sides, we recover the inequality

$$\mathrm{E}[\|x^{k+1}-x^*\|^2] \leq \left(1 - \frac{\rho(\nu-\epsilon)}{N}\right)\mathrm{E}[\|x^k - x^*\|^2] + \frac{\rho}{N}\left(\frac{1}{\epsilon} + \frac{\rho(1+\delta)}{Np_{\min}\delta}\right)\mathrm{E}[\|e^k\|^2] \ ,$$

for $\rho \in (0, Np_{\min}/(2(1+\delta)))$ and any $\delta > 0, \epsilon > 0$, which concludes the proof.

[1] For two nonnegative real numbers x and y, it holds that $xy \leq \frac{\delta x^2}{2} + \frac{y^2}{2\delta}$ for every $\delta > 0$.

Chapter 6

Extensions and Conclusions

The goal of this dissertation was to design decomposition methods that offer novel solutions to model-based constrained optimal control problems, both in the centralized and the distributed case. We studied the strengths of these methods in three distinct directions. First, we showed how proper splitting can efficiently solve a problem that was otherwise not solvable. Then we demonstrated how splitting can positively affect convergence speed in distributed optimization. Finally, we presented splitting methods as design mechanisms for protocols that reduce the need for communication in distributed settings. We offer below some more detailed concluding remarks for each chapter of this thesis, and an extended discussion on potential future directions.

6.1 Infinite horizon control

In Chapter 3 we proposed a way to solve the infinite-horizon constrained linear quadratic regulator problem, a problem that has, among others, inspired the development of the concept of receding horizon and, consequently, of linear model predictive control. The advantages of solving the CLQR over applying MPC are two: optimality and enlargement of the feasible region. Optimality is not that crucial since, in real setups, a plethora of factors might contribute to render the solution suboptimal, several of which are not related to the length of the horizon (*e.g.,* model mismatch). The size of the region of attraction is more significant; it can, however, be addressed by empirically choosing a large horizon. Taking these points into account, we are of the opinion that CLQR should be preferred over MPC only if the two schemes are comparable in terms of their computational overhead, that is to say, only when solving the CLQR is not (much) more expensive than applying MPC.

The computational complexity of the scheme is, of course, dependent on the type of optimal control problem that we want to solve. First and foremost, very few problems have a closed-form solution when taking an infinite horizon perspective, a fact that limits the applicability of CLQR to quadratic or discounted linear objectives. Focusing on control problems (infinite-horizon inventory models have also appeared in the literature [FZ84]), we discuss below a few possible directions.

1. **Tracking.** The setpoint tracking case should be a direct extension of the regulation problem, by considering, *e.g.,* the Δ-formulation of the former and by casting the problem as a regulation one. Pitfalls might, however, appear, when *e.g.,* the reference setpoint is on the boundary

of the constraint set, an event that would lead to the selection of very long horizons. The per-step computational complexity should remain the same as in the regulation case.

2. **Soft-constrained.** This problem arises when the states are given some constraint-violation margin and this margin is penalized in the objective [ZJM10]. It is not obvious how an infinite-horizon solution could be recovered in this case.

3. **Stochastic.** The family of stochastic optimal control problems is quite big and thus several interesting cases can be examined. The unconstrained counterpart of the LQR problem, the Linear Quadratic Gaussian controller can be computed in the case where the (linear) dynamics are affected by Gaussian process noise. Any attempt to introduce state and/or input constraints increases significantly the problem's difficulty and gives rise to a wide variety of formulations [Mes16] (probabilistic VS hard, joint VS individual etc.). It is common practice to fix the form of the control policy (typically affine) in order to deal with the increased complexity of these formulations. The addition of an infinite horizon is a topic open for experimentation.

4. **Large-scale systems.** Consider a physical system with a considerable amount of inputs and outputs, enough to be characterized 'large-scale'. There are several cases where the system can be represented in a 'computationally friendlier' way, where, *e.g.,* there exists a state-space transformation that decouples the system into smaller subsystems. This is the case with *symmetric* systems [CLP08; DB15]. In the framework of unconstrained optimal control, the latter property allows for the design of individual optimal LQR controllers for the decoupled subsystems. Unfortunately, when constraints are present, the coupling reappears. The application of decomposition methods in order to mitigate this coupling, or to, at least, tackle it in a computationally efficient way while preserving the infinite-horizon structure is an open question.

Note that the problems mentioned above can be combined with a variety of convex constraint sets for the inputs and/or the states. The solution proposed in this thesis is applicable to polytopic constraints only. More general convex sets should also be addressable since, in principle, the vanishing Lagrange multiplier sequence that permits the finite representation of the problem is a consequence of the linear complementarity KKT condition, *i.e.,* every Lagrange multiplier is an indicator of membership of a state-input pair with respect to (a subset of) an invariant set. The latter membership property should (eventually) hold for any convex set in the Lagrangian duality framework.

Wrapping up, there is a variety of directions that can be followed to extend the results of Chapter 3. What is left to be assessed is whether the infinite-horizon optimal control schemes that will emanate from these directions can do the job that traditional MPC does (i) better and (ii) at a comparable computational cost.

6.2 Asynchronous optimization

The proposed asynchronous inertial fixed-point iteration of Chapter 4 is an instance of a rapidly-growing family of asynchronous optimization algorithms. We see two distinct directions for improve-

ment and further development of these methods: computational improvements and reconsideration of their applicability in different contexts.

When it comes to computational speedup, injection of second-order information would seriously improve asynchronous (first-order) iterations, since problems of such large a scale tend to become ill-conditioned quickly. This suggestion has become commonplace for first-order methods and several successful attempts have been made toward this direction. There is, however, an evident absence of efforts to address this problem in asynchronous settings, probably due to the fact that this family of algorithms is still new and unexplored. To this end, we find promising recent advancements that treat splitting algorithms like FBS and DRS as unconstrained gradient iterations of smooth envelope functions [PSB14; STP17]. Adopting this viewpoint allows for the use of smooth unconstrained optimization algorithms, like Newton-type methods, with favorable convergence properties. The authors have already generalized the approach to devise superlinearly convergent KM iterations [TP17b], thus it would be valuable to try and embed this approach in an asynchronous framework, with the possible challenge of designing distributed quasi-Newton schemes.

The second aspect we want to discuss is the range of applications that asynchronous algorithms can serve. We insisted in Chapter 4 that, although currently these methods are heavily used in the machine learning community, there are worthwhile applications in contexts where heterogeneous populations of (physical) agents cooperate to solve a complex problem in a network structure, a description that pertains to *cyber-physical systems*, or, more specifically, to Internet of Things (IoT) applications. The asynchronous FBS scheme we proposed for the active dispatch of smart distribution networks only scratches the surface of the pool of potential applications.

1. Our scheme is limited to specific network topologies with one coordinator and several agents (star-shaped). A variety of other topologies can be tackled, with decentralized architectures being especially interesting.

2. Our scheme is applied to a specific smart grid-related application, but the potential of asynchronous optimization algorithms spans a wider range of applications. Take for example a setup similar to ours, *i.e.,* a coordinator and a (very) large population of agents. If the global signal that the coordinator broadcasts only depends on some characteristic of the aggregate population and not on the individual characteristics of the agents, the problem has a flavor of *mean field control* [SGL15]. Introducing asynchronicity to the agents' updates would result in novel and more practical versions of asynchronous mean field control. As a second example, one can consider sensor networks. Sensors typically run on low power requirements, thus having the flexibility to choose whether to transmit or not is crucial for extending their lifetime. An asynchronous scheme would enable such an option.

3. Our scheme only considers asynchronicity from the computational viewpoint, *i.e.,* it only takes into account the heterogeneity in the computation times of the optimizing agents. A more interesting setup emerges when the agents themselves operate at very *different timescales*, *i.e.,* when the dynamics differ significantly and the fastest systems evolve too fast for the algorithm to converge. Interesting questions arise both with respect to the asynchronous scheme's advantages over the synchronous one when it comes to early termination and with

respect to the *stability* properties of the overall network that might be jeopardized due to the highly suboptimal solution.

6.3 Learning a proximal problem

Chapter 3 deals with the subject of locating an optimizer of a special convex optimization problem, a process that can be viewed as online identification of a nonlinear mapping. Although the case we are tackling is quite specific in terms of the identified convex program's form, the idea of approximating or 'learning' an optimization process can be potentially implemented by employing more general tools developed by the machine learning (ML) community. From a technical viewpoint, the question of how to embed *structural* information about the process to be learned in a learning algorithm naturally arises. In the case of the proximal operator, *e.g.,* we would ideally want to use an ML method that incorporates information about smoothness and convexity of the Moreau envelope function.

In terms of applicability, we only used the scheme for potential communication reduction in a coordinator-driven setting. Let us consider a few more settings of potential interest.

1. In the context of *mechanism design*, the scheme we described could be used to infer preferences of competing players. In the design of games, it is commonly assumed that the marginal utility functions of the agents are concave. As a consequence, the market designer could set up a policy that forces the agents to solve proximal minimization problems so as to learn their preferences. This controversial policy would allow the designer to reach faster its system-level goal.

2. The problem of learning the value function of an MPC problem is of paramount importance to the control community and has led to the development of branches like explicit MPC, where the value function and the optimal control policy are computed a priori and are stored in a lookup table. This explicit representation of the controller is only possible for (very) small problems. A question possibly worth exploring is whether the proposed approach, essentially equivalent to learning the optimizer of a parametric convex program where the parameter enters quadratically in the cost, can be of any use to *approximate control policies.*

3. Our proposed process for 'learning' the set of optimizers can be described as follows:

   (a) The coordinator feeds an input vector to its local agent model.

   (b) If the resulting output, *i.e.,* the estimate of the optimizer, does not pass a certain validation test, the coordinator transmits the input vector to the agent.

   (c) The agent replies by sending the actual optimizer.

   (d) The coordinator corrects its agent model based on the newly received information from the agent.

   The framework described above has many similarities with the process of *online learning* [SS12]. Let us briefly give a description of the process based on the latter reference. Online learning is performed in a sequence of T consecutive rounds:

(a) On round t, the learner is first given a question, cast as a vector x_t, and is required to provide an answer to this question.

(b) The learner's prediction is performed based on a hypothesis, $h_t : \mathcal{X} \mapsto \mathcal{Y}$, where $\mathcal{X}$ is the set of questions and $\mathcal{Y}$ is the set of possible answers.

(c) After predicting an answer, the learner receives the correct answer to the question, denoted y_t, and suffers a loss according to a loss function $l(h_t, (x_t, y_t))$. The function l assesses the quality of the hypothesis h_t on the example (x_t, y_t).

(d) The ultimate goal of the learner is to minimize the cumulative loss he suffers along his run. To achieve this goal, the learner may choose a new hypothesis after each round so as to be more accurate in later rounds.

Finding the connections between the two problems might result in improvements of the proximal learning algorithm that we propose, both in terms of faster convergence and for verifying the quality of the approximation.

We are closing this section with a more general comment. This work focused strictly and deliberately on convex optimization algorithms. There is, however, a significant amount of work generated over the last few years on proximal splitting, and more general operator splitting methods, for nonconvex optimization problems [BST13; TP17a]. Distributed nonconvex optimization algorithms have been deployed to solve energy-network problems [LHSJ17; HJ17], and we recently had the first asynchronous nonconvex splitting schemes [Dav16]. Developments of the last few years have made nonconvexity a crucial issue in most ML applications, while at the same time, the majority of the physical systems in optimal control problems remains highly nonlinear. The explosive development of the aforementioned algorithms suggests that we should not shy away from directly tackling the nonconvexities. In our view, the preference in solving convex approximations of otherwise nonconvex problems should only stem from the fact that convex programs are more mature, and consequently less dependent on fine-tuning of parameters. Once nonconvex algorithms reach a similar state of maturity, there will be no reason why they should not be preferred.

Appendix A

Appendix

A.1 Derivation of the Alternating Direction Method of Multipliers from Douglas-Rachford Splitting

We consider the dual problem (2.4) and denote $h(\lambda) := f^\star(A^\top \lambda)$ and $f \in \Gamma_0(\mathbb{R}^n)$, $g \in \Gamma_0(\mathbb{R}^m)$. The DRS scheme constitutes of the three iterations:

$$v^{k+1} = \mathbf{prox}_{\gamma h}\left(\lambda^k - w^k\right) \tag{A.1}$$

$$\lambda^{k+1} = \mathbf{prox}_{\gamma g^\star}\left(v^{k+1} + w^k\right) \tag{A.2}$$

$$w^{k+1} = w^k + v^{k+1} - \lambda^{k+1} \ . \tag{A.3}$$

We are going to analyze the three iterations sequentially, following the approach form the lecture notes [Van10].

For (A.1), we have that

$$\mathbf{prox}_{\gamma h}\left(\lambda^k - w^k\right) = \underset{v\in\mathbb{R}^m}{\arg\min}\left\{h(v) + (1/2\gamma)\|v - \lambda^k + w^k\|_2^2\right\} \ ,$$

the optimality condition of which is

$$0 \in -A\partial f^\star(-A^\top v) + (1/\gamma)(v - \lambda^k + w^k) \tag{A.4}$$

Let us rewrite the proximal step (A.1) as a minimization update. For this purpose, consider the minimization problem

$$\text{minimize} \quad f(z) + (\gamma/2)\|Az + (\lambda^k - w^k)/\gamma\|_2^2$$

with variable $z \in \mathbb{R}^n$, which can be equivalently written as

$$\begin{array}{ll} \text{minimize} & f(z) + (\gamma/2)\|u\|_2^2 \\ \text{subject to} & Az + (\lambda^k - w^k)/\gamma = u, \end{array} \tag{A.5}$$

with variables $z \in \mathbb{R}^n, u \in \mathbb{R}^m$. Introducing a Lagrange multiplier $v \in \mathbb{R}^m$, the optimality conditions for problem (A.5) become:

$$-A^\top v \in \partial f(z), \quad \gamma u = v, \quad Az + (\lambda^k - w^k)/\gamma - u = 0 \ ,$$

which, by elimination of the variables z, u, can be written as

$$0 \in -A\partial f^\star(-A^\top v) + (1/\gamma)(v - \lambda^k + w^k),$$

which is (A.4). Furthermore, we have from (A.1) that

$$\begin{aligned} v^{k+1} - \lambda^k + w^k &\in -\gamma \partial h(v^{k+1}) \\ &= -\gamma(-A\partial f^\star(-A^\top v^{k+1})) \\ &= \gamma A z^{k+1} \ , \end{aligned}$$

where the first equality is due to the definition of h and z^{k+1} is a minimizer of problem (A.5). To wrap up, Step (A.1) can be written as

$$\begin{aligned} z^{k+1} &= \operatorname*{arg\,min}_{z \in \mathbb{R}^n} \left\{ f(z) + (\gamma/2)\|Az + (\lambda^k - w^k)/\gamma\|_2^2 \right\} \\ v^{k+1} &= \lambda^k - w^k + \gamma A z^{k+1} \ . \end{aligned} \tag{A.6}$$

Step (A.2) reads as $\lambda^{k+1} = \mathbf{prox}_{\gamma g^\star}\left(\lambda^k + \gamma A z^{k+1}\right)$. Using again Lemma 1, Step (A.2) is equivalent to

$$\begin{aligned} y^{k+1} &= \mathbf{prox}_{g/\gamma}\left(Az^{k+1} + \lambda^k/\gamma\right) \\ \lambda^{k+1} &= \lambda^k + \gamma(Az^{k+1} - y^{k+1}) \ . \end{aligned}$$

Finally, Step (A.3) results in $w^{k+1} = \gamma y^{k+1}$. Substituting w^{k+1} back to (A.6), we end up with the ADMM iterations.

Bibliography

[AA01] F. Alvarez and H. Attouch, "An Inertial Proximal Method for Maximal Monotone Operators via Discretization of a Nonlinear Oscillator with Damping", *Set-Valued Analysis*, vol. 9, no. 1, pp. 3–11, 2001.

[Abr17] J. Abrell, *The Swiss Wholesale Electricity Market*, https://www.ethz.ch/content/dam/ethz/special-interest/mtec/cer-eth/economics-energy-economics-dam/documents/people/jabrell/Abrell_Swiss_Wholesale_Electricity_Market.pdf, 2017.

[Alv04] F. Alvarez, "Weak convergence of a relaxed and inertial hybrid projection-proximal point algorithm for maximal monotone operators in Hilbert space", *SIAM Journal on Optimization*, vol. 14, no. 3, pp. 773–782, 2004.

[AM12] D. Axehill and M. Morari, "An alternative use of the riccati recursion for efficient optimization", *Systems & Control Letters*, vol. 61, no. 1, pp. 37 –40, 2012.

[Ant98] A. C. Antoulas, "Approximation of linear dynamical systems", in *Encyclopedia of Electrical and Electronics Engineering*, John Wiley and Sons, 1998, pp. 403–422.

[AP15] H. Attouch and J. Peypouquet, "The rate of convergence of Nesterov's accelerated forward-backward method is actually faster than $1/k^2$", *arXiv preprint arXiv:1510.08740*, 2015.

[BA12] L. M. Briceno-Arias, "Forward-Douglas-Rachford splitting and forward-partial inverse method for solving monotone inclusions", *arXiv preprint arXiv:1212.5942*, 2012.

[BC11] H. Bauschke and P. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer, 2011.

[Ber15] D. P. Bertsekas, *Convex Optimization Algorithms*. Athena Scientific, 2015.

[BPCPE11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers", *Found. Trends Mach. Learn.*, 2011.

[BST13] J. Bolte, S. Sabach, and S. Teboulle, "Proximal alternating linearized minimization for nonconvex and nonsmooth problems", *Mathematical Programming*, vol. 146, no. 1-2, pp. 459–494, 2013.

[BT09] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems", *SIAM J. Imaging Sci.*, 2009.

[BT14] A. Beck and M. Teboulle, "A fast dual proximal gradient algorithm for convex minimization and applications", *Operations Research Letters*, vol. 42, no. 1, pp. 1–6, 2014.

[BT89] D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and distributed computation*. Prentice Hall Inc., 1989.

[BV04] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[CCPV14] P. L. Combettes, L. Condat, J. C. Pesquet, and B. C. Vũ, "A forward-backward view of some primal-dual optimization methods in image recovery", in *The IEEE International Conference on Image Processing*, 2014, pp. 4141–4145.

[CD15] A. Chambolle and C. Dossal, "On the convergence of the iterates of "FISTA"", *Journal of Optimization Theory and Applications*, vol. Volume 166, no. Issue 3, p. 25, 2015.

[CE16] P. L. Combettes and J. Eckstein, "Asynchronous Block-Iterative Primal-Dual Decomposition Methods for Monotone Inclusions", *Mathematical Programming*, 2016.

[CLP08] R. Cogill, S. Lall, and P. A. Parrilo, "Structured semidefinite programs for the control of symmetric systems", *Automatica*, vol. 44, no. 5, pp. 1411 –1417, 2008.

[CM96] D. Chmielewski and V. Manousiouthakis, "On constrained infinite-time linear quadratic optimal control", *Systems & Control Letters*, vol. 29, no. 3, pp. 121 –129, 1996.

[CP07] P. L. Combettes and J.-C. Pesquet, "A Douglas-Rachford splitting approach to nonsmooth convex variational signal recovery", *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 564–574, Dec. 2007.

[CR12] I. CVX Research, *CVX: Matlab software for disciplined convex programming, version 2.0*, http://cvxr.com/cvx, Aug. 2012.

[CV14] P. L. Combettes and B. C. Vũ, "Variable metric forward–backward splitting with applications to monotone inclusions in duality", *Optimization*, vol. 63, no. 9, pp. 1289–1318, 2014.

[Dav15a] D. Davis, "Convergence rate analysis of the forward-Douglas-Rachford splitting scheme", *arXiv preprint arXiv:1410.2654*, 2015.

[Dav15b] D. Davis, "Convergence rate analysis of primal-dual splitting schemes", *SIAM Journal on Optimization*, vol. 25, no. 3, pp. 1912–1943, 2015.

[Dav16] ——, "The Asynchronous PALM Algorithm for Nonsmooth Nonconvex Problems", *arXiv preprint arXiv:1604.00526*, 2016.

[DB15] C. Danielson and S. Bauer, "Numerical decomposition of symmetric linear systems", in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 2061–2066.

[DHL17] I. Dunning, J. Huchette, and M. Lubin, "Jump: A modeling language for mathematical optimization", *SIAM Review*, vol. 59, no. 2, pp. 295–320, 2017. DOI: 10.1137/15M1020575.

[Dom13] A. Domahidi, *Methods and tools for embedded optimization and control*, Doctoral Thesis, 2013.

[DY16] D. Davis and W. Yin, "Convergence rate analysis of several splitting schemes", in *Splitting Methods in Communication, Imaging, Science, and Engineering*, R. Glowinski, S. J. Osher, and W. Yin, Eds. Springer International Publishing, 2016, pp. 115–163.

[DZZMJ12] A. Domahidi, A. Zgraggen, M. N. Zeilinger, M. Morari, and C. N. Jones, "Efficient Interior Point Methods for Multistage Problems Arising in Receding Horizon Control", in *The IEEE Conference on Decision and Control*, Maui, HI, USA, 2012, pp. 668 –674.

[EB92] J. Eckstein and D. Bertsekas, "On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators", *Math. Program.*, vol. 55, no. 3, pp. 293–318, 1992.

[EC10] J. M. Eyer and G. P. Corey, "Energy storage for the electricity grid : benefits and market potential assessment guide : a study for the DOE Energy Storage Systems Program.", Feb. 2010.

[FAJ14] H. R. Feyzmahdavian, A. Aytekin, and M. Johansson, "A delayed proximal gradient method with linear convergence rate.", in *Machine Learning for Signal Processing (MLSP), IEEE International Workshop*, 2014, pp. 1–6.

[FAVDFDJJJ17] H. Ferreau, S. Almèr, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, G. S. J.L. Jerez, and C. Jones, "Embedded Optimization Methods for Industrial Automatic Control ", *IFAC-PapersOnLine*, vol. 50, pp. 13 194–13 209, 1 2017.

[FBD08] H. J. Ferreau, H. G. Bock, and M. Diehl, "An online active set strategy to overcome the limitations of explicit MPC", *International Journal of Robust and Nonlinear Control*, 2008.

[FGNSPJ17] L. Fabietti, T. T. Gorecki, E. Namor, F. Sossan, M. Paolone, and C. N. Jones, "Dispatching active distribution networks through electrochemical storage systems and demand side management", in *2017 IEEE Conference on Control Technology and Applications (CCTA)*, 2017, pp. 1241–1247.

[FGQBLJ17] L. Fabietti, T. Gorecki, F. Qureshi, A. Bitlislioglu, I. Lymperopoulos, and C. Jones, "Experimental implementation of frequency regulation services using commercial buildings", *IEEE Transactions on Smart Grid*, vol. PP, no. 99, pp. 1–1, 2017.

[FJ13a] G. Frison and J. B. Jørgensen, "A fast condensing method for solution of linear-quadratic control problems", in *Proceedings of the 52nd IEEE Conference on Decision and Control, CDC 2013, December 10-13, 2013, Firenze, Italy*, 2013, pp. 7715–7720.

[FJ13b] ——, "Efficient implementation of the Riccati recursion for solving linear-quadratic control problems.", in *CCA*, IEEE, 2013, pp. 1117–1122.

[FSKJ16] L. Ferranti, G. Stathopoulos, T. Keviczky, and C. N. Jones, "Constrained LQR Using Online Decomposition Techniques ", in *Decision and Control (CDC), 2016 IEEE 55th Annual Conference on*, 2016.

[FZ84] A. Federgruen and P. Zipkin, "Computational Issues in an Infinite-Horizon, Multiechelon Inventory Model ", *Operations Research*, vol. 32, pp. 818–836, 4 1984.

[G91] O. Güler, "On the Convergence of the Proximal Point Algorithm for Convex Minimization", *SIAM Journal on Control and Optimization*, vol. 29, no. 2, pp. 403–419, 1991.

[GB] P. Giselsson and S. Boyd, "Linear Convergence and Metric Selection in Douglas Rachford Splitting and ADMM", *To appear in IEEE Transactions on Automatic Control*,

[GB14a] ——, "Diagonal scaling in Douglas-Rachford splitting and ADMM", *53rd IEEE Conference on Decision and Control*, 2014.

[GB14b] ——, "Monotonicity and restart in fast gradient methods", in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, 2014, pp. 5058–5063.

[GB15] ——, "Metric selection in fast dual forward backward splitting", *Automatica*, vol. 62, pp. 1–10, 2015.

[GBSJ15] T. T. Gorecki, A. Bitlislioglu, G. Stathopoulos, and C. N. Jones, "Guaranteeing input tracking for constrained systems: theory and application to demand response", in *American Control Conference (ACC).*, 2015, pp. 232–237.

[GBTM04] P. Grieder, F. Borrelli, F. Torrisi, and M. Morari, "Computation of the constrained infinite time linear quadratic regulator", *Automatica*, vol. 40, no. 4, pp. 701 –708, 2004.

[GFQJ17] T. T. Gorecki, L. Fabietti, F. A. Qureshi, and C. N. Jones, "Experimental demonstration of buildings providing frequency regulation services in the swiss market", *Energy and Buildings*, vol. 144, pp. 229 –240, 2017.

[Goh17] G. Goh, "Why momentum really works", *Distill*, 2017. DOI: 10.23915/distill.00006. [Online]. Available: http://distill.pub/2017/momentum.

[GOP17] M. Gurbuzbalaban, A. Ozdaglar, and P. Parrilo, "On the convergence rate of incremental aggregated gradient algorithms", *SIAM Journal on Optimization*, 2017.

[GQJ15] T. T. Gorecki, F. A. Qureshi, and C. N. Jones, *Openbuild: An integrated simulation environment for building control*, 2015.

[GTSJ15] E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson, "Optimal parameter selection for the alternating direction method of multipliers (admm): Quadratic problems", *IEEE Transactions on Automatic Control*, vol. 60, no. 3, pp. 644–658, 2015.

[Gül92] O. Güler, "New proximal point algorithms for convex minimization", *SIAM Journal on Optimization*, vol. 2, no. 4, pp. 649–664, 1992.

[HJ17] J.-H. Hours and C. N. Jones, "An alternating trust region algorithm for distributed linearly constrained nonlinear programs, application to the optimal power flow problem", *Journal of Optimization Theory and Applications*, vol. 173, no. 3, pp. 844–877, 2017.

[HKJM13] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0", in *Proc. of the European Control Conference*, http://control.ee.ethz.ch/~mpt, Zürich, Switzerland, 2013, pp. 502–510.

[IH16] F. Iutzeler and M. J. Hendrickx, "A Generic Linear Rate Acceleration of Optimization algorithms via Relaxation and Inertia", *arXiv preprint arXiv:1603.05398v2*, 2016.

[Jon12] C. N. Jones, *Model predictive control*, Course Notes, 2012.

[Kal60] R. Kalman, "Contributions to the theory of optimal control", *Boletin de la Sociedad Matematica Mexicana*, 1960.

[Kra55] A. Krasnosel'skiĭ, "Two remarks on the method of successive approximations", *Uspekhi Matematicheskikh Nauk*, vol. 10, no. 1, pp. 123–127, 1955.

[LÖ4] J. Löfberg, "Yalmip : A toolbox for modeling and optimization in MATLAB", in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[LFP16] J. Liang, J. Fadili, and G. Peyré, "Convergence rates with inexact non-expansive operators", *Mathematical Programming*, vol. 159, no. 1, pp. 403–434, 2016.

[LHSJ17] Y. Liu, J. H. Hours, G. Stathopoulos, and C. N. Jones, "Real-time distributed algorithms for nonconvex optimal power flow", in *2017 American Control Conference (ACC)*, 2017, pp. 3380–3385.

[Lor15] D. Lorenz, "The heavy ball method as 'perturbed gradient method'", *Blog: regularize*, 2015. [Online]. Available: https://regularize.wordpress.com/2015/06/30/the-heavy-ball-method-as-perturbed-gradient-method/#eqheavyball1.

[LS97] C. Lemaréchal and C. Sagastizábal, "Practical Aspects of the Moreau–Yosida Regularization: Theoretical Preliminaries", *SIAM Journal on Optimization*, vol. 7, no. 2, pp. 367–385, 1997.

[LW15] J. Liu and S. J. Wright, "Asynchronous stochastic coordinate descent: Parallelism and convergence properties", *SIAM Journal on Optimization*, vol. 25, no. 1, pp. 351–376, 2015.

[Mai08] P.-E. Maingé, "Convergence theorems for inertial KM-type algorithms ", *Journal of Computational and Applied Mathematics*, vol. 219, no. 1, pp. 223 –236, 2008.

[Man53] R. Mann, "Mean value methods in iteration", *Proceedings of the American Mathematical Society*, vol. 4, no. 3, pp. 506–510, 1953.

[Mar70] B. Martinet, "Régularisation d'inéquations variationnelles par approximations successives", *Revue Française de Informatique et Recherche Opérationelle*, vol. 4, no. R3, pp. 154–158, 1970.

[Mar72] ——, "Détermination approchée d'un point fixe d'une application pseudo-contractante", *C.R. Acad. Sci. Paris*, vol. 274A, pp. 163–165, 1972.

[MB12] J. Mattingley and S. Boyd, "CVXGEN: a code generator for embedded convex optimization", *Optimization and Engineering*, 2012.

[Mes16] A. Mesbah, "Stochastic model predictive control: An overview and perspectives for future research ", *IEEE Control Systems*, vol. 36, pp. 30–44, 2016.

[MKJSJRT17] C. Ma, J. Konecný, M. Jaggi, V. Smith, M. I. Jordan, P. Richtárik, and M. Takác, "Distributed optimization with arbitrary local solvers", *Optimization Methods Software*, vol. 32, no. 4, pp. 813–848, Jul. 2017, ISSN: 1055-6788.

[MO03] A. Moudafi and M. Oliny, "Convergence of a splitting inertial proximal method for monotone operators ", *Journal of Computational and Applied Mathematics*, vol. 155, no. 2, pp. 447 –454, 2003.

[Mor62] J. J. Moreau, "Fonctions convexes duales et points proximaux dans un espace hilbertien.", *Comptes Rendus de l'Académie des Sciences (Paris), Série A*, vol. 255, 1962.

[Mor65] ——, "Proximité et dualité dans un espace Hilbertien", *Bulletin de la Société Mathématique de France*, vol. 93, no. 2, pp. 273–299, 1965.

[MZHR16] I. Mitliagkas, C. Zhang, S. Hadjis, and C. Ré, "Asynchrony begets momentum, with an application to deep learning", in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2016, pp. 997–1004.

[Nes04a] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer, 2004, vol. 87.

[Nes04b] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. 2004.

[Nes83] ——, "A method for solving the convex programming problem with convergence rate $\mathcal{O}(1/k^2)$", *Dokl. Akad. Nouk SSSR*, 1983.

[NP15] I. Necoara and A. Patrascu, "DuQuad: an inexact (augmented) dual first order algorithm for quadratic programming", *arXiv preprint arXiv:1504.05708*, 2015.

[Nura] A. Nursimulu, *Demand-Side Flexibility for Energy Transitions: Policy Recommendations for Developing Demand Response.* EPFL Energy Center and International Risk Governance Center, Policy Brief.

[Nurb] ——, *Demand-Side Flexibility for Energy Transitions: Ensuring the Competitive Development of Demand Response Options.*

[NY83] A. Nemirovski and D. B. Yudin, *Problem complexity and method efficiency in optimization*, ser. Wiley-Interscience series in discrete mathematics. Chichester, New York: Wiley-Interscience, 1983, ISBN: 0-471-10345-4. [Online]. Available: http://opac.inria.fr/record=b1091338.

[OBP15] P. Ochs, T. Brox, and T. Pock, "iPiasco: Inertial Proximal Algorithm for strongly convex Optimization", *Journal of Mathematical Imaging and Vision*, vol. 53, no. 2, pp. 171–181, 2015.

[OSB13] B. O'Donoghue, G. Stathopoulos, and S. P. Boyd, "A splitting method for optimal control.", *IEEE Trans. Contr. Sys. Techn.*, vol. 21, no. 6, pp. 2432–2442, 2013.

[Pas77] G. Passty, "Ergodic convergence to a zero of the sum of monotone operators in hilbert space", *J. Math. Anal. Appl.*, 1977.

[PB14a] N. Parikh and S. Boyd, "Proximal algorithms", *Foundations and Trends in Optimization*, vol. 1, no. 3, 2014.

[PB14b] P. Patrinos and A. Bemporad, "An accelerated dual gradient-projection algorithm for embedded linear model predictive control", *Automatic Control, IEEE Transactions on*, vol. 59, no. 1, pp. 18–33, 2014.

[Pol64] B. Polyak, "Some methods of speeding up the convergence of iteration methods", *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1 –17, 1964.

[Pol87] B. Polyak, "Introduction to Optimization", *Optimization Software*, 1987.

[PSB14] P. Patrinos, L. Stella, and A. Bemporad, "Douglas-Rachford splitting: Complexity estimates and accelerated variants", in *The 53rd IEEE Annual Conference on Decision and Control*, 2014, pp. 4234–4239.

[PXYY16] Z. Peng, Y. Xu, M. Yan, and W. Yin, "ARock: an Algorithmic Framework for Asynchronous Parallel Coordinate Updates", *SIAM Journal on Scientific Computing*, vol. 38, no. 5, 2016.

[RB16] E. Ruy and S. Boyd, "A Primer on Monotone Operator Methods", *Applied and Computational Mathematics an International Journal*, vol. 15, no. 1, 2016.

[RFP13] H. Raguet, J. Fadili, and G. Peyré, "A Generalized Forward-Backward Splitting", *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1199–1226, 2013.

[RJM09] S. Richter, C. N. Jones, and M. Morari, "Real-time input-constrained MPC using fast gradient methods", in *Proceedings of the 48th IEEE Conference on Decision and Control, CDC 2009, combined withe the 28th Chinese Control Conference, December 16-18, 2009, Shanghai, China*, 2009, pp. 7387–7393.

[RL15] H. Raguet and L. Landrieu, "Preconditioning of a generalized forward-backward splitting and application to optimization on graphs", *SIAM Journal on Imaging Sciences*, vol. 8, no. 4, pp. 2706–2739, 2015.

[RM09] J. Rawlings and D. Mayne, *Model Predictive Control: Theory and Design*. Nob Hill Pub., 2009, ISBN: 9780975937709. [Online]. Available: https://books.google.ch/books?id=3_rfQQAACAAJ.

[RN04] M. Rabbat and R. Nowak, "Distributed optimization in sensor networks", in *Third International Symposium on Information Processing in Sensor Networks, 2004. IPSN 2004*, 2004, pp. 20–27.

[Roc70] R. T. Rockafellar, *Convex Analysis*. Princeton University Press, 1970.

[Roc76] R. T. Rockafellar, "Monotone Operators and the Proximal Point Algorithm", *SIAM Journal on Control and Optimization*, 1976.

[RW98] R. Rockafellar and R. J.-B. Wets, *Variational Analysis*. Heidelberg, Berlin, New York: Springer Verlag, 1998.

[San10] C. Sanderson, "Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments", NICTA, Tech. Rep., 2010.

[SBC14] W. Su, S. Boyd, and E. Candes, "A Differential Equation for Modeling Nesterov's Accelerated Gradient Method: Theory and Insights", in *Advances in Neural Information Processing Systems 27*, Curran Associates, Inc., 2014, pp. 2510–2518.

[SD87] M. Sznaier and M. Damborg, "Suboptimal control of linear systems with state and control inequality constraints", in *IEEE Conference on Decision and Control*, 1987.

[SDDGD17] S. Soori, A. Devarakonda, J. Demmel, M. Gurbuzbalaban, and M. Dehnavi, "Avoiding Communication in Proximal Methods for Convex Optimization Problems", *arXiv preprint arXiv:1710.08883*, 2017.

[SFMTJ16] V. Smith, S. Forte, C. Ma, J. M. Takác Martin, and M. Jaggi, "Cocoa: A general framework for communication-efficient distributed optimization", *arXiv preprint arXiv:1611.02189*, 2016.

[SGL15] M. C. S. Grammatico F. Parise and J. Lygeros, "Decentralized convergence to Nash equilibria in constrained mean field control", *IEEE Transactions on Automatic Control*, 2015.

[SGMS16] D. Sturzenegger, D. Gyalistras, M. Morari, and R. S. Smith, "Model Predictive Climate Control of a Swiss Office Building: Implementation, Results, and Cost-Benefit Analysis", *IEEE Transactions on Control Systems Technology*, vol. 24, no. 1, pp. 1–12, 2016.

[SJ18a] G. Stathopoulos and C. N. Jones, "An Inertial Parallel and Asynchronous Forward-Backward Iteration for Distributed Convex Optimization", *Submitted to Journal of Optimization Theory and Applications*, 2018.

[SJ18b] ——, "Communication reduction in distributed optimization via estimation of the proximal operator", *Submitted to IEEE Transactions on Control of Network Systems*, 2018.

[SJ18c] G. Stathopoulos and C. N. Jones, "A coordinator-driven communication reduction scheme for distributed optimization using the projected gradient method", in *Proceedings of the 17th IEEE European Control Conference, ECC 2018, Limassol, Cyprus*, 2018.

[SKJ14] G. Stathopoulos, M. Korda, and C. N. Jones, "Solving the infinite-horizon constrained LQR problem using splitting techniques", in *19th IFAC World Congress*, 2014.

[SKJ17] ——, "Solving the Infinite-Horizon Constrained LQR Problem using Accelerated Dual Proximal Methods", *IEEE Transactions on Automatic Control*, vol. 62, no. 4, pp. 1752–1767, 2017.

[SNCP16] F. Sossan, E. Namor, R. Cherkaoui, and M. Paolone, "Achieving the Dispatchability of Distribution Feeders Through Prosumers Data Driven Forecasting and Model Predictive Control of Electrochemical Storage", *IEEE Transactions on Sustainable Energy*, vol. 7, no. 4, pp. 1762–1777, 2016.

[SR98] P. Scokaert and J. B. Rawlings, "Constrained Linear Quadratic Regulation", *IEEE Transactions on Automatic Control*, vol. 43, no. 8, pp. 1163–1169, 1998.

[SRB11] M. W. Schmidt, N. L. Roux, and F. R. Bach, "Convergence Rates of Inexact Proximal-Gradient Methods for Convex Optimization", in *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011, Granada, Spain.*, 2011, pp. 1458–1466.

[SS12] S. Shalev-Shwartz, "Online learning and online convex optimization", *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012. [Online]. Available: http://dx.doi.org/10.1561/2200000018.

[SSPJ14] G. Stathopoulos, A. Szucs, Y. Pu, and C. N. Jones, "Splitting methods in control", in *Proceedings of the 13th IEEE European Control Conference, ECC 2014, Strasbourg, France*, 2014, pp. 2478 –2483.

[SSSPJ16] G. Stathopoulos, H. Shukla, A. Szucs, Y. Pu, and C. N. Jones, "Operator splitting methods in control", *Foundations and Trends® in Systems and Control*, vol. 3, no. 3, pp. 249–362, 2016.

[ST14] R. Shefi and M. Teboulle, "Rate of Convergence Analysis of Decomposition Methods Based on the Proximal Method of Multipliers for Convex Minimization", *SIAM Journal on Optimization*, 2014.

[STP17] L. Stella, A. Themelis, and P. Patrinos, "Forward–backward quasi-newton methods for nonsmooth optimization problems", *Computational Optimization and Applications*, vol. 67, no. 3, pp. 443–487, 2017.

[Stu98] J. F. Sturm, *Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones*, 1998.

[Swi03] Swissgrid, *Test for secondary control capability*, http://www.swissgrid.ch/dam/swissgrid/experts/ancillary_services/prequalification/D130422_Test-for-secondary-control-capability_V2R1_EN.pdf, 2003.

[THG17] A. B. Taylor, J. M. Hendrickx, and F. Glineur, "Smooth strongly convex interpolation and exact worst-case performance of first-order methods", *Mathematical Programming*, vol. 161, no. 1, pp. 307–345, 2017.

[TP17a] A. Themelis and P. Patrinos, "Douglas-Rachford splitting and ADMM for nonconvex optimization: tight convergence results", *arXiv preprint arXiv:1709.05747*, 2017.

[TP17b] ——, "SuperMann: a superlinearly convergent algorithm for finding fixed points of nonexpansive operators", *arXiv preprint arXiv:1609.06955*, 2017.

[Tse08] P. Tseng, "On accelerated proximal gradient methods for convex-concave optimization", *submitted to SIAM Journal on Optimization*, 2008.

[Tse91] ——, "Applications of splitting algorithm to decomposition in convex programming and variational inequalities", *SIAM J. Control Optim.*, vol. 29, no. 1, pp. 119–138, 1991.

[Ull11] E. Ullmann, *A Matlab toolbox for C-code generation for first order methods*, Master's thesis, ETH Zurich, 2011.

[Uza60] H. Uzawa, "Walras' tâtonnement in the theory of exchange", *The Review of Economic Studies*, vol. 27, no. 3, pp. 182–194, 1960.

[Van10] L. Vandenberghe, *Optimization methods for large-scale systems*, UCLA EE 236C lecture notes, 2010.

[VKMAC17a] E. Vrettos, E. C. Kara, J. MacDonald, G. Andersson, and D. S. Callaway, "Experimental Demonstration of Frequency Regulation by Commercial Buildings - Part I: Modeling and Hierarchical Control Design", *IEEE Transactions on Smart Grid*, vol. PP, no. 99, pp. 1–1, 2017, ISSN: 1949-3053. DOI: 10.1109/TSG.2016.2628897.

[VKMAC17b] ——, "Experimental Demonstration of Frequency Regulation by Commercial Buildings - Part II: Results and Performance Evaluation", *IEEE Transactions on Smart Grid*, vol. PP, no. 99, pp. 1–1, 2017.

[Wal96] L. Walras, "Éléments d'économie politique pure, ou théorie de la richesse sociale.", *F. Rouge*, 1896.

[Wri15] S. J. Wright, "Coordinate descent algorithms", *Mathematical Programming*, vol. 151, no. 1, pp. 3–34, 2015.

[ZJM10] M. N. Zeilinger, C. N. Jones, and M. Morari, "Robust stability properties of soft constrained mpc", in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5276–5282.

GIORGOS STATHOPOULOS

PERSONAL INFORMATION

email	georgios.stathopoulos@epfl.ch
website	http://people.epfl.ch/georgios.stathopoulos
phone	(M) +41 (0) 787 585 903

RESEARCH INTERESTS

- **Convex optimization algorithms & theory**

  Developing first order convex optimization algorithms with applications to small embedded-control problems and/or large-scale optimization problems

- **Demand-response for smart buildings**

  Proposing demand-response policies for commercial buildings and characterizing a building's flexibility by means of robust optimization techniques

- **Distributed model predictive control of smart grids**

  Application of distributed optimization to predictive control problems at the level of distribution networks

EDUCATION

2012-2018 École Polytechnique Fédérale de Lausanne, Switzerland

PhD

School: Mechanical Engineering
Thesis: *Distributed Optimization and Control using Operator Splitting Methods*
Description: Design decomposition methods that tackle both small-scale centralized control problems and larger-scale multi-agent distributed control problems. Devise faster algorithms for embedded applications and reduced-communication schemes for multi-agent applications. Develop optimizing contollers capable of coordinating the flow of power to and from large networks of smart buildings in order to offer critical services to the power grid. Optimize buildings' usage for several purposes and on different abstraction levels.
Advisor: Prof. Colin. N. Jones

2010-2012 Delft University of Technology, The Netherlands

MSc

School: Mechanical Engineering
Program: *Systems & Control*
Thesis: *Fast Optimization-based Control and Estimation Using Operator Splitting Methods*, conducted at Stanford University
Description: Participated in the development of an algorithm that solves large scale optimization problems at very high speeds. Applied to supply chain management and portfolio optimization problems. The algorithm has since become a benchmark for speed comparisons with other methods.
Advisors: Prof. Tamás Keviczky, Prof. Stephen Boyd, Prof. Bart De Schutter

2003-2009 Patras University, Greece

Diploma

School: Electrical and Computer Engineering
Thesis: *Robust Control and Fault Detection for a Flexible Link Manipulator*
Description: Development of an adaptive controller for a flexible link

manipulator, where a Set Membership Identifier computes the feasible set for the model parameters and the controller tunes its gains through an on-line minimization of a cost function.
Advisor: Prof. Anthony Tzes

HONORS AND AWARDS

Graduated *Cum Laude*, MSc in Systems and Control, Delft Uni. of Technology

Justus and Louise van Effen scholarship grant, 2011

EXPERIENCE

4-9/2017 LATERITE, Data Advisory

Contractor-Consultant

Description: Designed a systems dynamic model of Rwanda's education system using modern machine learning techniques. The model is under development and currently targeted to inform the next Education Sector Strategic Plan of the Government of Rwanda.

7-11/2016 Mitsubishi Electric Research Laboratories, Cambridge MA

Summer intern

Description: Evaluated algorithms for solving constrained optimal control problems, suggested improvements to existing methods and derived new approaches specific to the control of thermodynamic machines such as vapor compression systems.

PUBLICATIONS

Submitted Manuscripts

G. Stathopoulos and C. N. Jones. *Communication reduction in distributed optimization via estimation of the proximal operator*.

G. Stathopoulos and C. N. Jones. *An Inertial Parallel and Asynchronous Forward-Backward Iteration for Distributed Convex Optimization*.

Journal Articles

G. Stathopoulos, M. Korda and C. N. Jones. *Solving the Infinite-horizon Constrained LQR Problem using Accelerated Dual Proximal Methods*. In IEEE Transactions on Automatic Control, vol. 62, no.4, pp.17521767, 2017

G. Stathopoulos, H. Shukla, A. Szücs, Y. Pu and C. N. Jones. *Operator Splitting Methods in Control*. In Foundations and Trends® in Systems and Control, vol.3, no.3, pp.249-362, 2016

B. O' Donoghue, **G. Stathopoulos** and S. Boyd. *A Splitting Method for Optimal Control*. In IEEE Transactions on Control Systems Technology, vol. 21, no.6, pp.2432-2442, 2013

Conference Articles

G. Stathopoulos and C. N. Jones, *A coordinator-driven communication reduction scheme for distributed optimization using the projected gradient method*, in Proceedings of the 17th IEEE European Control Conference, ECC 2018, Limassol, Cyprus, 2018

H. Ferreau, S. Almèr, R. Verschueren, M. Diehl, D. Frick, A. Domahidi, **G. Stathopoulos**, J.L. Jerez, and C. N. Jones, *Embedded Optimization Methods for Industrial Automatic Control*, IFAC- PapersOnLine, vol. 50, pp. 13 19413 209, 1 2017

Y. Liu, J. H. Hours, **G. Stathopoulos**, and C. N. Jones, Real-time distributed algorithms for nonconvex optimal power flow, in 2017 American Control Conference (ACC), 2017

T. T. Gorecki, A. Bitlislioğlu, **G. Stathopoulos** and C. N. Jones. *Guaranteeing Input Tracking For Constrained Systems: Theory and Application to Demand Response*. American Control Conference, Chicago, Illinois, USA, 2015

G. Stathopoulos, M. Korda and C. N. Jones. *Solving the infinite-horizon constrained LQR problem using splitting techniques*. 19th IFAC World Congress, Cape Town, South Africa, 2014

L. Ferranti, **G. Stathopoulos**, T. Keviczky, and C. N. Jones, *Constrained LQR Using Online Decomposition Techniques*, in Decision and Control (CDC), 2016 IEEE 55th Annual Conference on, 2016

G. Stathopoulos, A. Szücs, Y. Pu and C. N. Jones. *Splitting methods in control*. 13th European Control Conference, Strasbourg, France, 2014

G. Stathopoulos, T. Keviczky and Y. Wang. *A hierarchical time-splitting approach for solving finite-time optimal control problems*. 12th European Control Conference, Zurich, Switzerland, 2013

N. Karamolegkos, **G. Stathopoulos** and A. Tzes. *Adaptive Minimum Uncertainty Control for a Flexible Link Manipulator*. 17th Mediterranean Conference on Control and Automation, Thessaloniki, Greece, 2009

TECHNICAL SKILLS

Scientific software / Programming: R, Matlab, Julia, C, C++

Other: LaTeX, Linux, MacOs, Microsoft Windows

OTHER SKILLS

Languages: Greek (native), English (C2 Level), French (B1 Level)

PERSONAL DEVELOPMENT

Volunteer Experience: Worked with the NPO SAVE Foundation to teach computer literacy and science courses to high school children who live in townships in the outskirts of Cape Town, South Africa. September 2014.

MOOCs: Challenges of Global Poverty, MIT, edX. Certified.

The Analytics Edge, MIT, edX. Certified.