UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
CORSO DI LAUREA IN INGEGNERIA DELL'AUTOMAZIONE

Distributed Convex Optimisation using the Alternating Direction Method of Multipliers (ADMM) in Lossy Scenarios

Laureando:
Nicola BASTIANELLO

Relatore:
Prof. Ruggero CARLI

PADOVA, 16 APRILE 2018

ANNO ACCADEMICO 2017/2018

Abstract

The interest for algorithms suited to solve convex optimisation problems in a distributed fashion has recently been motivated by many applications from Machine Learning, to wireless sensor networks, to Smart Grids, to Image Processing and many other. A particularly well suited algorithm for this class of problems is the Alternating Direction Method of Multipliers (ADMM) that shares ties with, on the one hand, augmented Lagrangian methods and, on the other, with splitting operator theory. After a thorough literature and theoretical background review, this Thesis proposes a new formulation of the ADMM that is robust to faulty communications among the agents collaborating to the solution of the problem and, moreover, that allows for local computations to be carried out asynchronously. The convergence of the proposed ADMM formulation is proved in a probabilistic framework and, restricting to the case of smooth and strongly convex costs, the rate of convergence is shown to be exponential and an explicit upper bound is given. The proposed algorithm is then extended to solve problems conforming to the partition-based framework, in which the local cost of each agent depends on local information and also information from its neighbours. The theoretical results presented are validated with extensive Monte Carlo simulations, performed on quadratic and linear regression problems.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ADMM	**A**lternating **D**irection **M**ethod of **M**ultipliers
R-ADMM	**R**elaxed **A**lternating **D**irection **M**ethod of **M**ultipliers
r-ADMM	**r**obust **A**lternating **D**irection **M**ethod of **M**ultipliers
ra-ADMM	**r**obust and **a**synchronous **A**lternating **D**irection **M**ethod of **M**ultipliers
ran-ADMM	**ran**domised **A**lternating **D**irection **M**ethod of **M**ultipliers
PPA	**P**roximal **P**oint **A**lgorithm
KM	**K**rasnosel'skiĭ-**M**ann iteration
s-KM	**s**tochastic **K**rasnosel'skiĭ-**M**ann iteration
PRS	**P**eaceman**R**achford **S**plitting
R-PRS	**R**elaxed **P**eaceman**R**achford **S**plitting
DRS	**D**ouglas-**R**achford **S**plitting
FBS	**F**orward-**B**ackward **S**plitting
KKT	**K**arush-**K**uhn-**T**ucker
ra-NR	**r**obust and **a**synchronous **N**ewton-**R**aphson
LASSO	**L**east **A**bsolute **S**hrinkage and **S**election **O**perator
WSN	**W**ireless **S**ensor **N**etwork

Notation

$\mathbb{N}$	field of natural numbers
$\mathbb{R}$	field of real numbers
$\mathbb{R} \cup \{+\infty\}$	extended real line
$\mathcal{X}$	real Hilbert space
$\mathbb{E}[\cdot]$	mathematical expectation
$\mathbb{P}[\cdot]$	probability of an event
$A \otimes B$	Kronecker product
$\mathrm{vect}(\cdot)$	vectorisation operator
$\mathbb{0}_n$	$n \times 1$ vector of zeros
$\mathbb{1}_n$	$n \times 1$ vector of ones
$\iota_{\mathcal{C}}(\cdot)$	indicator function of set $\mathcal{C}$
∇f	gradient of function f, by convention a column vector
∂f	subdifferential of non-smooth function f
$\bar{\nabla} f$	subgradient of f, $\bar{\nabla} f \in \partial f$
$\mathrm{dom}(f)$	domain of function f
$\mathrm{fix}(T)$	set of fixed points of mapping T
$\mathrm{zer}(A)$	set of zeros of set-valued operator A
f^*	convex conjugate of function f
$\mathrm{prox}_{\rho f}(\cdot)$	proximal operator of function f with penalty ρ
$\mathrm{refl}_{\rho f}(\cdot)$	reflective operator of function f with penalty ρ
$A \succ \mathbb{0}_{n \times n}$	matrix $A \in \mathbb{R}^{n \times n}$ is positive definite

Chapter 1

Introduction

1.1 Introduction

Distributed optimisation algorithms have recently become the focus of intense research, motivated by the increasing availability of large data sets, the so-called *Big Data*. Indeed in many scenarios it is increasingly easy and affordable to collect and store large amounts of data, which cannot be analysed or even stored in a centralised fashion. Examples of fields that are characterised by the possibility of collecting Big Data range from the Internet and global-scale communications, to social media and ubiquitous mobile communication devices, to surveillance cameras, medical and e-commerce applications [95, 10]. The availability of economical (wireless) sensors that can be deployed in different scenarios contributes to the diffusion of large data sets as well. And indeed the interest for algorithms that can be applied to sample averaging for sensor networks, distributed learning in wireless sensor networks (WSN) and Smart Grids has recently peaked [66, 50].

As hinted above, the hallmarks of Big Data are: the size of the data sets, which can total up to million of data points; the high dimensionality of these data points, favoured by the inexpensive collection and storage apparatuses; the often-times distributed nature of these datasets [10]. Therefore in order to extract value from Big Data it is necessary to develop efficient algorithms that can cope with the size and nature of these data sets, and in many applications also be able to run online. The central idea is to apply *divide et impera* schemes that assign a subset of the available data to each of a set of agents, for instance computers or sensors in a network, which then collaborate with each other in order to reach a *consensus* solution [67].

Big Data analytics finds its applications in a diverse range of scenarios, from signal processing and network analysis [95], to Machine Learning [10], to classic Control [29], to bioinformatics and health [66], and – to conclude this brief and incomplete overview – to Smart Grids [100]. In many of these fields it is possible to formulate the data analytics problems as convex optimisation problems, hence the particular focus of this work on convex problems. For instance in Machine Learning the template of the most common training problems is

$$\min_x \{\mathcal{L}(x; A, b) + \lambda \mathcal{R}(x)\}$$

where $\mathcal{L}(x; A, b)$ is a function that weights the accuracy, or *fitness*, of the solution computed on the data (A, b), and $\mathcal{R}(x; \lambda)$ a *regularisation* function which imposes a certain structure on the solution [80]. Both functions are usually convex but possibly non-smooth, as in the case of the Lasso problem [98] which is characterised by $\mathcal{L}(x; a, b) = \|Ax - b\|^2$ and $\mathcal{R}(x) = \|x\|_1$: both convex, but the latter is not smooth.

In order to successfully apply distributed optimisation algorithms in real-life scenarios it is important to keep into account two issues: the potential loss of communications between the co-operating agents, and the fact that the different capabilities of these agents might lead to asynchrony. While asynchrony can be leveraged by design in order to speed up the

execution of an algorithm, and indeed has been the topic of many works [6], the robustness of distributed optimisation algorithms to packet losses has seen less interest, despite the fact that in many applications such as WSN and control [94, 90, 115] it is an intrinsic problem that merits careful consideration from a rigorous point of view.

1.2 Convex Optimisation Literature Review

The present Section describes the state-of-the-art in distributed optimisation and fixed point algorithms, which will be shown to have strong ties to convex optimisation and the Alternating Direction Method of Multipliers (ADMM), the focus of this Thesis.

1.2.1 Distributed optimisation

Much research has been devoted to the design of efficient algorithms for solving distributed (convex) optimisation problems, given the interest induced by the advent of Big Data. This Section will present a brief overview of the current state-of-the-art algorithms and results, leaving out methods based on the ADMM which will be the focus of Section 1.3 below. Where available, robust and asynchronous versions of the mentioned algorithms will be referenced.

A first class of distributed optimisation algorithms are the so-called *subgradient methods*, first proposed in [73], which require that each agent compute locally a subgradient of the assigned cost function. The advantage of these methods is that they do not require the cost functions to be smooth, but they exhibit sub-linear convergence [74]. Subgradient methods have been extended to work over time-varying graphs both in a discrete-time [72] and a continuous-time settings [61]. The convergence is guaranteed by the choice of a suitably decreasing step-size which, however, depends on the global time and therefore in practice requires that all agents work synchronously.
In the case of smooth cost functions *gradient methods* have been proposed which can be seen as a particular case of subgradient methods and share the same underlying idea of locally computing a gradient [60, 107]. In [107] moreover, the problem of coordinating the step-sizes is overcome by the use of a consensus scheme which allows the step-sizes to be assigned independently to each agent.

A different class of algorithms called *Newton-based methods* exploits the smoothness of the local cost functions in order to choose the descent direction via the second order derivative (the Hessian) of the local cost functions. The works [104, 105] propose a possible implementation of Newton methods over time-varying graphs, but require a diminishing step-size to ensure the convergence. An alternative approach has been introduced in [111, 112] that employs an average consensus scheme in order to substitute the diminishing step-size with a constant one. This framework has been extended in [35] to solve non-convex optimisation problems as well. The recent [8] builds upon this framework in order to design an asynchronous algorithms which is also robust to packet losses. As a by-product, the authors are able to design a gradient and Jacobi methods characterised by the substitution of the Hessian computation with simpler descent direction choices.

Gradient descent methods have the issue of inexact convergence, that is, the algorithm is guaranteed to converge only to a neighbourhood of the optimal solution and similarly to subgradient methods, requires a decreasing step-size to do so. In order to solve this problem the EXTRA has been proposed in [92], which generalises the gradient descent by adding a cumulative correction term, ensuring exact convergence with a fixed step-size.
The ESOM algorithm [69] is instead based on the method of multipliers [10], which it modifies by performing an inexact update based on the second-order expansion of the augmented

Lagrangian of the problem. This method, however, is shown to converge only in the case of strongly convex local costs.
In [80] the sparsity of linear regression problems is exploited to design two algorithms for parallel computing. In particular, the first is an extension of the FISTA (fast iterative shrinkage algorithm) [5] to the distributed set-up of interest, while the second, named *GRock* merges a gradient descent method with a greedy descent direction choice criterion. The main issue with these two proposed algorithms is that a central co-ordinator is necessary to collate the results computed by each agent and transmit it back to the nodes, therefore requiring a semi-decentralised architecture.
In [63] an asynchronous gossip-based algorithm is introduced, which is characterised by the fact that each local variable is constrained to belong to the intersection of some convex sets and at each update a projection is performed on one of these sets randomly chosen. The authors show that exact convergence can be achieved by means of a diminishing step-size, while inexact convergence is still achieved even if a constant step-size is employed.

Partition-based distributed optimisation The partition-based optimization framework has been first introduced in [39] which presents a modified version of the Alternating Direction Method of Multipliers to solve cooperative localisation in WSN, while in [71] the partition-based ADMM is applied to solve an MPC problem. Other algorithms have been proposed to solve partition-based problems, for example the Block-Jacobi algorithm of [100] for the solution of quadratic programming problems, or the dual decomposition algorithm of [13]. Finally, a generalised gradient algorithm has been described in [99]. Notice that the works [100, 99] prove convergence in the case of lossy communications.
The author of this Thesis together with his advisor and colleagues have recently submitted a paper describing a robust version of the partition-based ADMM [1].

1.2.2 Operator Theory and splitting operators

Some distributed optimisation algorithms, the ADMM being among them, can be formulated making use of non-expansive operator theory. In particular the rationale is to recast a convex optimisation problem into the problem of finding the fixed points of one such operator, for which a large choice of algorithms is available. This Section serves as a brief overview of the literature on non-expansive operator and fixed point algorithms, many of which will be analysed in some detail in the following Chapters.

First of all, a very thorough overview of operator theory and its applications to convex optimisation can be found in the book [3], while [4] introduces some further characterisations of non-expansive operators. A particular class of non-expansive operators are the proximal operators and the closely related class of reflective operators. The properties of these operators are described in [3, 76], and [27, 24] which collect a large number of examples as well.

The prototypical fixed point algorithm is the *Banach-Picard iteration* [3] which consists of a simple iteration of a non-expansive operator, but is guaranteed to converge only if stronger assumptions are done on the properties of the operator. Proximal operators and the Banach-Picard iteration are connected by the *proximal point algorithm*, described first in [89] and analysed further for instance in [87, 76, 30]. Proximal point algorithms find application in a wide range of fields, from Statistics and Machine Learning [83, 76] to Signal Processing [27, 24].
Proximal point algorithms however might be unwieldy when dealing with complex functions to be minimised, which entail a complex evaluation in order to compute the corresponding proximal operator. However in the applications of Machine Learning and Signal Processing mentioned above, the objective functions of the problem that needs solving are often the

sum of two less complex functions (see Section 1.1 for an example). Therefore the class of *splitting operators* has been developed in order to design fixed point algorithms that leverage the separability of the objective function to break down the computational burden in more manageable subproblems. Possibly the most well-known splitting algorithm is the Douglas-Rachford splitting (DRS), introduced in [36], and the closely related Peaceman-Rachford splitting, first described in [78]. These algorithms hinge on the Peaceman-Rachford operator[1] which is the composition of two proximal operators, each dependent on one part of the objective function, and therefore enjoy of the convergence properties guaranteed by the use of the proximal operator. The Douglas-Rachford splitting has been shown to be a particular case of the proximal point algorithm in [38] and, more importantly, the ADMM on which this Thesis focuses can be derived applying the DRS on the Lagrange dual of the problem at hand [64].

Another example of splitting algorithm is the Forward-Backward splitting (FBS) which [3] discusses in detail, see also [27], and [49] for a tutorial with details on the practical implementation. The FBS has been designed as a more general version of the gradient method, and indeed requires that at least one of the functions in the problem be smooth. The FBS has been successfully applied to Signal Processing [27], Image Processing [85] and Automatic Control [45], to name a few. Distributed optimisation applications of the FBS has been described in [108] and [45].

The splitting schemes mentioned above can generalised and enhanced using the *Krasnosel'skiĭ-Mann iteration* (KM), introduced in [65] and [62], which is characterised by the iterated application of an operator derived by the convex combination of a non-expansive operator and the identity operator. The KM iteration is guaranteed to converge to the fixed points of the non-expansive operator to which it is applied [3], also in the presence of randomised co-ordinate selection [57, 7] and errors [25]. Therefore the KM iteration is of particular interest as it entails stronger robustness properties than the DRS and FBS alone. Furthermore, it is possible to combine the KM iterate with an *inertia* scheme in order to result in an acceleration of the algorithm [59].
Another splitting scheme that has been proposed is the *primal-dual splitting* (PDS) [28, 101], designed to solve the minimisation problem of the sum of three convex functions, one of which must be smooth. The algorithm presents a structure similar to the DRS, with two proximal updates and a linear update, but also to the FBS since one of the proximal updates is performed in conjunction with a gradient descent step. In the recent [106] the PDS has been shown to converge almost surely when only a randomly drawn subset of co-ordinates is updated at each iteration. Finally, [91] applies the PDS to the distributed consensus optimisation problem that the ADMM was designed to solve and obtains the so-called *primal-dual method of multipliers* (PDMM), which in a particular case can be shown to coincide with the simpler form of ADMM.

1.3 Alternating Direction Method of Multipliers

The Alternating Direction Method of Multipliers (ADMM) was first introduced by Glowinski and Marrocco in [48], and Gabay and Mercier in [42], but has only recently seen a surge in popularity due to its suitability for Big Data and distributed computation applications. For an historical perspective on the algorithm see [47] by Glowinski. For a tutorial about the ADMM with some applications see [10] and [37].

[1] Sometimes referred to as Douglas-Rachford opearator.

For better reference in the present Section, the problem that the ADMM has been designed to solve is the following

$$\begin{aligned} &\min_{x,y}\{f(x)+g(y)\} \\ &\text{s.t. } Ax+By=c \end{aligned} \tag{1.1}$$

where the functions f and g are convex. Defining the *augmented Lagrangian* of the problem as

$$\mathcal{L}(x,y;w)=f(x)+g(y)-w^\top(Ax+By-c)+\frac{\rho}{2}\|Ax+By-c\|^2$$

with w the vector of Lagrange multipliers, the ADMM is described by the following three update equations

$$\begin{aligned} y(k+1) &= \arg\min_y \mathcal{L}(x(k),y;w(k)) \\ w(k+1) &= w(k)-\rho\left(Ax(k)+By(k+1)-c\right) \\ x(k+1) &= \arg\min_x \mathcal{L}\left(x,y(k+1);w(k+1)\right). \end{aligned}$$

1.3.1 Asynchronous distributed ADMM

Results about the convergence of the ADMM in scenarios prone to packet loss have been first proposed in the paper [2] co-authored by this Thesis' author. On the other hand, many works have been devoted to the study of the ADMM under asynchronous communications with different assumptions and restrictions, and they will be reviewed in the present Section.

The first results on the convergence of an asynchronous version of ADMM that have been proposed are [57] and [103].
In the former, the ADMM is shown to converge if at each instant one single node performs an update. However, this result has been subsequently generalised to allow the update of a randomly selected subset of nodes at each time instant in [7]. The main idea underlying the results of [57, 7] is to prove the convergence of a stochastic version of the Krasnosel'skiĭ-Mann iteration in which randomly selected co-ordinates are updated, and then showing that the asynchronous ADMM indeed conforms to this update scheme.
In [103] the authors introduce a reformulation of the ADMM in which at each instant a randomly selected subset of edges is activated, and only the nodes connected by the activated edges perform an update. Moreover, the convergence is shown to have a $O(1/k)$ rate. The drawback of this update scheme is that the agents need to agree on a global clock in order to be able to perform an update at the time of activation of an edge.

A more recent result is the *ARock* framework described in [79] which has been designed as a general purpose parallel and decentralised optimisation algorithm, that in some cases requires a central memory storage. The ARock can be particularised to define an asynchronous ADMM in which a single randomly selected node performs an update at each time instant and then transmits to each of its neighbours the new information it has computed. The authors prove the convergence of the ARock algorithm in a *partially asynchronous* scenario in which, that is, a maximum delay between activations of a node is allowed. The authors of [52] however are able to relax this constraint proving convergence with unbounded maximum delays, *i.e. totally asynchronous* updates.

The asynchronous ADMM methods described above are characterised by a fully distributed architecture, in that the agents have all equal status and the solution is reached by means of exchanges of locally computed updates between neighbours. Many works have instead studied the convergence of the ADMM with a star topology for the communication graph and asynchronous local updates, in which a central node – called *master* – has the task

of collating the updates performed with locally available information by all the other nodes – referred to as *slaves*. Algorithms designed with this particular graph architecture are particularly appealing in parallel computing applications, but fall short in other such as Smart Grids, distributed Machine Learning, WSN.
A first example of master-slave asynchronous ADMM is introduced in the paper [55]. In this case, at each instant a random subset of slaves is activated and each computes a gradient on the base of a local cost function, deputising the master to performing the necessary updates based on the local gradients. This algorithm has two main advantages, the first being that it is shown to converge even in the presence of non-convex local costs, and the second being that it does not require a global clock to be available.
The paper [116] on the other hand requires a global clock to reach convergence, but is characterised by a more complete distribution of the update and computational tasks to the slaves. In particular each of the slaves stores and updates a local variable on the base of information available to itself only, and the role of the master is to aggregate these updates once a minimum number of them is received and broadcast the result.
Finally, [16] presents a partially asynchronous ADMM on a star communication graph that reaches convergence even though the local functions are non-convex (but smooth). In order to do so, the authors show that the parameters of the ADMM need to be chosen as functions of the maximum delay allowed.

1.3.2 Linear convergence of the ADMM

The ADMM has first been shown to converge with a rate of $O(1/k)$ in the general case of convex non-smooth cost functions in [53] and more recently in the comprehensive [31], which also proves that the ADMM for one-dimensional problems converges with a rate of $O(1/k^2)$. Linear convergence on the other hand has been shown for different applications and assumptions, which are reviewed below, concluding with the results available for distributed and asynchronous versions of the ADMM. As in the case of [31], many of the works that will be mentioned make use of the relationship between the DRS and the ADMM in order to derive the convergence results.

Among the first results to be presented is [9] in which local linear convergence is proved for a class of quadratic and linear programming problems using spectral analysis on a matrix that describes the evolution of the iterates of the ADMM, suitably manipulated. Another local result is proved in [81] in case the variables of the algorithm are constrained to belong to closed sets, with a global extension provable if the sets are also convex.
Focused on quadratic programming problems as well is [84] in which the linear convergence is derived by the imposition of upper and lower bounds on the value of the variables in the algorithm.

The work [77] presents an original approach to prove linear convergence of the DRS for convex non-smooth optimisation, based on the introduction of the so-called *Douglas-Rachford envelope*, which is a continuously differentiable function with stationary points the minimisers of the problem at hand. Moreover, the authors are able to extend the results to a class of variants of the ADMM characterised by the substitution of matrix-weighted norms in the definition of the augmented Lagrangian.
The authors of [75] prove linear convergence in the case of smooth cost functions with the use of dynamical systems theory. In particular they show that the convergence of the ADMM is linked to the stability of a suitably defined dynamical system, which provides also a more general framework for convergence proofs.

A more recent and very comprehensive study has been presented in [34] that analyses the global linear convergence of the ADMM under a large choice of assumptions on the cost functions and the matrices that define the linear constraints of the problem. In particular

linear convergence is proved for different combinations of strong convexity and Lipschitz continuous derivative of (one of) the objective functions, and full row rank of the matrices in the constraints. Moreover the results are proved to hold also for a generalised version of the ADMM that, similarly to [77], substitutes 2-norms with matrix-weighted norms in the definition of the iterates.
A companion paper to the work [31] mentioned above is [32] in which the authors study extensively the linear convergence of the DRS, PRS and ADMM under different regularity conditions such as strong convexity, Lipschitz continuous derivative and bounded linear regularity of the cost functions.
To conclude, [46] proves global linear convergence of the ADMM under assumptions of strong convexity and smoothness of the objective functions, presenting a set of convergence rate bounds that are shown to be tight. Moreover, the authors describe a heuristic for the choice of the step-size and of the metric selection matrices (see below for an explanation) in order to provide faster rates of convergence.

The linear convergence of the distributed ADMM has first been proved in [93] for a *node-based* implementation of the algorithm, that is, in which each node updates a fixed set of local variables, whose number does not depend on the network topology (see Section 3.3 for more details). The authors restrict the analysis to the case of strongly convex local cost functions and present a formula for the theoretical rate that depends on the communication graph topology, on the properties of the local functions and the ADMM parameters. Therefore the results can be used to design speed-up criteria for the algorithm.
The more recent [58] proves the linear convergence of the ADMM in its most classic formulation with the assumption that the local costs are strongly convex in a neighbourhood of the optimum of the problem. The authors propose a reformulation of the ADMM as a linear update equation perturbed by a term that tends to zero super-linearly in a neighbourhood of the solution, therefore showing linear convergence as soon as the algorithm enters the neighbourhood.
Finally, [19] studies the linear convergence of the master-slave ADMM introduced in [16] in a partially asynchronous set-up with strongly convex cost functions with Lipschitz continuous derivatives. In particular the rates are explicitly described as a function of the ADMM parameters, the maximum delay and the number of agents in the network.

1.3.3 Variants of the ADMM

Many variants of the ADMM have been proposed with the aims of obtaining faster convergence, reducing the computational costs or tailoring the algorithm for particular classes of problems. This Section presents a brief and not exhaustive review of works on the most common variants of the ADMM, both in a centralised and distributed set-up.

Multi-block ADMM A natural extension of the ADMM is to solve problems that are the sum of three or more convex functions each in a different variable, with linear constraints, that is modify problem (1.1) to

$$\begin{aligned} &\min_{x,y,z}\{f(x)+g(y)+h(z)\} \\ &\quad \text{s.t. } Ax+By+Cz=d. \end{aligned} \tag{1.2}$$

This class of problems is of particular interest in Image Processing applications.
Differently from the classic ADMM, the convergence of the three-block ADMM is not guaranteed in the case of convex cost functions, as [20] proved only recently, and indeed it is necessary to provide stronger assumptions on the problem (1.2). Among the first results to be proposed is [51] which shows convergence in case all three objective functions are

strongly convex, while [22] relaxes this condition by allowing one of the functions to be simply convex.
In order to ensure convergence of the three-block ADMM with only convex functions – as well as the further extension of the multi-block ADMM – works [97, 21] design a *random permutation* scheme for performing the variables' updates. In particular, the proposed algorithm randomly selects at each iteration the order in which to perform the update of the three – or m – variables.
More recent results are [33] and [56]. In particular, [33] presents a convergence result for a splitting algorithm designed to solve the monotone inclusion problem of three operators. The three- and multi-block ADMM follow as particular cases and can be shown to converge with a rate of $O(1/k)$ if at least one of the cost functions is strongly convex. On the other hand [56] proves that the convergence of the multi-block ADMM is linear for strongly convex problems, and to do so the authors show that for a sufficiently small step-size the primal and dual optimality gaps decrease after each iteration of the proposed ADMM.

Acceleration schemes Different acceleration schemes have been proposed in order to speed up the convergence of the ADMM, which employ different techniques. In the following two possible approaches will be discussed: adaptive parameters selection and the metric selection or preconditioning.

The classic ADMM is characterised by a penalty parameter that weights the violation of the constraints at each iteration (the ρ parameter in the augmented Lagrangian above), and the algorithm is shown to converge for any positive value of this penalty. However in [54] the authors propose an adaptive scheme for the choice of the penalty at each iteration on the basis of the current reached value of the cost function, and prove that the ADMM endowed with this choice criterion converges. See also [10] for a discussion of this variant of ADMM. More recently this idea has been extended to the decentralised ADMM in [96] with a modified penalty selection criterion that bases the choice on the network topology.
The same rationale behind the adaptive ADMM described above is applied in [109] and [110] to the relaxed ADMM for both the tunable parameters that characterise the algorithm in a distributed set-up.

The convergence rate of the ADMM depends on the conditioning of the problem data, that is, on the matrices that appear in the linear constraints of the problem, in particular on their maximum and minimum eigenvalues. Therefore preconditioning the data by multiplying it by an invertible matrix can lead to better performance of the algorithm in terms of number of iterations required.
The authors of [44, 46] propose a preconditioning criterion for a class of problems for which the ADMM exhibits linear convergence, which is based on the explicit derivation of the convergence rate as a function of the eigenvalues of the preconditioned data matrices. Moreover, they are able to provide an heuristic for the selection of the conditioning matrices even in the case some of the assumptions are violated. The same idea is explored also in [43] in the context of quadratic problems.
Other algorithms exploiting metric selection are presented in [45] for the FBS, in [12] for the Uzawa splitting algorithm in the case of saddle point problems, and in [82] for primal-dual algorithms.
Notice that this techniques are also referred to as metric selection since preconditioning the data corresponds to requiring that the variables belong to a different space than the original.

Miscellaneous algorithms The classic ADMM needs to solve two optimisation problems at each iteration in order to update the primal variables but, depending on the problem at hand, this can be a computationally intensive task. In particular in distributed scenarios

this can lead to infeasibly long convergence times as the local agents struggle to solve very complex problems with limited capabilities. Therefore the class of *inexact ADMM* algorithms has been proposed, first in [17] and then [18], in which each agent is not required to solve precisely the local problem. Instead, each update is performed via a step of the proximal gradient method exploiting the smoothness of the cost function, resulting in simpler local update steps that guarantee convergence nonetheless. An example of inexact ADMM has recently been applied in [23] to the problem of Poisson image deblurring with promising results.

On the other hand, if the system solving the optimisation does not have stringent computational constraints, a possible class of algorithms are the *bi-alternating direction method of multipliers* (bi-ADMM), introduced in [113] and later in [114]. The idea is to incorporate in the augmented Lagrangian both informations on the cost functions and the corresponding dual functions, in order to perform at each iteration a more effective update. This class of ADMM variants has been shown in [113] to converge with a rate that is $O(1/k)$.

An idea similar to that of the bi-ADMM is to substitute the 2-norm term in the augmented Lagrangian with matrix-weighted norms and additional quadratic terms [34], or with a Bregman divergence term [102]. The Bregman divergence is particularly versatile as it can be tailored to the problem at hand in order to obtain faster and more efficient updates. Moreover, in [102] the authors show that the proposed algorithm defines a general framework that encompasses different variants of the ADMM such as the inexact ADMM discussed above, the Bethe-ADMM [40] and the generalised ADMM of [34].

1.4 Contributions and Outline of the Thesis

This Thesis presents for the first time an analysis of the convergence of the ADMM and the more general relaxed version in a distributed set-up in the presence of possibly faulty communications between the agents collaborating to the solution of a convex optimisation problem. In particular the convergence of the proposed robust ADMM is proved with the use of operator theoretical results, also in the case of asynchronous update criteria or more general co-ordinate selection schemes.
Moreover, the convergence rate of the proposed algorithm is shown to be linear in case the cost functions defining the problem are strongly convex and a theoretical upper bound to this rate is given as the maximum eigenvalue of a matrix depending on the ADMM parameters and the packet loss and update probabilities.
Finally, the convergence results are extended to the partition-based framework in which local cost functions depend not only on the local variable but also on the variables of the agent's neighbours.

All the results are validated by extensive Monte Carlo simulations presenting the performance of the ADMM as a function of its parameters and the packet loss and update probabilities. In particular, the proposed algorithm is applied to quadratic problems, for which an analysis of the convergence rate is presented as well, and to the Lasso problem.

A partial selection of the results presented in this Thesis appears in [2] and in the submitted [1].

The Thesis is organised as follows. Chapter 2 presents a review of the ADMM and the more general relaxed ADMM, as well as the necessary operator theory notions used throughout the Thesis. In Chapter 3 the ADMM is applied to distributed convex optimisation problems and analysed in detail. Chapter 4 introduces the randomised ADMM and proves convergence in the presence of packet losses and asynchronous updates. Moreover, the linear convergence is proved in the case of strongly convex cost functions. Chapter 5 extends the results of the previous one to the partitioned-based framework, while Chapter 6 describes

the simulations results testing the proposed algorithms. Chapter 7 concludes the work and describes some further lines of study.

At the end of the Thesis, Appendix A presents a review of the necessary mathematical background and of some further theoretical results omitted from the main text. Appendix B presents and discusses the convergence of the stochastic Krasnosel'skiĭ-Mann iteration at the base of the algorithms proposed in this Thesis. Appendix C presents a simple security analysis of the proposed algorithm for the quadratic case.

Chapter 2

Alternating Direction Method of Multipliers

The present Chapter serves as a background on the Alternating Direction Method of Multipliers in the deterministic formulation and as an introduction to operator theory with a brief overview of useful results.

2.1 Alternating Direction Method of Multipliers

This Section introduces the Alternating Direction Method of Multipliers (hereafter, ADMM), popularised by the work of Stephen Boyd [10] and of widespread use. As the next Sections will show, however, this formulation is actually a particular case of the general *relaxed ADMM*, so the present Section serves also as a bridge to understand the link between the two formulations.

Consider the following optimisation problem, which represents the most general formulation usually associated with the ADMM,

$$\begin{aligned} \min_{x\in\mathcal{X},\ y\in\mathcal{Y}}\ & \{f(x)+g(y)\} \\ \text{s.t. } & Ax+By=c \end{aligned} \tag{2.1}$$

where $\mathcal{X}$ and $\mathcal{Y}$ are Hilbert spaces, $f:\mathcal{X}\to\mathbb{R}\cup\{+\infty\}$ and $g:\mathcal{X}\to\mathbb{R}\cup\{+\infty\}$ are closed, proper and convex functions[1]. Hereafter it is assumed that problem (2.1) has at least one solution.
In order to define the ADMM it is necessary to introduce the so-called *augmented Lagrangian* of the problem, which is

$$\mathcal{L}(x,y;w)=f(x)+g(y)-w^{\top}(Ax+By-c)+\frac{\rho}{2}\left\|Ax+By-c\right\|^2 \tag{2.2}$$

where $\rho>0$ is a free parameter and w represents the vector of Lagrange multipliers. Note that (2.2) differs from the commonly used Lagrangian (see Appendix A.1) for the addition of the *penalty term* $(\rho/2)\left\|Ax+By-c\right\|^2$. The purpose of the penalty term is to ensure that the optimisation algorithm is, at all times, close to satisfying the linear constraint $Ax+By=c$. Finally, the ADMM is defined by the iteration of the following three equations

$$y(k+1)=\arg\min_{y}\mathcal{L}(x(k),y;w(k)) \tag{2.3}$$

$$w(k+1)=w(k)-\rho\left(Ax(k)+By(k+1)-c\right) \tag{2.4}$$

$$x(k+1)=\arg\min_{x}\mathcal{L}\left(x,y(k+1);w(k+1)\right) \tag{2.5}$$

[1] See Appendix A.1 for a review of the necessary mathematical background.

that consist of the two optimisation problems (2.3) and (2.5) for the *primal variables* y and x, respectively, and a simple update for the Lagrange multipliers w or *dual variables*. Notice that this formulation is equivalent to that presented in [10], but for a change in the order of the update equations.
The ADMM is provably shown to converge to the optimum of problem (2.1) for any $\rho > 0$. See for instance the convergence proof of [10], which requires that the Lagrangian of the problem (*i.e.* the augmented Lagrangian with $\rho = 0$) have a saddle point. Other, less restrictive proofs are presented in [41, 38].

As described above, the formulation (2.3)–(2.5) of the ADMM can be interpreted as an augmented Lagrangian method. However the algorithm naturally arises from the application of the Douglas-Rachford splitting operator to the dual of problem (2.1). In order to show this result, the following Section introduces some necessary background on splitting methods and operator theory.

2.2 Splitting Operators

The interest in operator theory in the context of convex optimisation stems from the fact that such problems can be cast into the problem of finding the fixed points of suitable non-expansive operators. This Section introduces therefore some background on operator theory and the mathematical tools necessary to solve fixed points problems (while more details can be found in the Appendix A.2).
Most of the theory presented hereafter is discussed very thoroughly in the book by Bauschke and Combettes [3], therefore this reference will generally be omitted.

2.2.1 Operator Theory

Let $\mathcal{X}$ and $\mathcal{Y}$ be non-empty sets, an *operator* or *mapping* T maps every point $x \in \mathcal{X}$ to a point $Tx \in \mathcal{Y}$. Throughout this Section the two sets $\mathcal{X}$ and $\mathcal{Y}$ are assumed to be real Hilbert spaces equipped with the inner product $\langle \cdot, \cdot \rangle$ and the associated norm $\|\cdot\|$[2].

Definition 2.1 (Fixed points of an operator)**.** Let $T : \mathcal{X} \to \mathcal{X}$ be an operator on the Hilbert space $\mathcal{X}$, then the set of *fixed points* of T is defined as the set

$$\mathrm{fix}(T) = \left\{ x \in \mathcal{X} \;\middle|\; Tx = x \right\}.$$

The following particular class of operators is important from a convex optimisation perspective as it enjoys good convergence properties. Here "convergence properties" means that it is possible to develop efficient algorithms for finding the fixed points of these operators.

Definition 2.2 (Non-expansive operators)**.** Let $\mathcal{X}$ be a Hilbert space, an operator $T : \mathcal{X} \to \mathcal{X}$ is said to be *non-expansive* if it has unitary Lipschitz constant, that is it verifies

$$\|Tx - Ty\| \leq \|x - y\|$$

for any two $x, y \in \mathcal{X}$.

The following class of operators is characterised by even more restrictive convergence properties.

[2]Note that in this Section the Hilbert spaces might in general be infinite dimensional. However in the following the attention will be restricted to finite dimensional spaces.

Definition 2.3 (Firmly non-expansive operators)**.** Let $\mathcal{X}$ be a Hilbert space, an operator $T : \mathcal{X} \to \mathcal{X}$ is said to be *firmly non-expansive* if for any $x, y \in \mathcal{X}$ it satisfies

$$\|Tx - Ty\|^2 + \|(I - T)x - (I - T)y\|^2 \leq \|x - y\|^2$$

where I is the identity operator on the Hilbert space $\mathcal{X}$.

Clearly firm non-expansiveness implies non-expansiveness, and moreover the following result holds.

Proposition 2.4 ([3, Proposition 4.2])**.** *Let $\mathcal{X}$ be a Hilbert space and let $T : \mathcal{X} \to \mathcal{X}$. Then the following are equivalent*

(i). T is firmly non-expansive,

(ii). $I - T$ is firmly non-expansive,

(iii). $2T - I$ is non-expansive,

(iv). $\|Tx - Ty\|^2 \leq \langle x - y, Tx - Ty\rangle$ for any $x, y \in \mathcal{X}$.

The following class of operators will be instrumental in developing fixed point algorithms.

Definition 2.5 (Averaged operators)**.** Let $\mathcal{X}$ be a Hilbert space, let the operator $T : \mathcal{X} \to \mathcal{X}$ be non-expansive and let $\alpha \in (0, 1]$. Then the operator $T_\alpha : \mathcal{X} \to \mathcal{X}$ defined as $T_\alpha = (1 - \alpha)I + \alpha T$ is said to be *averaged* or *α-averaged*.

Notice that the α-averaging of an operator is sometimes referred to as *relaxation*.

Indeed averaged operators satisfy the following property, which can be easily verified using the definition of fixed point.

Proposition 2.6. *Let $T : \mathcal{X} \to \mathcal{X}$ be a non-expansive operator on the Hilbert space $\mathcal{X}$ and $T_\alpha : \mathcal{X} \to \mathcal{X}$ be the corresponding averaged operator $T_\alpha = (1 - \alpha)I + \alpha T$. Then it is* $\mathrm{fix}(T) = \mathrm{fix}(T_\alpha)$.

Remark 2.7. Note that a non-expansive operator is averaged with $\alpha = 1$, while an operator is firmly non-expansive if and only if it is $1/2$-averaged.

Figure 2.1 is a graphical representation of the action performed by α-averaged and, for the remark above, non-expansive and firmly non-expansive operators in $\mathbb{R}^2$. In particular the areas shaded in grey represent the region of $\mathbb{R}^2$ in which $Tx - Ty$ can be located, for any two $x, y \in \mathbb{R}^2$.

The following property will be useful in the following.

Proposition 2.8 ([4, Lemma 4.10])**.** *The composition of two non-expansive operators is a non-expansive operator itself.*

To conclude this brief overview the following two examples of operators are presented.

Definition 2.9 (Proximal and reflective operators)**.** Let $\mathcal{X}$ be a Hilbert space, let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be a closed, proper, convex function and let $\rho > 0$. The *proximal operator*, or *proximity operator*, of f with penalty ρ is the operator $\mathrm{prox}_{\rho f} : \mathcal{X} \to \mathcal{X}$ defined as

$$\mathrm{prox}_{\rho f}(x) = \underset{y \in \mathcal{X}}{\arg\min} \left\{ f(y) + \frac{1}{2\rho} \|x - y\|^2 \right\}. \tag{2.6}$$

Moreover, the *reflective operator* of f is defined as

$$\mathrm{refl}_{\rho f} = 2\,\mathrm{prox}_{\rho f} - I. \tag{2.7}$$

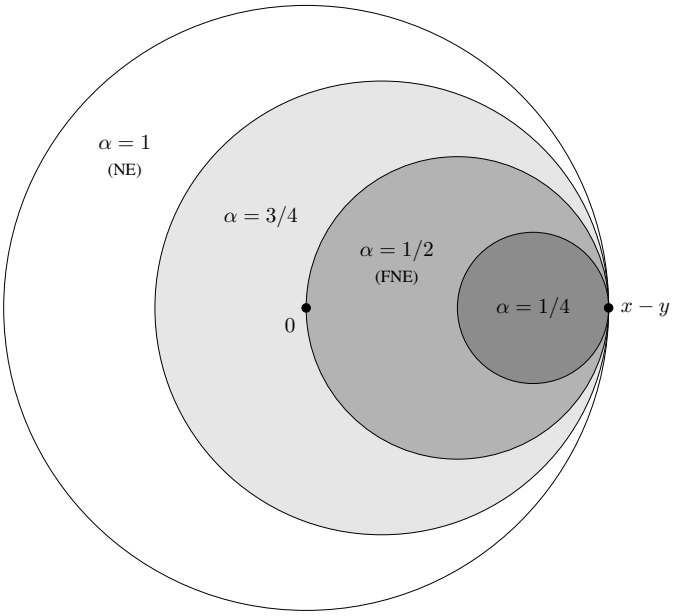


FIGURE 2.1: Representation of the action performed by α-averaged operators.

Remark 2.10. The proximal operator is firmly non-expansive and the reflective operator is non-expansive. The first result is proved in [3, Proposition 12.27] and the second is a consequence of Proposition 2.4.

Remark 2.11. Notice that the proximal operator is unique at any point $x \in \mathcal{X}$ since it is the minimum of a strongly convex function, by the fact that f is convex and the squared norm is strongly convex. Moreover, since the function f is assumed to be proper, then the domain of the proximal is the whole space $\mathcal{X}$, because the solution to the minimisation problem is always well defined.

Figure 2.2 depicts the relationship between the proximal and reflective operators of a function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ with penalty parameter ρ. Clearly the name of reflective operator stems from the fact that $\mathrm{refl}_{\rho f}(x)$ is the reflection of x with respect to the symmetry axis passing through $\mathrm{prox}_{\rho f}(x)$ and orthogonal to the vector $\mathrm{prox}_{\rho f}(x)$.

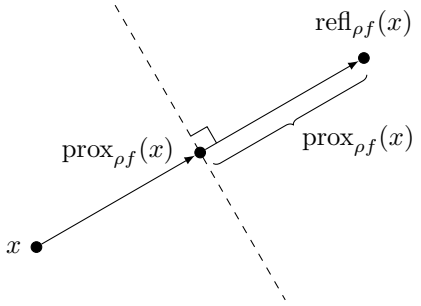


FIGURE 2.2: Representation of the relationship between proximal and reflective operators.

Given the important role that the proximal operator will play in the following, it is interesting to analyse the effect that it has. The proximal operator has two important characteristics:

- it pushes points that are outside the domain of f onto points that are on the boundary of $\mathrm{dom}(f)$. This can be seen for example if the proximal is applied to the indicator function of the norm-2 ball $\mathcal{C} = \{x \in \mathbb{R}^2 \mid \|x\|_2 \leq 1\}$. In this case the proximal becomes the *projection operator*

$$\mathrm{P}_2(x) = \begin{cases} x & \|x\|_2 \leq 1 \\ \frac{x}{\|x\|_2} & \text{otherwise.} \end{cases} \tag{2.8}$$

  and it is depicted in Figure 2.3 (left).

- Moreover, it draws the points which it is applied to closer to the minimum of f. Consider for example the simple quadratic cost $f(x) = ax^2$, it is easily verified that $\mathrm{prox}_{\rho f}(x) = x/(2\rho + 1)$, which indeed draws x closer to the origin each time it is applied. Figure 2.3 (right) depicts the repeated application of the proximal operator for the quadratic function. The extent by which the point is drawn closer to the minimum depends on the penalty ρ, with larger values that correspond to points mapped closer, which in this example is given by prox_f depending on the inverse of ρ.

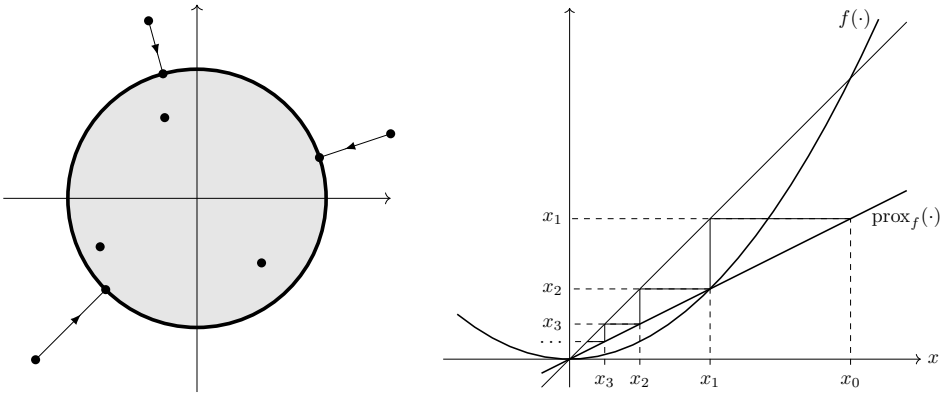


FIGURE 2.3: Depiction of the two-fold effect of the proximal operator.

2.2.2 Fixed point algorithms

As mentioned above, often it is required to find the fixed points of a non-expansive operator. Different algorithms are available to perform this task, and in the following two of them will be briefly reviewed, the Banach-Picard iteration and the Krasnosel'skiĭ-Mann iteration.

Before describing the two fixed-point algorithms the following definitions are introduced.

Definition 2.12 (Weak and strong convergence)**.** Let $\{x(k)\}_{k\in\mathbb{N}}$ be a sequence of points in the Hilbert space $\mathcal{X}$. The sequence is said to *converge weakly* to a point $\bar{x} \in \mathcal{X}$ if

$$\langle x(k), y\rangle \to \langle \bar{x}, y\rangle \; \forall y \in \mathcal{X}$$

for k that tends to infinity. Weak convergence will be denoted as $x(k) \rightharpoonup \bar{x}$.
Moreover, the sequence is said to *converge strongly* if

$$\|x(k) - \bar{x}\| \to 0$$

for $k \to \infty$, and this fact will be denoted with $x(k) \to \bar{x}$.

The simplest algorithm that can be applied to find the fixed points of an operator is the *Banach-Picard iteration* [3, Theorem 1.48], defined as

$$x(k+1) = Tx(k). \tag{2.9}$$

The sequence $\{x(k)\}_{k\in\mathbb{N}}$ generated by this algorithm can be proved to converge weakly to a fixed point of the non-expansive operator T if the *asymptotic regularity* property $x(k) - Tx(k) \to 0$ holds [3, Theorem 5.13]. Moreover, if the operator is firmly non-expansive then the weak convergence to a point in $\mathrm{fix}(T)$ is guaranteed [3, Example 5.17].
However if this is not the case, the algorithm might not converge. Therefore the Krasnosel'skiĭ-Mann iteration [65, 62] is introduced, which enjoys more robust convergence properties that impose only the non-expansiveness on the operator.

Theorem 2.13 (Krasnosel'skiĭ-Mann iteration [3, Theorem 5.14])**.** *Let $\mathcal{D}$ be a non-empty, closed, convex subset of the Hilbert space $\mathcal{X}$, let $T : \mathcal{D} \to \mathcal{X}$ be a non-expansive operator such that $\mathrm{fix}(T) \neq \emptyset$. Moreover, let $\{\alpha_k\}_{k\in\mathbb{N}}$ be a sequence in $[0, 1]$ such that*

$$\sum_{k\in\mathbb{N}} \alpha_k(1 - \alpha_k) = +\infty. \tag{2.10}$$

The Krasnosel'skiĭ-Mann iteration is defined as

$$x(k+1) = (1 - \alpha_k)x(k) + \alpha_k Tx(k) \tag{2.11}$$

with initial condition $x(0) \in \mathcal{D}$.
The following results hold for the sequence of points generated by the Krasnosel'skiĭ-Mann iteration.

(i). $\{Tx(k) - x(k)\}_{k\in\mathbb{N}}$ *converges strongly to 0*[3]*,*

(ii). $\{x(k)\}_{k\in\mathbb{N}}$ *converges weakly to a point in* $\mathrm{fix}(T)$.

This result therefore ensures that the Krasnosel'skiĭ-Mann iteration of a non-expansive operator converges (weakly) to a fixed point of said operator, given a suitable sequence of *step-sizes* $\{\alpha_k\}_{k\in\mathbb{N}}$.

Remark 2.14. The Krasnosel'skiĭ-Mann iteration (2.11) at each time instant k essentially evaluates the α_k-averaged operator of T in the current state $x(k)$.

Remark 2.15. Note that the convergence result is satisfied in particular by a sequence of step-sizes $\alpha_k = \alpha$ for a constant α. In this case the Krasnosel'skiĭ-Mann iteration of operator T coincides with the Banach-Picard iteration of the α-averaged operator T_α.

2.2.3 Application to convex optimisation

The two fixed point algorithms described above find a natural application in the context of convex optimisation, giving rise to the class of *proximal algorithms*.

The first proximal algorithm to be introduced was the *Proximal Point Algorithm* (PPA) [89], which establishes the framework in which many fixed point algorithms can be defined, such as the Douglas-Rachford splitting.
Let $\mathcal{X}$ be a Hilbert space, let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be a closed, proper and convex function, and consider the following convex optimisation problem

$$\min_{x\in\mathcal{X}} \; f(x). \tag{2.12}$$

[3] Which implies that the asymptotic regularity condition mentioned above is satisfied.

The PPA algorithm is then defined by the following iteration

$$x(k+1) = \text{prox}_{\rho f}\left(x(k)\right) \tag{2.13}$$

for some parameter $\rho > 0$. Notice that this algorithm corresponds to the Banach-Picard iteration (2.9) applied to the proximal operator of f, and since the proximal operator is firmly non-expansive (see Remark 2.10), the weak convergence to a fixed point is ensured.
Moreover, since $x^* \in \mathcal{X}$ is a fixed point of the proximal operator if and only if it is a minimiser of the function f[4], the convergence to the solution of problem (2.12) is guaranteed.
The main issue with the PPA is that at each iteration it requires to compute the proximal operator of f, which might be quite difficult and time consuming, depending on the structure of the function.

Many convex optimisation problems, however, have a structure amenable to the implementation of more efficient algorithms. Indeed, in particular in Machine Learning applications [80], the problems that need solving are characterised by an objective function that can be split into the sum of two terms

$$\min_{x\in\mathcal{X}} \{f(x) + g(x)\} \tag{2.14}$$

with f and g closed, proper and convex. A first approach to finding the solution might be to simply apply the PPA to the function $F(x) = f(x) + g(x)$. However this is not always an efficient solution, because as noted above the proximal operator of function $F(x)$ might be very difficult to compute.
Therefore a different approach relies instead on the so-called *splitting operators*, which leverage the separability of the cost function in order to define a series of smaller and simpler steps to be performed at each iteration.

Peaceman-Rachford splitting

An important class of splitting operators, particularly for their relationship with the ADMM, is centred on the *Peaceman-Rachford* operator [78] defined as

$$T_{PR} = \text{refl}_{\rho f} \circ \text{refl}_{\rho g}\,. \tag{2.15}$$

Consider now the sequence of step-sizes $\{\alpha_k\}_{k\in\mathbb{N}}$ satisfying property (2.10), applying the Krasnosel'skiĭ-Mann iteration to the PR operator yields the so-called *Relaxed Peaceman-Rachford Splitting* (R-PRS) scheme

$$z(k+1) = (1-\alpha_k)z(k) + \alpha_k T_{PR}(z(k)) \tag{2.16}$$

where z is an auxiliary variable from which the optimum of problem (2.14) can be computed according to $x^* = \text{prox}_{\rho g}(z^*)$ with z^* a fixed point of T_{PR}.
A possible implementation of the R-PRS is given by the following equations

$$\psi(k) = \text{prox}_{\rho g}(z(k)) \tag{2.17}$$

$$\xi(k) = \text{prox}_{\rho f}\left(2\psi(k) - z(k)\right) \tag{2.18}$$

$$z(k+1) = z(k) + 2\alpha_k\left(\xi(k) - \psi(k)\right). \tag{2.19}$$

[4]This fact can be easily proved by imposing that $x^* = \text{prox}_{\rho f}(x^*)$ and recalling the definition of proximal operator.

Proof. Substituting the definition of reflective operator in T_{PR} and dropping the time instant indication yields

$$\begin{aligned} T_{PR}(z) = \left(\mathrm{refl}_{\rho f} \circ \mathrm{refl}_{\rho g}\right)(z) &= \left(2\,\mathrm{prox}_{\rho f} - I\right) \circ \left(2\,\mathrm{prox}_{\rho g} - I\right)(z) \\ &= 2\,\mathrm{prox}_{\rho f}\left(2\,\mathrm{prox}_{\rho g}(z) - z\right) + 2\,\mathrm{prox}_{\rho g}(z) + z \\ &= 2\left(\mathrm{prox}_{\rho f}\left(2\,\mathrm{prox}_{\rho g}(z) - z\right) - \mathrm{prox}_{\rho g}(z)\right) + z \\ &= 2(\xi - \psi) + z \end{aligned}$$

where

$$\begin{aligned} \psi &= \mathrm{prox}_{\rho g}(z) \\ \xi &= \mathrm{prox}_{\rho f}\left(2\,\mathrm{prox}_{\rho g}(z) - z\right) = \mathrm{prox}_{\rho f}(2\psi - z) \end{aligned}$$

which correspond to Equations (2.17) and (2.18). Finally, substituting the new formulation obtained for $T_{PR}(z)$ in (2.16) Equation (2.19) follows. □

This scheme splits the computational effort between the two steps (2.17) and (2.18), unlike the PPA (2.13), taking advantage of the particular structure of (2.14).

Finally, there are two more splitting schemes based on the PR operator: the *Douglas-Rachford Splitting* (DRS) [36] and the *Peaceman-Rachford Splitting* (PRS) [78]. Both can be derived from the R-PRS with the particular choice of step-sizes $\alpha_k = 1/2$ and $\alpha_k = 1$, for all $k \in \mathbb{N}$, respectively.

Remark 2.16. The DRS therefore can be defined as the Banach-Picard iteration of the operator $T_{DR} = (I + T_{PR})/2$ while the PRS as the Banach-Picard iteration of T_{PR}.

Forward-backward splitting

A splitting scheme that is not based on the Peaceman-Rachford operator is the so-called *forward-backward splitting* (FBS).
Suppose that in problem (2.14) the function f is continuosly differentiable and that its gradient is Lipschitz continuous with Lipschitz constant γ[5]. Then the *forward-backward operator* is defined as

$$T_{FB} = \mathrm{prox}_{\eta g} \circ (I - \eta \nabla f) \tag{2.20}$$

where $\eta \in (0, 2/\gamma]$, and can be shown to be α_{FB}-averaged [31] with

$$\alpha_{FB} = \frac{1}{2 - \eta\gamma/2}.$$

The FBS is therefore characterised by the Krasnosel'skiĭ-Mann iterate of T_{FB} and can be shown to converge to a minimum of problem (2.14) if the step-sizes are a sequence in $[0, \delta]$ with $\delta = \min\{1, 1/(\gamma\eta)\} + 1/2$ such that $\sum_{k\in\mathbb{N}} \alpha_k(\delta - \alpha_k) = +\infty$ [3].

Remark 2.17. The FBS is sometimes defined as the Banach-Picard iterate of the T_{FB} operator, and the Krasnosel'skiĭ-Mann iterate of T_{FB} is referred to as *generalised FBS*.
Moreover, it is possible to vary the parameter η at each instant, provided that each element in the sequence $\{\eta_k\}_{k\in\mathbb{N}}$ lies in $(0, 2/\gamma]$.

[5] A function $h : \mathcal{X} \to \mathcal{X}$ is said to be *Lipschitz continuous* with constant γ if it verifies $\|h(x) - h(y)\| \leq \gamma \|x - y\|$ for any $x, y \in \mathcal{X}$.

2.2.4 Subdifferential representation of the R-PRS

Turning to the theory of *subdifferentials* it is possible to give an interpretation of the R-PRS that highlights the relationships between the different terms that compose the algorithm.

In the following some notions and results about subdifferentials are briefly introduced, for a broader overview see Appendix A.3.

Definition 2.18 (Subdifferential)**.** Let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be proper. The *subdifferential* of f is the set-valued operator

$$\partial f : \mathcal{X} \to 2^{\mathcal{X}} : x \mapsto \{u \in \mathcal{X} \mid \langle y - x, u\rangle + f(x) \leq f(y)\ \forall y \in \mathcal{X}\}. \tag{2.21}$$

Let $x \in \mathcal{X}$, then f is *subdifferentiable* at x if $\partial f(x) \neq \emptyset$. The elements of $\partial f(x)$ are the *subgradients* of f at x and they will be indicated as $\bar{\nabla} f(x) \in \partial f(x)$.

The definition therefore states that a vector $u \in \mathcal{X}$ is a subgradient of f at x if the continuous affine functional $y \mapsto \langle y - x, u\rangle + f(x)$ coincides with f at x and is below it elsewhere. In other words a vector $u \in \mathcal{X}$ is a subgradient of f at x if it is the "slope" of a continuous affine minorant of f which equals $f(x)$ at x. This second geometrical interpretation is depicted in Figure 2.4.

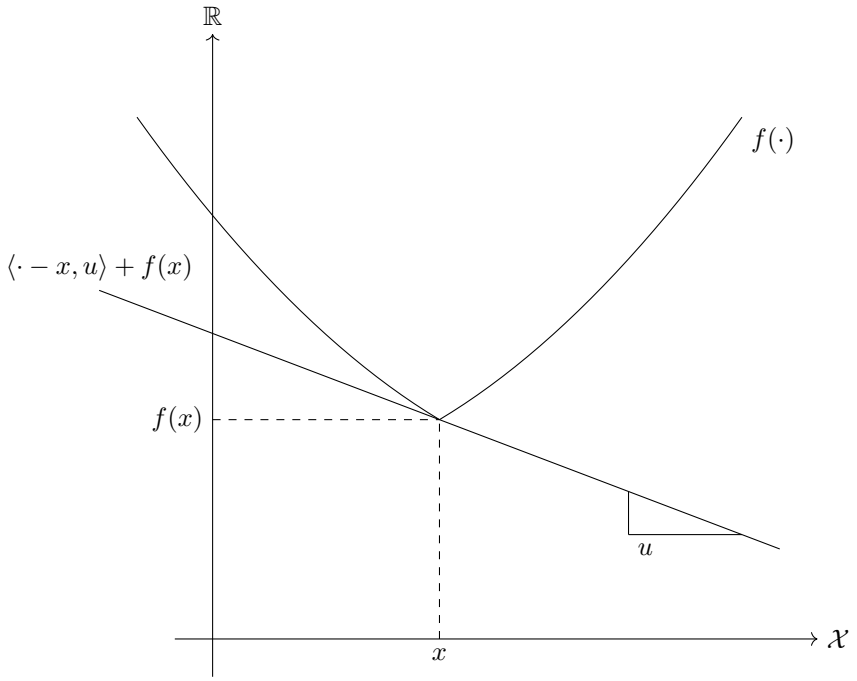


FIGURE 2.4: Graphical representation of subdifferential.

The relationship of subdifferential and convex optimisation is given by the following Theorem.

Theorem 2.19 (Fermat's rule [3, Theorem 16.2])**.** *Let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be proper. Then*

$$\arg\min f = \operatorname{zer} \partial f = \{x \in \mathcal{X} \mid 0 \in \partial f(x)\}.$$

Finally, the proximal operator can be reinterpreted in term of the subdifferential of f.

Proposition 2.20. *Let f be closed, proper and convex, let $x, u \in \mathcal{X}$ and let $\rho > 0$. Then*

$$u = \operatorname{prox}_{\rho f}(x) \quad \Leftrightarrow \quad x - u \in \rho \partial f(u). \tag{2.22}$$

Let now $\bar{\nabla} f(u) \in \partial f(u)$ be a subgradient, then by Proposition 2.20 it holds that

$$x - u = \rho \bar{\nabla} f(u)$$

and substituting $u = \text{prox}_{\rho f}(x)$ and rearranging the terms it follows that

$$\text{prox}_{\rho f}(x) = x - \rho \bar{\nabla} f\left(\text{prox}_{\rho f}(x)\right). \tag{2.23}$$

Moreover, recalling the definition of the reflective operator it holds

$$\text{refl}_{\rho f}(x) = x - 2\rho \bar{\nabla} f\left(\text{prox}_{\rho f}(x)\right). \tag{2.24}$$

The theory of subdifferentials can now be employed in order to derive the following Proposition.

Proposition 2.21. *Let $\psi = \text{prox}_{\rho g}(z)$ and $\xi = \text{prox}_{\rho f}(\text{refl}_{\rho g}(z))$, then the R-PRS implementation* (2.17)–(2.19)*, omitting the time instant indication, is equivalent to*

$$\psi = z - \rho \bar{\nabla} g(\psi) \tag{2.25}$$
$$\xi = \psi - \rho\left(\bar{\nabla} g(\psi) + \bar{\nabla} f(\xi)\right) \tag{2.26}$$
$$T_{PR,\alpha}(z) = z - 2\alpha\rho\left(\bar{\nabla} g(\psi) + \bar{\nabla} f(\xi)\right) \tag{2.27}$$

where $T_{PR,\alpha} = (1-\alpha)I + \alpha T_{PR}$.

Proof. Equation (2.25) follows by the definition used for ψ and Equation (2.23).
By (2.23) applied to function f and (2.24) applied to g it follows

$$\begin{aligned}\xi &= \text{refl}_{\rho g}(z) - \rho \bar{\nabla} f(\xi) \\ &= z - 2\rho \bar{\nabla} g(\psi) - \rho \bar{\nabla} f(\xi) \\ &= (z - \rho \bar{\nabla} g(\psi)) - \rho\left(\bar{\nabla} g(\psi) + \rho \bar{\nabla} f(\xi)\right).\end{aligned}$$

Equation (2.27) follows by substituting (2.25) and (2.26) into (2.19). □

Making use of Proposition 2.21 it is possible to represent the relationships between the variables involved in the Peaceman-Rachford algorithm with the diagram of Figure 2.5.

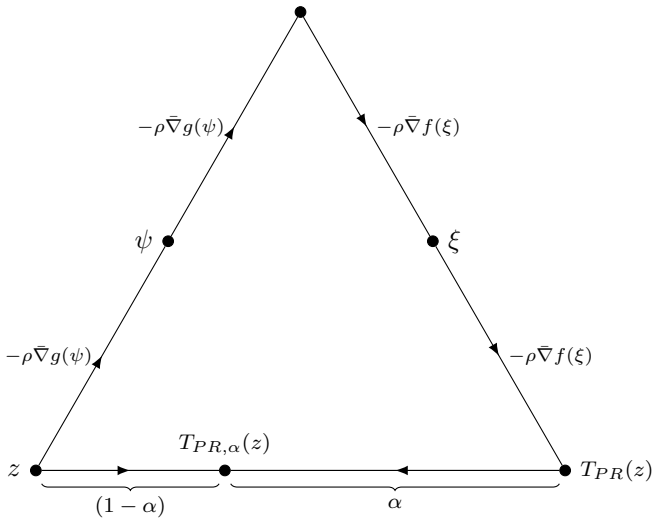


FIGURE 2.5: Relationships between the variables of a relaxed PRS iteration.

Therefore during one iteration of the R-PRS the algorithm travels clock-wise around the perimeter of the triangle starting in z and ending in $T_{PR}(z)$, and then computes the convex combination of z and $T_{PR}(z)$ which yields $T_{PR,\alpha}(z)$.

To conclude, it is possible to use the subdifferential theory to prove that the solution to problem (2.14) is indeed given by $x^* = \text{prox}_{\rho g}(z^*)$ with z^* a fixed point of the Peaceman-Rachford operator.

Proposition 2.22. *Let $x^* = \arg\min_{x\in\mathcal{X}}\{f(x)+g(x)\}$ be a solution of problem* (2.14)*, then it holds*

$$x^* \in \text{zer}(\partial f + \partial g) = \{\text{prox}_{\rho g}(z) \mid z \in \mathcal{X} \text{ s.t. } T_{PR}(z) = z\}$$

or equivalently, if z^* is a fixed point of T_{PR},

$$z^* - x^* = \rho\bar{\nabla} g(x^*) \in \partial g(x^*).$$

Proof. From the Fermat's rule of Theorem 2.19 and the fact that $\partial(f+g) = \partial f + \partial g$ then clearly $\arg\min\{f+g\} = \text{zer}(\partial f + \partial g)$, and the definition of $\text{zer}(\partial f + \partial g)$ is given in [3, Proposition 25.1]. The equivalent characterisation follows from the fact that

$$x^* = \text{prox}_{\rho g}(z^*) = z^* - \rho\bar{\nabla} g(\text{prox}_{\rho g}(z^*))$$

where the second equality is derived from Equation (2.23). Therefore rearranging the terms and using the first equality above it follows that $z^* - x^* = \rho\bar{\nabla} g(x^*)$. □

Finally, note that if $z^* \in \text{fix}(T_{PR})$ then the base of the triangle reduces to a point and the two catheti coincide and therefore at the optimum it is $x^* = \psi^* = \xi^*$.

2.3 ADMM and Splitting Operators

The previous Section introduced operator theory, the problem of finding the fixed points of a non-expansive operator and algorithms designed to solve this problem. Moreover in the last paragraph the application of these concepts was shown to be useful for the solution of convex optimisation problems.
The aim of this Section is now to explain the relationship between the splitting operators and the ADMM or, better, a more general version of the ADMM than that described in Section 2.1.

Recall that the ADMM was introduced to solve problems of the type

$$\begin{aligned} \min_{x\in\mathcal{X},\, y\in\mathcal{Y}} \; & \{f(x) + g(y)\} \\ \text{s.t. } & Ax + By = c \end{aligned}$$

with f and g closed, proper and convex functions.

2.3.1 Dual problem

The first step is to derive the Lagrange dual (or simply dual) problem of (2.1) (reported above for convenience).

Proposition 2.23 ([79])**.** *The dual problem of* (2.1) *is*

$$\min_{w\in\mathcal{W}} \{d_f(w) + d_g(w)\} \tag{2.28}$$

where w represents the vector of Lagrange multipliers, with

$$d_f(w) = f^*(A^\top w), \tag{2.29}$$
$$d_g(w) = g^*(B^\top w) - w^\top c. \tag{2.30}$$

Proof. The Lagrange function of the primal problem is given by

$$\mathcal{L}_0(x, y; w) = f(x) + g(y) - w^\top (Ax + By - c)$$

where the subscript 0 indicates that the Lagrangian is not augmented.
Therefore the Lagrange dual function can be computed as follows

$$\begin{aligned} d(w) &= \min_{x\in\mathcal{X}, y\in\mathcal{Y}} \mathcal{L}_0(x, y; w) \\ &= \min_{x\in\mathcal{X}} \left\{ f(x) - w^\top Ax \right\} + \min_{y\in\mathcal{Y}} \left\{ g(y) - w^\top By \right\} + w^\top c \\ &= -\max_{x\in\mathcal{X}} \left\{ -f(x) + w^\top Ax \right\} - \max_{y\in\mathcal{Y}} \left\{ -g(y) + w^\top By \right\} + w^\top c \\ &= -f^*(A^\top w) - g^*(B^\top w) + w^\top c \end{aligned}$$

where the definition of convex conjugate was used, see Appendix A.1.
Defining the dual functions d_f and d_g according to (2.29) and (2.30) yields

$$d(w) = -(d_f(w) + d_g(w)).$$

Therefore maximising the dual function d is equivalent to minimising the sum of d_f and d_g, which proves (2.28) and hence the Proposition. □

Clearly the dual problem (2.28) belongs to the class of problems (2.14) and therefore can be solved by the application of the splitting methods described in the previous Section 2.2. Note that this is indeed the case because the dual functions d_f and d_g are closed, proper and convex, which can be proved from the fact that both f and g have these properties, see Appendix A.1.

2.3.2 Applying the splitting operators

As was mentioned at the beginning of the Chapter, applying the R-PRS to the dual problem (2.28) yields a general formulation of the ADMM, the so-called *relaxed ADMM* (R-ADMM), of which (2.3)–(2.5) is a particular case.
The R-PRS becomes in this case

$$\psi(k) = \mathrm{prox}_{\rho d_g}(z(k)) \tag{2.31}$$
$$\xi(k) = \mathrm{prox}_{\rho d_f}(2\psi(k) - z(k)) \tag{2.32}$$
$$z(k+1) = z(k) + 2\alpha_k (\xi(k) - \psi(k)) \tag{2.33}$$

with the step-sizes satisfying (2.10). The following Proposition is instrumental in order to implement the algorithm.

Proposition 2.24. *The two problems* (2.31) *and* (2.32) *can be solved according to the following schemes*

$$\begin{aligned} y(k) &= \arg\min_y \left\{ g(y) - z^\top(k)(By - c) + \frac{\rho}{2} \|By - c\|^2 \right\} \\ \psi(k) &= z(k) - \rho\,(By(k) - c) \end{aligned}$$

and

$$\begin{aligned} x(k) &= \arg\min_x \left\{ f(x) - (2\psi(k) - z(k))^\top Ax + \frac{\rho}{2} \|Ax\|^2 \right\} \\ \xi(k) &= 2\psi(k) - z(k) - \rho Ax(k) \end{aligned}$$

respectively.

Proof. Consider first problem (2.31), by the definition of proximal operator and of the dual function d_g it is necessary to find the argument of

$$\begin{aligned} \min_s \left\{ d_g(s) + \frac{1}{2\gamma} \|s - z\|^2 \right\} &= \min_s \left\{ g^*(B^\top s) - s^\top c + \frac{1}{2\gamma} \|s - z\|^2 \right\} \\ &= \min_s \left\{ \max_u \left\{ s^\top Bu - g(u) \right\} - s^\top c + \frac{1}{2\gamma} \|s - z\|^2 \right\} \\ &= \min_s \left\{ \max_u \left\{ s^\top Bu - g(u) - s^\top c + \frac{1}{2\gamma} \|s - z\|^2 \right\} \right\} \\ &= \max_u \left\{ \min_s \left\{ s^\top Bu - g(u) - s^\top c + \frac{1}{2\gamma} \|s - z\|^2 \right\} \right\} \\ &= \max_u \left\{ \min_s \left\{ s^\top (Bu - c) - g(u) + \frac{1}{2\gamma} \|s - z\|^2 \right\} \right\} \\ &= \max_u \left\{ \min_s \left\{ s^\top (Bu - c) + \frac{1}{2\gamma} \|s - z\|^2 \right\} - g(u) \right\}. \end{aligned}$$

Imposing the first-order necessary condition, the solution to problem

$$\min_s \left\{ s^\top (Bu - c) + \frac{1}{2\gamma} \|s - z\|^2 \right\}$$

is $s^* = z - \gamma(Bu - c)$ and hence

$$\min_s \left\{ s^\top (Bu - c) + \frac{1}{2\gamma} \|s - z\|^2 \right\} = z^\top (Bu - c) - \frac{\gamma}{2} \|Bu - c\|^2 .$$

Therefore it follows that

$$\begin{aligned} \min_s \left\{ d_g(s) + \frac{1}{2\gamma} \|s - z\|^2 \right\} &= \max_u \left\{ z^\top (Bu - c) - \frac{\gamma}{2} \|Bu - c\|^2 - g(u) \right\} \\ &= -\min_u \left\{ g(u) - z^\top (Bu - c) + \frac{\gamma}{2} \|Bu - c\|^2 \right\}. \end{aligned}$$

This problem can now be solved applying the *method of multipliers* [10] which gives the desired result

$$\begin{aligned} y &= \arg\min_y \left\{ g(y) - z^\top (By - c) + \frac{\rho}{2} \|By - c\|^2 \right\} \\ \psi &= z - \rho\,(By - c)\,. \end{aligned}$$

For what concerns problem (2.32) the result

$$x = \arg\min_x \left\{ f(x) - (2\psi - z)^\top Ax + \frac{\rho}{2} \|Ax\|^2 \right\}$$
$$\xi = 2\psi - z - \rho Ax$$

can be obtained with the same procedure applied above to (2.31) where g is substituted by f, B by A and c by the null vector. □

Therefore the R-ADMM is characterised by the following set of five equations

$$y(k) = \arg\min_y \left\{ g(y) - z^\top(k)(By - c) + \frac{\rho}{2} \|By - c\|^2 \right\} \tag{2.34}$$
$$\psi(k) = z(k) - \rho \left(By(k) - c \right) \tag{2.35}$$
$$x(k) = \arg\min_x \left\{ f(x) - (2\psi(k) - z(k))^\top Ax + \frac{\rho}{2} \|Ax\|^2 \right\} \tag{2.36}$$
$$\xi(k) = 2\psi(k) - z(k) - \rho Ax(k) \tag{2.37}$$
$$z(k+1) = z(k) + 2\alpha_k \left(\xi(k) - \psi(k) \right). \tag{2.38}$$

Recalling now the definition of the augmented Lagrangian (2.2)

$$\mathcal{L}(x, y; w) = f(x) + g(y) - w^\top (Ax + By - c) + \frac{\rho}{2} \|Ax + By - c\|^2$$

the following Proposition traces a parallel between the R-ADMM and the ADMM presented in the first Section.

Proposition 2.25. *The relaxed ADMM defined by the five equations* (2.34)–(2.38) *is equivalently expressed in the formulation*

$$y(k+1) = \arg\min_y \left\{ \mathcal{L}(x(k), y; w(k)) + \rho(2\alpha_k - 1)\langle By, (Ax(k) + By(k) - c) \rangle \right\} \tag{2.39}$$
$$w(k+1) = w(k) - \rho(Ax(k) + By(k+1) - c) - \rho(2\alpha_k - 1)(Ax(k) + By(k) - c) \tag{2.40}$$
$$x(k+1) = \arg\min_x \mathcal{L}(x, y(k+1); w(k+1)). \tag{2.41}$$

Proof. From Equations (2.37) and (2.35) follow respectively

$$2\psi(k) - z(k) = \xi(k) + \rho Ax(k) \tag{2.42}$$
$$z(k) = \psi(k) + \rho(By(k) - c). \tag{2.43}$$

Combining these two results yields

$$\xi(k) = \psi(k) - \rho(Ax(k) + By(k) - c) \tag{2.44}$$

and substituting back into (2.42) it follows

$$2\psi(k) - z(k) = \psi(k) - \rho(By(k) - c). \tag{2.45}$$

Plugging now (2.45) into Equation (2.36) yields

$$\begin{aligned} x(k) &= \arg\min_x \left\{ f(x) - \psi(k)^\top Ax + \rho(By(k) - c)^\top Ax + \frac{\rho}{2} \|Ax\|^2 \right\} \\ &= \arg\min_x \left\{ f(x) + g(y(k)) - \psi(k)^\top (Ax + By(k) - c) + \frac{\rho}{2} \|Ax + By(k) - c\|^2 \right\} \\ &= \arg\min_x \mathcal{L}(x, y(k); \psi(k)) \end{aligned}$$

where the second equality was derived by adding the terms $g(y(k))$, $(\rho/2)\|By(k) - c\|^2$ and $-\psi(k)^\top(By(k) - c)$ which do not depend on x, and the third equality by using the definition of augmented Lagrangian (2.2).

From Equation (2.44) it follows that $\xi(k) - \psi(k) = -\rho(Ax(k) + By(k) - c)$, which can be plugged into (2.38) to obtain

$$\begin{aligned} z(k+1) &= z(k) - 2\alpha_k \rho(Ax(k) + By(k) - c) \\ &= \psi(k) + \rho(By(k) - c) - 2\alpha_k \rho(Ax(k) + By(k) - c) \\ &= \psi(k) - \rho(2\alpha_k - 1)(Ax(k) + By(k) - c) - \rho Ax(k) \end{aligned} \tag{2.46}$$

where the second equality holds because of (2.43) and the third by simply adding and subtracting $\rho Ax(k)$.
Evaluating (2.43) at time $k+1$ and reordering the terms yields $\psi(k+1) = z(k+1) - \rho(By(k+1) - c)$, therefore combining it with (2.46) results in

$$\psi(k+1) = \psi(k) - \rho(Ax(k) + By(k+1) - c) - \rho(2\alpha_k - 1)(Ax(k) + By(k) - c). \tag{2.47}$$

Finally computing Equation (2.34) at time $k+1$ and using (2.46) gives

$$\begin{aligned} y(k+1) = \arg\min_y \Bigg\{ & f(x(k)) + g(y) - \psi^\top(k)(Ax(k) + By - c) + \\ & + \frac{\rho}{2} \|Ax(k) + By - c\|^2 + \rho(2\alpha_k - 1)\langle By, Ax(k) + By(k) - c \rangle \Bigg\} \end{aligned}$$

where the terms $f(x(k))$, $-\psi^\top(k)Ax(k)$ and $\rho\|Ax(k)\|^2$ were added since they do not depend on y, and for the same reason the term $-\rho(2\alpha_k - 1)\langle c, Ax(k) + By(k) - c\rangle$ was discarded. And recalling once again the definition of the definition of the augmented Lagrangian yields the desired result.

Finally, by renaming ψ as the dual variable w, Proposition 2.25 is proved. □

Remark 2.26. Note that the ADMM and the R-ADMM of the Proposition differ only for the terms weighted by $2\alpha_k - 1$ in the updates for the y and w. Therefore the simple ADMM can be recovered setting $\alpha_k = 1/2$ for all $k \in \mathbb{N}$. Which in turn implies that it is the application of the DRS on the dual problem (2.28), as indeed was shown in [41].

Remark 2.27. A key difference between ADMM and R-ADMM is that the first has one single tunable parameter, ρ, while the second in addition to ρ has the entire sequence of step-sizes $\{\alpha_k\}_{k\in\mathbb{N}}$ to be chosen. Section 6.2 will explore the importance of the step-size sequence for the convergence rate of the algorithm with the aid of simulations.

Remark 2.28. The trajectory $k \to x(k)$ that is generated by the implementation of the R-ADMM in (2.34)–(2.38) is equal to the trajectory generated by the implementation (2.39)–(2.41) if the initial conditions $x(0)$ and $y(0)$ are the same and if, given the initial condition

$w(0)$ for the Lagrange multipliers, it holds

$$\begin{aligned}\psi(0) &= w(0),\\ \xi(0) &= w(0) - \rho(Ax(0) + By(0) - c),\\ z(0) &= w(0) + \rho(By(0) - c).\end{aligned}$$

This is a clear consequence of the relationships between the variables used in the two implementations.

To conclude this Section, Figure 2.6 depicts the relationships between the algorithms introduced in this Chapter.

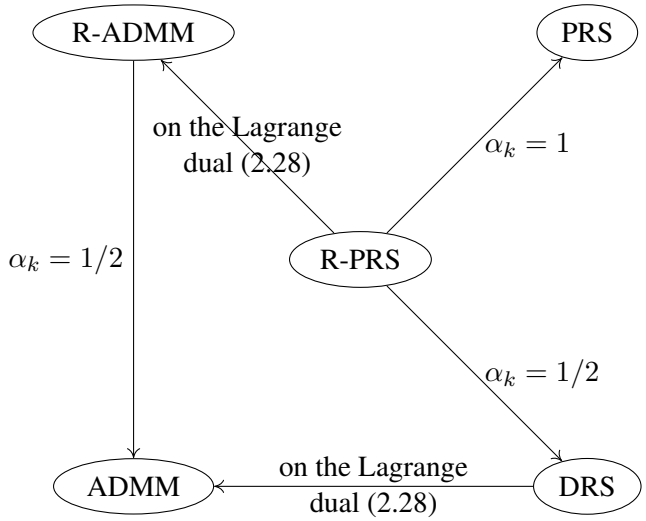


FIGURE 2.6: Relationships between the algorithms.

Chapter 3

Distributed Alternating Direction Method of Multipliers

The present Chapter describes the application of the Alternating Direction Method of Multipliers to *distributed consensus optimisation* problems of which it proves convergence and, in the particular case of strongly convex cost functions, linear convergence.

3.1 Distributed Consensus Optimisation with ADMM

This Section introduces the problem that will be the focus of the following Chapters.

3.1.1 Problem formulation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, with $\mathcal{V}$ the set of N nodes and $\mathcal{E}$ the set of undirected edges. Consider the convex optimisation problem

$$\min_{x} \sum_{i=1}^{N} f_i(x) \tag{3.1}$$

where each function $f_i : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is closed, proper and convex, and available only to node i[1]. The aim is to solve (3.1) in a distributed fashion in which nodes use only the information available to them and that is sent from their neighbours.
The problem then is cast as a *consensus optimisation* by the introduction of the new variables $x_i, i = 1, \dots, N$, each one assigned to a node, and requiring that at the optimum the variable of node i be equal to those of its neighbours. That is, (3.1) becomes

$$\begin{aligned} &\min_{x_i,\, \forall i} \sum_{i=1}^{N} f_i(x_i) \\ &\text{s.t. } x_i = x_j \ \ \forall (i,j) \in \mathcal{E} \end{aligned} \tag{3.2}$$

where the introduction of the $2|\mathcal{E}|$ constraints ensures that the consensus is reached at the optimum.
In order to apply the R-ADMM to this problem it is necessary to reformulate the constraints with the introduction of the *bridge variables* y_{ij} and y_{ji}, one for each constraint:

$$\begin{aligned} x_i &= y_{ij} \\ x_j &= y_{ji} \qquad \forall (i,j) \in \mathcal{E}. \\ y_{ij} &= y_{ji} \end{aligned}$$

[1] Note that hereafter the focus is restricted to problems defined on $\mathbb{R}^n$ and not the general Hilbert space $\mathcal{X}$.

Let now $\mathbf{x} \in \mathbb{R}^{nN}$ be the vector containing all variables x_i, $i = 1, \dots, N$, $\mathbf{y} \in \mathbb{R}^{2n|\mathcal{E}|}$ be the vector containing all bridge variables, and define the function $f(\mathbf{x}) = \sum_{i=1}^{N} f_i(x_i)$. The problem can be equivalently formulated as follows

$$\begin{aligned} \min_{\mathbf{x}} \ & f(\mathbf{x}) \\ \text{s.t.} \ & A\mathbf{x} + \mathbf{y} = 0 \\ & \mathbf{y} = P\mathbf{y} \end{aligned}$$

with a suitable matrix A and P the permutation matrix swapping element y_{ij} with y_{ji}.
In order to apply the R-ADMM of the previous Section to this problem, define the *indicator function* of a matrix M as

$$\iota_M(x) = \begin{cases} 0 & \text{if } Mx = 0 \\ +\infty & \text{otherwise} \end{cases} \tag{3.3}$$

which therefore penalises all vectors that are not in the kernel of matrix M. Hence the constraint $\mathbf{y} = P\mathbf{y}$ is enforced using the indicator function $\iota_{(I-P)}(\mathbf{y})$, and the problem is finally

$$\begin{aligned} \min_{\mathbf{x}} \ & \left\{ f(\mathbf{x}) + \iota_{(I-P)}(\mathbf{y}) \right\} \\ \text{s.t.} \ & A\mathbf{x} + \mathbf{y} = 0 \end{aligned} \tag{3.4}$$

which corresponds to problem (2.1) with $f(x) = f(\mathbf{x})$, $g(y) = \iota_{(I-P)}(\mathbf{y})$, $B = I$ and $c = 0$. Hereafter (one of) the optimal solution(s) of problem (3.4) will be denoted with $\mathbf{x}^* = \mathbb{1} \otimes x^*$ where x^* is the solution to (3.1).

3.1.2 Distributed ADMM

The following Proposition reports the formulation of the distributed R-ADMM.

Proposition 3.1. *The distributed implementation of the R-ADMM for problem* (3.4) *is given by the two iterates*

$$x_i(k) = \arg\min_{x_i} \left\{ f_i(x_i) - \sum_{j \in \mathcal{N}_i} z_{ji}^\top(k) x_i + \frac{\rho|\mathcal{N}_i|}{2} \|x_i\|^2 \right\} \tag{3.5}$$

$$z_{ij}(k+1) = (1 - \alpha_k) z_{ij}(k) - \alpha_k z_{ji}(k) + 2\alpha_k \rho x_i(k), \quad \forall j \in \mathcal{N}_i \tag{3.6}$$

that are carried out by each node in parallel.

Proof. Equation (2.34) applied to the distributed consensus problem (3.4) becomes

$$\begin{aligned} \mathbf{y}(k) &= \arg\min_{\mathbf{y}} \left\{ \iota_{(I-P)}(\mathbf{y}) - \mathbf{z}^\top(k)\mathbf{y} + \frac{\rho}{2} \|\mathbf{y}\|^2 \right\} \\ &= \arg\min_{\mathbf{y}=P\mathbf{y}} \left\{ -\mathbf{z}^\top(k)\mathbf{y} + \frac{\rho}{2} \|\mathbf{y}\|^2 \right\} \end{aligned}$$

which is a convex constrained optimisation problem. The Karush-Kuhn-Tacker (KKT) conditions [11] for this problem are

$$(I - P)\bar{\mathbf{y}} = 0 \tag{3.7}$$

$$-\mathbf{z}(k) + \rho\bar{\mathbf{y}} + (I - P)\bar{\boldsymbol{\nu}} = 0 \tag{3.8}$$

where $\bar{\mathbf{y}}$ and $\bar{\boldsymbol{\nu}}$ represent the optimal value of the primal and dual variables respectively. Reorganising Equation (3.8) yields

$$\bar{\mathbf{y}} = \frac{1}{\rho}\left[\mathbf{z}(k) + (I-P)\bar{\boldsymbol{\nu}}\right] \tag{3.9}$$

and substituting it into the right-hand side of $\bar{\mathbf{y}} = P\bar{\mathbf{y}}$ (3.7) results in

$$\bar{\mathbf{y}} = \frac{1}{\rho}P\left[\mathbf{z}(k) + (I-P)\bar{\boldsymbol{\nu}}\right] = \frac{1}{\rho}\left[P\mathbf{z}(k) - (I-P)\bar{\boldsymbol{\nu}}\right] \tag{3.10}$$

which is true since by $P^2 = I$ it holds $P(I-P) = P - I = -(I-P)$.
Finally, summing (3.9) and (3.10) gives

$$\mathbf{y}(k) = \bar{\mathbf{y}} = \frac{1}{2\rho}(I+P)\mathbf{z}(k). \tag{3.11}$$

For what concerns Equations (2.35) and (2.37) by the result (3.11) obtained above it follows that

$$\begin{aligned}\boldsymbol{\psi}(k) &= \frac{1}{2}(I-P)\mathbf{z}(k)\\ \boldsymbol{\xi}(k) &= -P\mathbf{z}(k) - \rho A\mathbf{x}(k)\end{aligned}$$

where the fact $2\boldsymbol{\psi}(k) - \mathbf{z}(k) = -P\mathbf{z}(k)$ was used. Moreover (2.36) becomes

$$\mathbf{x}(k) = \arg\min_{\mathbf{x}}\left\{f(\mathbf{x}) + (P\mathbf{z}(k))^\top A\mathbf{x} + \frac{\rho}{2}\left\|A\mathbf{x}\right\|^2\right\}. \tag{3.12}$$

Using the results derived above it is possible to rewrite (2.38) as

$$\mathbf{z}(k+1) = (1-\alpha_k)\mathbf{z}(k) - \alpha_k P\mathbf{z}(k) - 2\alpha_k\rho A\mathbf{x}(k). \tag{3.13}$$

Notice that the updates for the primal and auxiliary variable depend only on the vectors $\mathbf{z}(k)$ and $\mathbf{x}(k)$, which means that all the information necessary to apply the algorithm is contained in Equations (3.12) and (3.13).
Making use of the particular structure of A and P the proof can now be concluded. Recall that the matrix $A \in \mathbb{R}^{2|\mathcal{E}|\times N}$ with the (ij)-th row selects the component x_i of $\mathbf{x}$ and multiplies it by -1, and that the permutation matrix $P \in \mathbb{R}^{2|\mathcal{E}|\times 2|\mathcal{E}|}$ swaps the (ij)-th element with the (ji)-th.
Therefore it follows

$$\begin{aligned}(P\mathbf{z}(k))^\top A\mathbf{x} &= \begin{bmatrix}\cdots & z_{ji}^\top(k) & \cdots & z_{ij}^\top(k) & \cdots\end{bmatrix}\begin{bmatrix}\vdots\\ -x_i\\ \vdots\\ -x_j\\ \vdots\end{bmatrix}\\ &= -\sum_{(i,j)\in\mathcal{E}}\left(z_{ji}^\top(k)x_i + z_{ij}^\top(k)x_j\right)\\ &= -\sum_{i=1}^{N}\left(\sum_{j\in\mathcal{N}_i} z_{ji}^\top(k)\right)x_i.\end{aligned}$$

Moreover since x_i appears in $|\mathcal{N}_i|$ constraints, it holds $\|A\mathbf{x}\|^2 = \sum_{i=1}^{N} |\mathcal{N}_i| \, \|x_i\|^2$.
Finally substituting these results back into (3.12) and (3.13) proves the Proposition. □

Notice that the only information that node i needs in order to compute the updates (3.5) and (3.6) are the variables $\{z_{ji}\}_{j\in\mathcal{N}_i}$ received from its neighbours. The variables that the node needs to store in local memory are therefore x_i, $\{z_{ij}\}_{j\in\mathcal{N}_i}$ and $\{z_{ji}\}_{j\in\mathcal{N}_i}$.

Suppose however that node i is tasked with updating variables $\{z_{ji}\}_{j\in\mathcal{N}_i}$ instead of variables $\{z_{ij}\}_{j\in\mathcal{N}_i}$. In this scenario the last two terms of update (3.6), which is performed by node j, both require information that is transmitted from node i.
Therefore it is possible to derive a new implementation of the R-ADMM in which the nodes exchange the auxiliary variables

$$q_{i\to j} = -z_{ji}(k) + 2\rho x_i(k) \tag{3.14}$$

instead of the z variables, and (3.6) becomes

$$z_{ij}(k+1) = (1-\alpha_k) z_{ij}(k) + \alpha_k q_{i\to j}. \tag{3.15}$$

The main advantage of this formulation is that update (3.5) depends on information that is stored and updated by node i itself, and only the $\{z_{ji}\}_{j\in\mathcal{N}_i}$ variables depend on outside information, namely $\{q_{j\to i}\}_{j\in\mathcal{N}_i}$.
The following Algorithm 1 formalises the implementation of the R-ADMM for distributed scenarios described above. Note that this implementation assumes the updates to occur in a synchronous fashion.

Algorithm 1 Modified distributed R-ADMM.

Input: step-sizes $\{\alpha_k\}_{k\in\mathbb{N}}$, penalty ρ, termination condition K.
Initialise: $x_i(0)$ and $z_{ji}(0)$ for each node i and neighbour $j \in \mathcal{N}_i$.

1: $k \leftarrow 0$
2: **while** $k < K$ each agent i **do**
3: compute $x_i(k)$ according to (3.5)

$$x_i(k) = \arg\min_{x_i} \left\{ f_i(x_i) - \sum_{j\in\mathcal{N}_i} z_{ji}^\top(k) x_i + \frac{\rho|\mathcal{N}_i|}{2} \|x_i\|^2 \right\}$$

4: for all $j \in \mathcal{N}_i$ compute $q_{i\to j}$ as in (3.14)

$$q_{i\to j} = -z_{ji}(k) + 2\rho x_i(k)$$

5: transmit $q_{i\to j}$ to node j
6: gather $q_{j\to i}$ from each neighbour j
7: update z_{ji} as in (3.15)

$$z_{ji}(k+1) = (1-\alpha_k) z_{ji}(k) + \alpha_k q_{j\to i}$$

8: $k \leftarrow k+1$
9: **end while**

While Algorithm 1 was obtained applying the five equations (2.34)–(2.38), a different implementation can be derived by applying the three equations given in Proposition 2.25. Algorithm 2 presents such implementation.

Proposition 3.2. *Applying the three equations of Proposition 2.25 to the distributed consensus optimisation problem* (3.4) *yields Algorithm 2.*

Algorithm 2 Distributed R-ADMM using Proposition 2.25.

Input: step-sizes $\{\alpha_k\}_{k\in\mathbb{N}}$, penalty ρ, termination condition K.
Initialise: $x_i(0)$, $y_{ij}(0)$ and $w_{ij}(0)$ for each node i and neighbour $j \in \mathcal{N}_i$.

1: $k \leftarrow 0$
2: **while** $k < K$ each agent i **do**
3: gather $x_j(k)$, $y_{ji}(k)$ and $w_{ji}(k)$ from each neighbour $j \in \mathcal{N}_i$
4: compute in order

$$y_{ij}(k+1) = \frac{1}{2\rho}\Big[(w_{ij}(k) + w_{ji}(k)) + \\ + 2\alpha\rho(x_i(k) + x_j(k)) - \rho(2\alpha - 1)(y_{ij}(k) + y_{ji}(k))\Big]$$

$$w_{ij}(k+1) = \frac{1}{2}\Big[(w_{ij}(k) - w_{ji}(k)) + \\ + 2\alpha\rho(x_i(k) - x_j(k)) - \rho(2\alpha - 1)(y_{ij}(k) - y_{ji}(k))\Big]$$

$$x_i(k+1) = \arg\min_{x_i} \Bigg\{ f_i(x_i) + \frac{\rho}{2}|\mathcal{N}_i|\|x_i\|^2 + \\ + \Big(\sum_{j\in\mathcal{N}_i} w_{ij}(k+1) - \rho y_{ij}(k+1)\Big)^\top x_i \Bigg\}$$

5: broadcast $x_i(k+1)$, $y_{ij}(k+1)$ and $w_{ij}(k+1)$ to all neighbours
6: $k \leftarrow k+1$
7: **end while**

Proof. The proof of this result resembles very closely that of Proposition (3.1) and therefore is only sketched.
By applying the KKT conditions to problem (2.39) gives the following

$$\bar{\mathbf{y}} = \frac{1}{\rho}\left[\mathbf{w}(k) - 2\alpha_k\rho A\mathbf{x}(k) - \rho(2\alpha_k - 1)\mathbf{y}(k) + (I - P)\bar{\boldsymbol{\nu}}\right]$$
$$\bar{\mathbf{y}} = P\bar{\mathbf{y}}.$$

Now it is possible to substitute the first equation into the right-hand side of the second and sum the result with the first equation, which yields

$$\mathbf{y}(k+1) = \bar{\mathbf{y}} = \frac{1}{2\rho}(I + P)\left[\mathbf{w}(k) - 2\alpha_k\rho A\mathbf{x}(k) - \rho(2\alpha_k - 1)\mathbf{y}(k)\right]. \tag{3.16}$$

A simple substitution of (3.16) into (2.40) results in

$$\mathbf{w}(k+1) = \frac{1}{2}(I - P)\left[\mathbf{w}(k) - 2\alpha_k\rho A\mathbf{x}(k) - \rho(2\alpha_k - 1)\mathbf{y}(k)\right].$$

Using the results obtained above and the structure of the A and P matrices yields the desired result. □

Note that Algorithms 1 and 2 differ for the number of variables that need to be stored, updated, and transmitted to each neighbour, which are reported in Table 3.1. Moreover, Algorithm 1 makes use of temporary variables that do not have to be stored from iteration to iteration. Recall however that what here is referred to as 'variable' is in general a vector of dimension n.

TABLE 3.1: Comparison of R-ADMM implementations.

	Algorithm 1	Algorithm 2
Update and store	$\lvert\mathcal{N}_i\rvert + 1$	$2\lvert\mathcal{N}_i\rvert + 1$
Temporary	$\lvert\mathcal{N}_i\rvert$	$-$
Send	$\lvert\mathcal{N}_i\rvert$	$2\lvert\mathcal{N}_i\rvert + 1$

3.2 Convergence and Linear Convergence Rate

The following Lemma will be instrumental in proving the convergence of the proposed algorithm.

Lemma 3.3 ([3, Corollary 27.4])**.** *Consider problem* (2.14)*, rewritten here*

$$\min_{x\in\mathcal{X}} \{f(x) + g(x)\},$$

with $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ closed, proper and convex functions. Assume that the problem has solution, and let $\rho > 0$, $\{\alpha_k\}_{k\in\mathbb{N}}$ be a sequence in $[0, 1]$ such that $\sum_{k\in\mathbb{N}} \alpha_k(1 - \alpha_k) = +\infty$, and $x(0) \in \mathcal{X}$.
Assume to apply equations (2.17)–(2.19) *to the problem. Then there exists z^* such that*

- $x^* = \mathrm{prox}_{\rho g}(z^*) \in \arg\min_x\{f(x) + g(x)\}$,
- *with $\{x(k)\}_{k\in\mathbb{N}}$ and $\{z(k)\}_{k\in\mathbb{N}}$ that converge weakly to x^* and z^* respectively.*

Now Proposition 3.4 states the convergence result for Algorithm 1.

Proposition 3.4. *Consider Algorithm 1. Let $\rho > 0$ and let $\{\alpha_k\}_{k\in\mathbb{N}}$ be a sequence in $[0, 1]$ such that $\sum_{k\in\mathbb{N}} \alpha_k(1 - \alpha_k) = +\infty$. Then, for any initial conditions, the trajectories $k \to x_i(k)$, $i \in \mathcal{V}$, generated by the Algorithm converge to the optimal solution of* (3.1)*, i.e.*

$$\lim_{k\to\infty} x_i(k) = x^*, \qquad \forall i \in \mathcal{V},$$

for any $x_i(0)$ and $z_{ji}(0)$, $j \in \mathcal{N}_i$.

Proof. The aim is to show that Lemma 3.3 applies to the dual of problem (3.4). First of all, by the formulation of the problem f is proper and convex. Moreover, the set of vectors $\mathbf{y}$ that satisfy $(I - P)\mathbf{y} = 0$ is convex, and given that the indicator function of a convex set is convex [3, Example 8.3] and, by definition, proper, it follows that g is proper and convex.
Now [88, Theorem 12.2] states that the convex conjugate of a proper and convex function is closed, proper and convex. Therefore both d_f and d_g are closed, proper and convex, which means that Lemma 3.3 applies to the dual of problem (3.4).
Therefore the R-ADMM applied to the dual converges weakly to the dual optimum $w^* = \mathrm{prox}_{\rho d_g}(z^*)$. But since the duality gap is zero, attaining the optimum for the dual problem implies that the optimum for the primal is attained as well. □

In order to prove exponential convergence of the R-ADMM the cost functions f_i in (3.1) are assumed to be strongly convex[2]. Suppose for simplicity the step-size to be constant, *i.e.* $\alpha_k = \alpha$ for any $k \in \mathbb{N}$. Before stating the main result the following Lemmas are derived.

Lemma 3.5. *The update equations* (3.5) *and* (3.6) *in the case of strongly convex local cost functions can be rewritten as*

$$\mathbf{x}(k) = \rho N \operatorname{prox}_h \left(\frac{1}{\rho} N^{-1} D \mathbf{z}(k) \right) \tag{3.17}$$

$$\mathbf{z}(k+1) = A\mathbf{z}(k) - FE(\mathbf{x}(k) - \mathbf{x}^*) + c \tag{3.18}$$

with the notation conventions defined in the proof of this result. Moreover Equation (3.17) *can be rewritten as*

$$\mathbf{x}(k) = H^{-1} D \mathbf{z}(k) - H^{-1} E(\mathbf{x}(k) - \mathbf{x}^*) - H^{-1} b. \tag{3.19}$$

Proof. The update equation (3.5) for the x_i can be rewritten as follows

$$\begin{aligned} x_i(k) &= \arg\min_{x_i} \left\{ f_i(x_i) - (D_i \mathbf{z}(k))^\top x_i + \frac{\rho |\mathcal{N}_i|}{2} \|x_i\|^2 \right\} \\ &= \rho |\mathcal{N}_i| \arg\min_{x_i} \left\{ \frac{1}{\rho |\mathcal{N}_i|} f_i(x_i) - \frac{1}{\rho |\mathcal{N}_i|} (D_i \mathbf{z}(k))^\top x_i + \frac{1}{2} \|x_i\|^2 \right\} \\ &= \rho |\mathcal{N}_i| \arg\min_{x_i} \left\{ h_i(x_i) + \frac{1}{2} \left\| x_i - \frac{1}{\rho |\mathcal{N}_i|} D_i \mathbf{z}(k) \right\|^2 \right\} \end{aligned}$$

where the definition $h_i(x_i) = f_i(x_i)/\rho|\mathcal{N}_i|$ was used and the last inequality was obtained by adding the term $\|D_i \mathbf{z}(k)/(\rho|\mathcal{N}_i|)\|^2$, independent from x_i.
Recalling the definition of proximal operator, it follows that

$$x_i(k) = \rho |\mathcal{N}_i| \operatorname{prox}_{h_i} \left(\frac{1}{\rho |\mathcal{N}_i|} D_i \mathbf{z}(k) \right). \tag{3.20}$$

Defining $h(\mathbf{x}) = \begin{bmatrix} h_1(x_1) & \cdots & h_N(x_N) \end{bmatrix}^\top$, with a slight abuse of notation it is possible to give the following matricial update equation

$$\mathbf{x}(k) = \rho N \operatorname{prox}_h \left(\frac{1}{\rho} N^{-1} D \mathbf{z}(k) \right) \tag{3.21}$$

where $N = \operatorname{diag}\{|\mathcal{N}_i|, i = 1, \ldots, N\}$, $D = \begin{bmatrix} D_1^\top & \cdots & D_N^\top \end{bmatrix}^\top$ and

$$\operatorname{prox}_h(\mathbf{u}) = \begin{bmatrix} \operatorname{prox}_{h_1}(u_1) \\ \vdots \\ \operatorname{prox}_{h_N}(u_N) \end{bmatrix}.$$

In order to derive an expression for the z variables first it is necessary to reformulate the update equation of the x_i in a suitable manner.
Recalling that the function f_i is twice differentiable, the first order optimality condition for the problem (3.20) is

$$\nabla f_i(x_i(k)) - D_i \mathbf{z}(k) + \rho |\mathcal{N}_i| x_i(k) = 0 \tag{3.22}$$

[2]That is for any $x \in \mathbb{R}^n$ there exists $c > 0$ such that $\nabla^2 f_i(x) \succ cI$.

where $x_i(k)$ plays the role of the optimum.
Moreover, when x_i is sufficiently close to the (unique) optimum x^* the first order Taylor expansion for the *gradient* of f_i can be computed, which is

$$\nabla f_i(x_i) = \nabla f_i(x^*) + \nabla^2 f_i(x^*)(x_i - x^*) + E(x_i - x^*) \tag{3.23}$$

where $\|E_i(x)\| / \|x\| \to 0$ when $x \to 0$.
Now combining equations (3.22) and (3.23), with the second evaluated in $x_i(k)$, it follows

$$\begin{aligned} D_i\mathbf{z}(k) - \rho|\mathcal{N}_i|x_i(k) = \\ = \nabla f_i(x^*) + \nabla^2 f_i(x^*)(x_i(k) - x^*) + E(x_i(k) - x^*). \end{aligned}$$

Solving for the $x_i(k)$ yields

$$x_i(k) = H_i^{-1} D_i\mathbf{z}(k) - H_i^{-1} E_i(x_i(k) - x^*) - H_i^{-1} b_i$$

where $H_i = \nabla^2 f_i(x^*) + \rho|\mathcal{N}_i|I$, invertible by strong convexity, and $b_i = \nabla f_i(x^*) - \nabla^2 f_i(x^*)x^*$. The following matricial update can hence be obtained by stacking together all these equations

$$\mathbf{x}(k) = H^{-1} D\mathbf{z}(k) - H^{-1} E(\mathbf{x}(k) - \mathbf{x}^*) - H^{-1} b$$

choosing matrix $H = \mathrm{diag}\{H_1, \dots, H_N\}$, vector $b = [b_1^\top \cdots b_N^\top]^\top$, and the o-little function $E(\mathbf{x}) = [E_1(x_1)^\top \cdots E_N(x_N)^\top]^\top$.

Plugging (3.19) into the update $\mathbf{z}(k+1) = ((1-\alpha)I - \alpha P)\mathbf{z}(k) + 2\alpha\rho C\mathbf{x}(k)$ results in

$$\mathbf{z}(k+1) = A\mathbf{z}(k) - FE(\mathbf{x}(k) - \mathbf{x}^*) + c$$

with $A = (1-\alpha)I - \alpha P + 2\alpha\rho C H^{-1} D$, $F = 2\alpha\rho C H^{-1}$ and $c = -2\alpha\rho C H^{-1} b$. □

Remark 3.6. Lemma 3.5, proved with an approach similar to that employed in [58], shows that in the case of strongly convex cost functions the updates characterising the R-ADMM can be written as the sum of a linear term and of a *perturbation term* which in practice is a little-o w.r.t. the linear evolution of the $\mathbf{x}$. However this result in general holds only in a neighbourhood of the optimum, but in the case of quadratic cost functions the result holds globally since the second order expansion of the costs around the optimum is exact.

The following Lemma will provide a useful result for the derivation of an upper bound on the distance of the algorithms' iterate from the optimum.

Lemma 3.7. *Let A be defined as in Lemma 3.5, then $\left\|DA^k\right\| \leq a|\lambda|^k$ with λ the eigenvalue of A inside the unitary circle with maximum absolute value.*

Proof. Suppose that A is diagonalisable by [68, p. 517] it is possible to apply the *spectral decomposition*

$$A = \lambda_1 G_1 + \ldots + \lambda_r G_r \tag{3.24}$$

where $\lambda_1, \dots, \lambda_r$ are the distinct eigenvalues of A, and the G_i matrices are such that

- they are projection matrices orthogonal w.r.t. each other, *i.e.*

$$G_i G_j = \begin{cases} G_i & \text{if } i = j \\ 0 & \text{if } i \neq j, \end{cases}$$

- $(A - \lambda_i I)G_i = G_i(A - \lambda_i I) = 0$.

Therefore the following chain of equalities holds

$$\begin{aligned} A^k &= \lambda_1^k G_1^k + \ldots + \lambda_r^k G_r^k \\ &= \lambda_1^k G_1 + \ldots + \lambda_r^k G_r \\ &= \sum_{i:|\lambda_i|=1} \lambda_i^k G_i + \sum_{i:|\lambda_i|<1} \lambda_i^k G_i \end{aligned} \tag{3.25}$$

where the first and second follow from $G_i G_j = 0$, $i \neq j$, and $G_i^2 = G_i$ respectively.

Consider now $\left\|DA^k\right\|$, since Proposition 3.4 guarantees the convergence of the algorithm, then it must be that $\left\|DA^k\right\| \to 0$ for $k \to \infty$. But by Equation (3.25) follows

$$\begin{aligned} \left\|DA^k\right\| &= \left\| D \sum_{i:|\lambda_i|=1} \lambda_i^k G_i + D \sum_{i:|\lambda_i|<1} \lambda_i^k G_i \right\| \\ &= \left\| \sum_{i:|\lambda_i|=1} \lambda_i^k DG_i \right\| + \left\| \sum_{i:|\lambda_i|<1} \lambda_i^k DG_i \right\| \\ &= \sum_{i:|\lambda_i|=1} \|DG_i\| + \sum_{i:|\lambda_i|<1} |\lambda_i^k| \, \|DG_i\| \end{aligned}$$

where the second equality holds for the properties of the spectral projectors G_i and the third for the properties of the norm and the fact that the eigenvalues in the first sum have unitary absolute value.
Now, since the right-most sum converges to zero with a rate that is smaller than or equal[3] to the absolute value of the largest eigenvalue of A that is inside the unitary circle, then for $k \to \infty$ it must be that $\sum_{i:|\lambda_i|=1} \|DG_i\| \to 0$, which can only be possible if the columns of G_i are in the kernel of D. Therefore the value $\left\|DA^k\right\|$ converges to zero with a rate dictated by

$$\lambda_M = \max_i \left\{ |\lambda_i| \;\middle|\; \lambda_i \in \Lambda(A) \text{ s.t. } |\lambda_i| < 1 \right\}. \tag{3.26}$$

Suppose now that A is not diagonalisable, then the more general spectral decomposition [68, p. 603-604] can be applied, which states that

$$A = \sum_{i=1}^{r} (\lambda_i G_i + N_i)$$

with the spectral projectors G_i and matrices N_i that satisfy

- the G_i matrices are projection matrices orthogonal w.r.t. each other (as above),
- $(A - \lambda_i I)^{k_i} G_i = G_i (A - \lambda_i I)^{k_i} = 0$ where k_i is the index of the eigenvalue λ_i,
- $N_i = (A - \lambda_i I) G_i = G_i (A - \lambda_i I)$ and it is nilpotent of index k_i.

Consider now the product $(\lambda_i G_i + N_i)(\lambda_j G_j + N_j)$ with $i \neq j$, by the properties of the spectral projectors and the fact that by definition $N_i N_j = 0$ then it follows that it is zero. Which implies that

$$A^k = \sum_{i=1}^{r} (\lambda_i G_i + N_i)^k.$$

[3] Since some modes could not be excited.

Expanding $(\lambda_i G_i + N_i)^k$ and assuming that $k > k_i$, it holds

$$\begin{aligned}(\lambda_i G_i + N_i)^k &= \lambda_i^k G_i + \sum_{j=1}^{k-1} \binom{k}{j} \lambda_i^{k-j} G_i N_i^j + N_i^k \\ &= \lambda_i^k G_i + \sum_{j=1}^{k_i-1} \binom{k}{j} \lambda_i^{k-j} G_i N_i^j\end{aligned}$$

where the fact $G_i^2 = G_i$ and the nilpotency of N_i were used.

Using the results derived above it is now possible to compute $\left\|DA^k\right\|$ as follows

$$\begin{aligned}\left\|DA^k\right\| &= \left\| D \sum_{i=1}^{r} \left(\lambda_i^k G_i + \sum_{j=1}^{k_i-1} \binom{k}{j} \lambda_i^{k-j} G_i N_i^j \right) \right\| \\ &= \sum_{i=1}^{r} \left\| \lambda_i^k DG_i + \sum_{j=1}^{k_i-1} \binom{k}{j} \lambda_i^{k-j} DG_i N_i^j \right\| \\ &= \sum_{i=1}^{r} |\lambda_i^k| \left\| DG_i \left(I + \sum_{j=1}^{k_i-1} \binom{k}{j} \lambda_i^{-j} N_i^j \right) \right\|\end{aligned}$$

where the fact $G_i G_j = 0$ for $j \neq i$ was used.
Similarly to the diagonalisable case, the terms that correspond to eigenvalues with $|\lambda_i| < 1$ converge to zero with a rate smaller than or equal to λ_M as defined above. Therefore for $\left\|DA^k\right\|$ to be zero it must be that $DG_i = 0$ which implies that the columns of G_i are in the kernel of D.

□

Finally it is possible to prove the linear convergence of the R-ADMM.

Proposition 3.8. *Assume the cost functions f_i are strongly convex. Then, there exists a neighbourhood $\mathcal{N}_{\mathbf{x}^*}$ of the optimal point $\mathbf{x}^*$ such that, if $\mathbf{x}(0) \in \mathcal{N}_{\mathbf{x}^*}$, then Algorithm 1 converges exponentially fast, i.e.,*

$$\|x_i(k) - x^*\| \leq C\lambda^{-k} \|x_i(0) - x^*\|, \qquad \forall i \in \mathcal{V},$$

for suitable constants $C > 0$ and $0 \leq \lambda < 1$.

Proof. The proof is divided in two parts, the first one concerned with the derivation of a fundamental inequality on the norm of the error $\mathbf{x}(k) - \mathbf{x}^*$, and the second that proves exponential convergence with an argument similar to that employed in [58].

Error inequality By iterating the update equation (3.18) the following formula is obtained

$$\mathbf{z}(k) = A^k \mathbf{z}(0) - \sum_{l=0}^{k-1} A^{k-l-1} \left(FE(\mathbf{x}(l) - \mathbf{x}^*) - c \right). \tag{3.27}$$

Note that this equation is satisfied by any $\mathbf{z}^* \in \mathrm{fix}(T_{PR})$ with $\mathbf{z}(0) = \mathbf{z}^*$ and $\mathbf{x}(l) = \mathbf{x}^*$ for any l.
Since the components of prox_h are non-expansive so is it, and therefore the following chain

of inequalities holds

$$\begin{aligned}\|\mathbf{x}(k)-\mathbf{x}^*\| &= \left\|\rho N\left(\operatorname{prox}_h\left(\frac{N^{-1}}{\rho}D\mathbf{z}(k)\right)-\operatorname{prox}_h\left(\frac{N^{-1}}{\rho}D\mathbf{z}^*\right)\right)\right\| \\ &\leq \|D(\mathbf{z}(k)-\mathbf{z}^*)\| \\ &\leq \left\|DA^k(\mathbf{z}(0)-\mathbf{z}^*)\right\| + \sum_{l=0}^{k-1}\left\|DA^{k-l-1}FE(\mathbf{x}(l)-\mathbf{x}^*)\right\| \\ &\leq \left\|DA^k\right\|\|\mathbf{z}(0)-\mathbf{z}^*\| + \sum_{l=0}^{k-1}\left\|DA^{k-l-1}\right\|\|F\|\,\delta\,\|\mathbf{x}(l)-\mathbf{x}^*\|\end{aligned}$$

Note that for the second and third inequality the properties of the norm were used and the fact that by definition of E it is possible to find $\delta \in (0,1)$ such that $\|E(\mathbf{x}(l)-\mathbf{x}^*)\| \leq \delta\,\|\mathbf{x}(l)-\mathbf{x}^*)\|$.

Applying Lemma 3.7 it is possible to rewrite the upper bound for the error as

$$\|\mathbf{x}(k)-\mathbf{x}^*\| \leq \bar{c}\lambda^k + \bar{c}\delta\sum_{l=0}^{k-1}\lambda^{k-l-1}\|\mathbf{x}(l)-\mathbf{x}^*\| \tag{3.28}$$

where it was defined $\bar{c} = a\max\{\|\mathbf{z}(0)-\mathbf{z}^*\|, \|F\|\}$.

Linear convergence The aim now is to exploit inequality (3.28) in order to prove the linear convergence, that is in particular, to show that

$$\forall\epsilon>0 \quad \sup_k(\lambda+\epsilon)^{-k}\|\mathbf{x}(k)-\mathbf{x}^*\| < \infty. \tag{3.29}$$

Defining the function $w(k) = (\lambda+\epsilon)^{-k}\|\mathbf{x}(k)-\mathbf{x}^*\|$ proving that (3.29) holds can be done by showing that $w(k) < B\ \forall k$ for some $B \in \mathbb{R}_+$.
Assuming $\|\mathbf{x}(0)-\mathbf{x}^*\| < B$, then by definition $w(0) < B$. By induction suppose that $w(l) < B$ holds for any $l < k$ and prove that indeed $w(k) < B$ as well.

By inequality (3.28) it holds

$$w(k) \leq \bar{c}\left(\frac{\lambda}{\lambda+\epsilon}\right)^k + \bar{c}\delta\sum_{l=0}^{k-1}\frac{\lambda^{k-l-1}}{(\lambda+\epsilon)^k}\|\mathbf{x}(l)-\mathbf{x}^*\|.$$

And given that

$$\frac{\lambda^{k-l-1}}{(\lambda+\epsilon)^k} = \frac{1}{\lambda}\left(\frac{\lambda}{\lambda+\epsilon}\right)^{k-l}\frac{1}{(\lambda+\epsilon)^l}$$

it follows

$$\begin{aligned}w(k) &\leq \bar{c}\left(\frac{\lambda}{\lambda+\epsilon}\right)^k + \frac{\bar{c}\delta}{\lambda}\sum_{l=0}^{k-1}\left(\frac{\lambda}{\lambda+\epsilon}\right)^{k-l}w(l) \\ &< \bar{c} + \frac{\bar{c}\delta B}{\lambda}\sum_{l=0}^{k-1}\left(\frac{\lambda}{\lambda+\epsilon}\right)^{k-l}\end{aligned}$$

where the last inequality holds because $\lambda/(\lambda+\epsilon) < 1$ and by the inductive hypothesis. Finally using a new index $m = k - l$ for the sum and extending it to infinity yields

$$w(k) < \bar{c} + \frac{\bar{c}\delta B}{\lambda} \sum_{m=0}^{\infty} \left(\frac{\lambda}{\lambda+\epsilon} \right)^{k-l} = \bar{c} + \frac{\bar{c}\delta B}{\lambda} \frac{\lambda+\epsilon}{\epsilon}. \tag{3.30}$$

Up until this point the two parameters δ and B have not yet been fixed, therefore these two degrees of freedom can be exploited to impose that $w(k) < B$. By inequality (3.30) this is true if, for instance, the right-hand side is equal to B, that is if

$$B = \frac{\bar{c}\lambda\epsilon}{\lambda\epsilon - \bar{c}(\lambda+\epsilon)\delta}.$$

However it must be $B > 0$, which can be guaranteed by choosing $\delta < \lambda\epsilon/(\bar{c}(\lambda+\epsilon))$.
Notice that the condition for δ does not respect the constraint $\delta < 1$ if $\bar{c} < \epsilon$; but if this is the case the definition $\bar{c} = \max\{\epsilon, a \left\| \mathbf{z}(0) - \mathbf{z}^* \right\|, a \left\| F \right\|\}$ can be used and obtain the same results.

Therefore the linear convergence of the algorithm has been proved in a neighbourhood of radius B from the optimum of the problem. □

Remark 3.9. Many works have proved the linear convergence of the ADMM with strongly convex costs – see Section 1.3.2. The purpose of this Section was to introduce useful notation and results that will be used in the proof of the linear convergence for the R-ADMM in a lossy and asynchronous scenario, on which Chapter 4 focuses.

3.3 Edge- and Node-Based Optimisation

The formulation of relaxed ADMM for distributed optimisation problems is not the only one possible, and presents both pros and cons. This formulation belongs to the class of *edge-based* optimisation algorithms, because each node updates and stores a local variable and a set of auxiliary variables one for each of its neighbours.
However, the ADMM can also be reformulated as a *node-based* optimisation algorithm, in which each node does not need to store and update an auxiliary variable for each of its neighbours. The following Algorithm 3 introduced in [93] presents an example of a node-based implementation of the ADMM. Another implementation is presented in [70], which however requires the costs to be strongly convex and a colouring of the graph to be available and agreed upon by all nodes.

By inspection of the node-based ADMM formulation it is clear that the number of variables that each node needs to store is always constant (equal to two), regardless of the number of neighbours that the node has. On the other hand, recalling Table 3.1, the number of variables stored by the R-ADMM is $|\mathcal{N}_i| + 1$, excluding temporary variables. Therefore in the worst-case scenario the storage requirement is $O(1)$ and $O(N)$ respectively. However, it is possible to reformulate the R-ADMM in order to have an $O(1)$ storage requirement. Indeed consider the update equations

$$\begin{aligned} z_{ij}(k+1) &= (1-\alpha_k) z_{ij}(k) - \alpha_k z_{ji}(k) + 2\alpha_k \rho x_i(k) \\ z_{ji}(k+1) &= (1-\alpha_k) z_{ji}(k) - \alpha_k z_{ij}(k) + 2\alpha_k \rho x_j(k) \end{aligned}$$

Algorithm 3 Node-based implementation of the ADMM [93].

Input: penalty ρ, termination condition K.
Initialise: $x_i(0), v_i(0) \leftarrow 0$ for each node i.

1: $k \leftarrow 0$
2: **while** $k < K$ each agent i **do**
3: update $x_i(k+1)$ by solving

$$\bar{\nabla} f_i(x_i(k+1)) + v_i(k) + 2\rho|\mathcal{N}_i|x_i(k+1) - \rho\left(|\mathcal{N}_i|x_i(k) + \sum_{j\in\mathcal{N}_i} x_j(k)\right) = 0$$

4: broadcast $x_i(k+1)$ to all neighbours $j \in \mathcal{N}_i$
5: gather $x_j(k+1)$ from each neighbour j
6: compute $v_i(k+1)$ with

$$v_i(k+1) = v_i(k) + \rho\left(|\mathcal{N}_i|x_i(k+1) + \sum_{j\in\mathcal{N}_i} x_j(k+1)\right)$$

7: $k \leftarrow k+1$
8: **end while**

of which the first is carried out by node i and the second by neighbour j.
Summing the updates for $\{z_{ij}\}_{j\in\mathcal{N}_i}$ and the updates for $\{z_{ji}\}_{j\in\mathcal{N}_i}$ yields

$$\sum_{j\in\mathcal{N}_i} z_{ij}(k+1) = (1-\alpha_k)\sum_{j\in\mathcal{N}_i} z_{ij}(k) - \alpha_k \sum_{j\in\mathcal{N}_i} z_{ji}(k) + 2\alpha_k\rho|\mathcal{N}_i|x_i(k)$$
$$\sum_{j\in\mathcal{N}_i} z_{ji}(k+1) = (1-\alpha_k)\sum_{j\in\mathcal{N}_i} z_{ji}(k) - \alpha_k \sum_{j\in\mathcal{N}_i} z_{ij}(k) + 2\alpha_k\rho\sum_{j\in\mathcal{N}_i} x_j(k)$$

and defining $v_i(k) = \sum_{j\in\mathcal{N}_i} z_{ij}(k)$ and $w_i(k) = \sum_{j\in\mathcal{N}_i} z_{ji}(k)$ they become

$$v_i(k+1) = (1-\alpha_k)v_i(k) - \alpha_k w_i(k) + 2\alpha_k\rho|\mathcal{N}_i|x_i(k) \tag{3.31}$$
$$w_i(k+1) = (1-\alpha_k)w_i(k) - \alpha_k v_i(k) + 2\alpha_k\rho\sum_{j\in\mathcal{N}_i} x_j(k). \tag{3.32}$$

Note that in order to compute (3.32) node i needs only the state variables of its neighbours $\{x_j(k)\}_{j\in\mathcal{N}_i}$, and moreover that in (3.5) the only necessary information to find $x_i(k)$ is precisely $w_i(k)$. Therefore if each node stores both v_i and w_i, updating them with its own local state and the states of its neighbours, them the storage requirement is $O(1)$.

The computational complexity instead is always the same for both edge- and node-based methods, because they need to solve a single minimisation problem at each time instant and then perform updates that are $O(1)$. Moreover, the data that each node needs to exchange with its neighbours in both formulations is the same, that is, one packet containing a vector in $\mathbb{R}^n$ to each neighbour, but for Algorithm 3 a broadcast transmission can be used.

Therefore the edge-based formulation is worse than the node-based formulation from a storage point of view only, and they are equal for what concerns the computational complexity and the number of transmissions required.
However the choice of using the R-ADMM is motivated by the fact that it shares a clear link with operator theory and splitting methods. Indeed these mathematical tools will be important in the following Chapter to develop a robust and asynchronous version of the R-ADMM

for which convergence guarantees can be given.

Chapter 4

Randomised Alternating Direction Method of Multipliers

The aim of the present Chapter is to derive a new formulation of the ADMM described above that is robust to packet losses and can be implemented asynchronously. Note that for simplicity hereafter ADMM will refer to the *relaxed* ADMM.

4.1 ADMM in Lossy Scenarios

In order to give a clearer and more intuitive derivation of the *randomised ADMM* the current Section will focus only on the problem of making the ADMM robust to packet losses. This Section therefore introduces the *robust ADMM* (r-ADMM), which can be seen as a particular case of the more general randomised ADMM.

4.1.1 Robust ADMM

The ADMM described in Algorithm 1 in the previous Chapter consists of the synchronous iteration of the following updates

$$x_i(k) = \arg\min_{x_i} \left\{ f_i(x_i) - \sum_{j \in \mathcal{N}_i} z_{ji}^\top(k) x_i + \frac{\rho|\mathcal{N}_i|}{2} \|x_i\|^2 \right\}$$
$$z_{ji}(k+1) = (1-\alpha_k) z_{ji}(k) + \alpha_k q_{j\to i}$$

where

$$q_{j\to i} = -z_{ij}(k) + 2\rho x_j(k)$$

is computed and sent to node i by neighbour j.
Therefore if the loss of a packet transmitted from node j to node i occurs, the variable $q_{j\to i}$ is unavailable to i, which as a consequence cannot update z_{ji}. The following Assumption describes the failure mode of the communication network.

Assumption 4.1. During any iteration of Algorithm 1 the communication from node i to node j can be lost with some probability p.

In order to describe the possible communication failures more rigorously, a binary random variable L_{ij} is associated to each edge $(i,j) \in \mathcal{E}$ of the graph, such that

$$\mathbb{P}[L_{ij} = 1] = p, \qquad \mathbb{P}[L_{ij} = 0] = 1 - p$$

The random variables $\{L_{ij}\}_{(i,j)\in\mathcal{E}}$ are assumed to be independent from each other, that is, they are i.i.d.

With this formalism, if the packet sent from i to j at the k-th iteration of Algorithm 1 is lost then $L_{ij}(k) = 1$, with $L_{ij}(k)$ realisation of L_{ij} at time k, otherwise $L_{ij}(k) = 0$.

The policy that is adopted in order to deal with communication failures is therefore that of updating variable z_{ji} if and only if the transmission $q_{j\to i}$ was received.
Note that this is not the only possible policy, indeed a naïve approach could be that of using the last information available to compute the update. However for this case there is no theoretical guarantee of convergence, while for the first policy there is, as will be explained below. See Section 6.2.3 for a comparison of the two policies carried out in simulations.

Algorithm 1 can now be rewritten as Algorithm 4 in order to handle possible packet losses.

Algorithm 4 Robust distributed ADMM.

Input: step-sizes $\{\alpha_k\}_{k\in\mathbb{N}}$, penalty ρ, termination condition K.
Initialise: $x_i(0)$ and $z_{ji}(0)$ for each node i and neighbour $j \in \mathcal{N}_i$.

$k \leftarrow 0$
while $k < K$ each agent i **do**
  compute $x_i(k)$ according to (3.5)

$$x_i(k) = \arg\min_{x_i} \left\{ f_i(x_i) - \sum_{j\in\mathcal{N}_i} z_{ji}^\top(k) x_i + \frac{\rho|\mathcal{N}_i|}{2} \|x_i\|^2 \right\}$$

  for all $j \in \mathcal{N}_i$ compute $q_{i\to j}$ as in (3.14)

$$q_{i\to j} = -z_{ji}(k) + 2\rho x_i(k)$$

  transmit $q_{i\to j}$ to node j
  for $j \in \mathcal{N}_i$ **if** $q_{j\to i}$ is received **do**
    update z_{ji} as in (3.15)

$$z_{ji}(k+1) = (1-\alpha_k) z_{ji}(k) + \alpha_k q_{j\to i}$$

  end for
  $k \leftarrow k + 1$
end while

The update (3.15) can therefore be rewritten as follows in order to account for the communication failures

$$z_{ji}(k+1) = L_{ji}(k) z_{ji}(k) + (1 - L_{ji}(k))((1-\alpha_k) z_{ji}(k) + \alpha_k q_{j\to i}). \tag{4.1}$$

However note that this is only a mathematical formalism, if the packet is lost node i actually does not perform any action, simply leaving the stored z_{ji} unchanged.
A matricial formulation of Equation (4.1) can be given as

$$\mathbf{z}(k+1) = L(k)\mathbf{z}(k) + (I - L(k)) T_{PR} \mathbf{z}(k) \tag{4.2}$$

by defining the random matrix

$$L = \mathrm{diag}\{L_{ij},\ \forall i \in \mathcal{V},\ j \in \mathcal{N}_i\}$$

and its realisation at each time k as $L(k)$. Note that $((1-\alpha_k)P\mathbf{z}(k) + \alpha_k \mathbf{q}$ with the vector $\mathbf{q} = [q_{i\to j}^\top,\ \forall i \in \mathcal{V},\ j \in \mathcal{N}_i]^\top$.

4.1.2 Convergence

The following Proposition guarantees the convergence of the r-ADMM.

Proposition 4.2. *Consider Algorithm 4 working under the scenario described in Assumption 4.1. Let $\{\alpha_k\}_{k\in\mathbb{N}}$ be a sequence of step-sizes such that*

$$0 < \liminf_k \alpha_k \leq \limsup_k \alpha_k < 1 \tag{4.3}$$

and $\rho > 0$. Then, for any initial conditions, the trajectories $k \to x_i(k)$, $i \in \mathcal{V}$, generated by Algorithm 4, converge almost surely to the optimal solution of (3.4)*, i.e.,*

$$\lim_{k\to\infty} x_i(k) = x^*, \qquad \forall i \in \mathcal{V},$$

with probability one, for any $x_i(0)$ and $z_{ji}(0)$, $j \in \mathcal{N}_i$.

Proof. The first step to proving Proposition 4.2 is to introduce the *stochastic Krasnosel'skiĭ-Mann iteration* described in [57, 7]. It is then possible to show that the r-ADMM conforms to such iterate and therefore enjoys the same convergence properties.

Recall from Chapter 2.2 that the Krasnosel'skiĭ-Mann iteration was defined for a non-expansive operator $T : \mathcal{X} \to \mathcal{X}$ as

$$z(k+1) = (1-\alpha_k)z(k) + \alpha_k T z(k) \tag{4.4}$$

for a sequence $\{\alpha_k\}_{k\in\mathbb{N}}$ of step-sizes in $(0, 1]$. This iterate updates at each time instant all the components of z. However this is not the case of the r-ADMM, which possibly updates only a subset of co-ordinates at each instant.
Consider the following stochastic operator $T^{(\mu)} : \mathcal{X} \to \mathcal{X}$, where μ is a subset of the co-ordinates $\mathcal{M} = \{1, \dots, M\}$ of z, whose components are defined such that

$$T_i^{(\mu)} z = \begin{cases} T_i z & \text{if } i \in \mu, \\ z_i & \text{if } i \notin \mu. \end{cases}$$

Therefore the i-th component is updated if and only if it is included in the subset μ of selected co-ordinates.
The stochastic operator $T^{(\mu)}$ substitutes now operator T in the Krasnosel'skiĭ-Mann iteration, resulting in

$$z(k+1) = (1-\alpha_k)z(k) + \alpha_k T^{(\mu_{k+1})} z(k) \tag{4.5}$$

where μ_{k+1} indicates that in general the subset of co-ordinates to be updated can vary at each time instant. Indeed given the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ the sequence $\{\mu_k\}_{k\in\mathbb{N}}$ is a random i.i.d. sequence with $\mu_k : \Omega \to 2^{\mathcal{M}}$.

The stochastic iteration (4.5) satisfies the following convergence result, adapted from [7] using the fact that a non-expansive operator is 1-averaged. For the sake of completeness, Appendix B reports the more general result along with the proof and some comments.

Proposition 4.3 ([7, Theorem 3])**.** *Let T be a non-expansive operator with $\mathrm{fix}(T) \neq \emptyset$. Assume that for all k the sequence of step-sizes $\{\alpha_k\}_{k\in\mathbb{N}}$ is such that*

$$0 < \liminf_k \alpha_k \leq \limsup_k \alpha_k < 1.$$

Let $\{\mu_k\}_{k\in\mathbb{N}}$ be a random i.i.d. sequence on $2^{\mathcal{M}}$ such that the following condition holds

$$\forall i \in \mathcal{M},\ \exists \mu \in 2^{\mathcal{M}} \ \text{ s.t. } \ i \in \mu \text{ and } \mathbb{P}[\mu_0 = \mu] > 0. \tag{4.6}$$

Then for any deterministic initial condition $z(0)$ *the stochastic Krasnosel'skiĭ-Mann iteration* (4.5) *converges almost surely to a random variable with support in* $\mathrm{fix}(T)$.

Remark 4.4. The meaning of condition (4.6) is that the random sequence $\{\mu_k\}_{k\in\mathbb{N}}$ must be such that, for each co-ordinate $i \in \mathcal{M}$, there is at least one subset μ with $i \in \mu$ which has a non-zero probability of being selected as the first subset of co-ordinates to be updated, that is, as μ_0. In other words, there must be a non-zero probability that each co-ordinate be updated at the first iteration. This condition is rather technical but not very restrictive, and is required to avoid a possible division by zero in the proof, see Appendix B.

For what concerns now the r-ADMM, recall that the update for z can be compactly written as

$$T_{PR}^{(\mu_{k+1})}\mathbf{z}(k) = L(k)\mathbf{z}(k) + (I - L(k))T_{PR}\mathbf{z}(k) \tag{4.7}$$

where $L(k)$ is a realisation of the diagonal random matrix with elements the packet loss random variables L_{ij}. This operator conforms to the definition used in the stochastic Krasnosel'skiĭ-Mann iteration, since T_{PR} is non-expansive and at each time instant only a subset of the co-ordinates is updated.
Notice that in the derivation above the α-averaging of the Krasnosel'skiĭ-Mann iteration is applied first and the stochastic co-ordinate update selection second, while in the stochastic (4.5) they are applied in the inverse order. However it is easy to see that the are equivalent.
Therefore the convergence result 4.3 holds, which means that the sequence $\{\mathbf{z}(k)\}_{k\in\mathbb{N}}$ converges almost surely to a fixed point of T_{PR}. Moreover, the dual optimum is reached and can be computed as $\mathbf{w}^* = \mathrm{prox}_{\rho d_g}(\mathbf{z}^*)$. But since the duality gap is zero, then also the primal optimum is reached, which means that the sequence $\{\mathbf{x}(k)\}_{k\in\mathbb{N}}$ converges to $\mathbf{x}^*$. □

4.2 Asynchronous and Robust ADMM

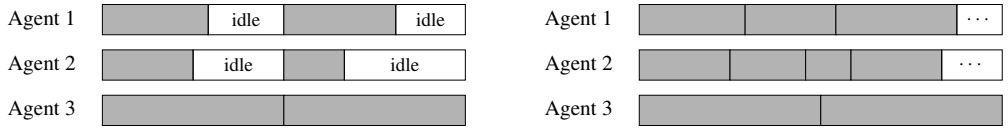
The present Section will extend the formulation described above for the r-ADMM to the case of asynchronous and potentially lossy communications. This more general algorithm will be called the *robust and asynchronous ADMM*, ra-ADMM for short.

The results presented in the previous Section assume that the update and transmission phases happen all at the same time and are perfectly synchronised. This is often very difficult to guarantee in a real-life scenario, or might lead to inefficient execution of the algorithms, hence the interest for an asynchronous version of the r-ADMM. A technical difficulty for implementing synchronous distributed algorithms is that all agents must agree on a global clock that regulates the update and transmission phases, which in practice might require that a central node be present to regulate the clocks of each agent, defying some of the advantages of distributed set-ups. The following Remark instead presents an example of application in which asynchrony can improve the performance of an algorithm.

Remark 4.5. As mentioned in the introduction, one of the possible use cases of the ADMM is in a collaborative effort to the solution of a Machine Learning problem, involving many different and independent agents, that is, computers, connected in a network. Ensuring that the computers all act synchronously, if at all possible, would require a loss of CPU cycles as the faster computers, once terminated the update phase, would have to wait for the slower to catch up. This situation is depicted in Figure 4.1.

Therefore Assumption 4.1 will be now complemented by the following Assumption on the asynchrony of the node updates.

Assumption 4.6. During any iteration of Algorithm 1, any node i performs an update, that is, computes $x_i(k)$, with some probability q.



(A) Synchronous update. (B) Asynchronous update.

FIGURE 4.1: Representation of the synchronous and asynchronous update criteria.

To the family of random variables L_{ij} introduced in Assumption 4.1 the following family of independent binary variables $u_i(k)$, $k = 1, \ldots, N$ is added, one for each node, such that

$$\mathbb{P}[\text{node } i \text{ updates}] = \mathbb{P}[u_i = 1] = q.$$

Furthermore the two sets of variables are assumed to be independent from each other.

Remark 4.7. Note that all the results that follow, as well as those for the r-ADMM, can be extended to the case of different packet loss probabilities and update probabilities for each edge and node, respectively. Assumptions 4.1 and 4.6 have been chosen only for the sake of simplicity in the discussion.

With regard to Algorithm 1 and the discussion in Section 4.1, clearly node i is able to update z_{ji} if and only if node j performed an update and then successfully sent the variable $q_{j\to i}$ to it.
Therefore a third set of random variables can be introduced that account for both the update and packet loss events $\beta_{ij}(k)$, $k = 0, 1, 2, \ldots$, $i \in \mathcal{V}$, $j \in \mathcal{N}_i$, such that

$$\mathbb{P}[\beta_{ij} = 1] = \mathbb{P}[u_j = 1]\,\mathbb{P}[L_{ji} = 0] = q(1-p) = r.$$

Note that this set of variables are identically distributed *but not* independent, since the variables $\{\beta_{ij}\}_{j\in\mathcal{N}_i}$ all depend on the value of u_i.
A simple expression for the β_{ij} random variables is

$$\beta_{ij} = 1 - \max\{L_{ji}, 1 - u_j\}.$$

Algorithm 5 describes now the implementation of the ra-ADMM.
In this case, at k-th iteration node i updates x_i as in (3.5) if it scheduled to do so, that is if $u(k) = 1$. Then, for $j \in \mathcal{N}_i$, it computes $q_{i\to j}$ as in (3.14) and transmits it to node j. If node j receives $q_{i\to j}$, then it updates z_{ij} as $z_{ij}(k+1) = (1-\alpha_k)z_{ij}(k) + \alpha_k q_{i\to j}$, otherwise z_{ij} remains unchanged, i.e., $z_{ij}(k+1) = z_{ij}(k)$. This last step can be compactly described as

$$z_{ij}(k+1) = (1-\beta_{ij}(k))\, z_{ij}(k) + \beta_{ij}(k)\, ((1-\alpha_k)z_{ij}(k) + \alpha_k q_{i\to j})$$

which shares the same structure of (4.1).
Therefore the convergence result Proposition 4.2 extends to the ra-ADMM, since in the proof the co-ordinate selection can be performed over any possible probability space.

4.3 Linear Convergence Rate

In this Section the linear convergence of the ra-ADMM is proved in the case of strongly convex local cost functions in problem (3.4). Before stating the main result, two useful Lemmas are presented and proved.

Algorithm 5 Robust and asynchronous distributed ADMM.

Input: step-sizes $\{\alpha_k\}_{k\in\mathbb{N}}$, penalty ρ, termination condition K.
Initialise: $x_i(0)$ and $z_{ji}(0)$ for each node i and neighbour $j \in \mathcal{N}_i$.

$k \leftarrow 0$
while $k < K$ each agent i **do**
  if update scheduled **then**
    compute $x_i(k)$ according to (3.5)

$$x_i(k) = \arg\min_{x_i} \left\{ f_i(x_i) - \sum_{j\in\mathcal{N}_i} z_{ji}^\top(k) x_i + \frac{\rho|\mathcal{N}_i|}{2} \|x_i\|^2 \right\}$$

    for all $j \in \mathcal{N}_i$ compute $q_{i\to j}$ as in (3.14)

$$q_{i\to j} = -z_{ji}(k) + 2\rho x_i(k)$$

    transmit $q_{i\to j}$ to node j
  end if
  for $j \in \mathcal{N}_i$ **if** $q_{j\to i}$ is received **do** ▷ *i.e.* j updated and the packet was not lost
    update z_{ji} as in (3.15)

$$z_{ji}(k+1) = (1-\alpha_k) z_{ji}(k) + \alpha_k q_{j\to i}$$

  end for
  $k \leftarrow k+1$
end while

Lemma 4.8. *Assume that the local costs are strongly convex functions. The update of Lemma 3.5 for the* $\mathbf{z}$ *can be written for the ra-ADMM as*

$$\mathbf{z}(k+1) = A(k)\mathbf{z}(k) - B(k)\left(FE(\mathbf{x}(k) - \mathbf{x}^*) - c\right) \tag{4.8}$$

where $A(k) = I - B(k) + B(k)A$ *and* $B(k)$ *a realisation of the random matrix* $B = \mathrm{diag}\{\beta_{ij},\ i \in \mathcal{V},\ j \in \mathcal{N}_i\}$.
Moreover the following formula holds

$$\begin{aligned}\mathbf{z}(k) = Q(k)\mathbf{z}(0) - B(k-1)\left(FE(\mathbf{x}(k-1) - \mathbf{x}^*) - c\right) + \\ - \sum_{l=0}^{k-2} A(k-1)\cdots A(l+1)B(l)\left(FE(\mathbf{x}(l) - \mathbf{x}^*) - c\right)\end{aligned} \tag{4.9}$$

with $Q(k) = A(k-1)\cdots A(1)$.

Proof. The first equation derives from the application of the random co-ordinate selection to the update of Lemma 3.5, that is from

$$\mathbf{z}(k+1) = (I - B(k))\mathbf{z}(k) + B(k)\left(A\mathbf{z}(k) - FE(\mathbf{x}(k) - \mathbf{x}^*) + c\right),$$

while the second results by iterating the former. □

Lemma 4.9. *Let the eigenspace of* A *relative to the unitary eigenvalue(s) be*

$$U_1 = \langle v_1, \ldots, v_m \rangle,$$

with $m \geq 1$. Then it is

$$Q_\infty = \lim_{k\to\infty} Q(k) = v_1\rho_1^\top + \ldots + v_m\rho_m^\top$$

for ρ_i stochastic random vectors.

Proof. By the convergence theorem it is known that the $\mathbf{z}$ vector converges almost surely to a fixed point of the operator T_{PR}. Therefore defining $\lim_{k\to\infty} \mathbf{z}(k) = \mathbf{z}_\infty$ it holds $\mathbf{z}_\infty = A\mathbf{z}_\infty + c$ and also that $\mathbf{z}^* = A\mathbf{z}^* + c$ for $\mathbf{z}^* \in \mathrm{fix}(T_{PR})$. Subtracting the two equations it follows $\mathbf{z}_\infty - \mathbf{z}^* = A(\mathbf{z}_\infty - \mathbf{z}^*)$, which means that $\mathbf{z}_\infty - \mathbf{z}^*$ must be in the eigenspace of A relative to the unitary eigenvalue.
Moreover by equation (4.9) it is

$$\mathbf{z}_\infty - \mathbf{z}^* = \lim_{k\to\infty} (\mathbf{z}(k) - \mathbf{z}^*) = \lim_{k\to\infty} Q(k)(\mathbf{z}(0) - \mathbf{z}^*)$$

which means that $Q_\infty(\mathbf{z}(0) - \mathbf{z}^*)$ must belong to the eigenspace U_1 as well.

Suppose now that $U_1 = \langle v \rangle$, therefore it can be written $Q_\infty(\mathbf{z}(0) - \mathbf{z}^*) = \gamma v$ for γ a random variable. This is satisfied if Q_∞ can be written as $v\rho^\top$ where ρ is a stochastic random vector such that $\gamma = \rho^\top(\mathbf{z}(0) - \mathbf{z}^*)$.
The same result holds in the case that $m > 1$, indeed it is $Q_\infty(\mathbf{z}(0) - \mathbf{z}^*) = \gamma_1 v_1 + \ldots + \gamma_m v_m$ which is satisfied if $Q_\infty = v_1\rho_1^\top + \ldots + v_m\rho_m^\top$ with $\gamma_i = \rho_i^\top(\mathbf{z}(0) - \mathbf{z}^*)$. □

Lemma 4.10. *Let $U_1 = \langle v_1, \ldots, v_m \rangle$ be the eigenspace of A relative to the unitary eigenvalue. Then it holds $v_i \in \ker(D)$ for all $i = 1, \ldots, m$.*

Proof. Recall that by the strong convexity of the cost functions, the optimum $\mathbf{x}^*$ is unique. Moreover, given a fixed point $\mathbf{z}^* \in \mathrm{fix}(T_{PR})$ it satisfies $\mathbf{z}^* = A\mathbf{z}^* + c$, having set $\mathbf{x}(l) = \mathbf{x}^*$ in (3.18). But also any vector $\mathbf{z}^* + \gamma v_i$ with $\gamma \in \mathbb{R}$ and $v_i \in U_1$ satisfies the equation $\mathbf{z}^* + \gamma v_i = A(\mathbf{z}^* + \gamma v_i) + c$ since $Av_i = v_i$.
Therefore by the uniqueness of the optimum and using (3.19) it must hold

$$H^{-1}D\mathbf{z}^* - H^{-1}b = \mathbf{x}^* = H^{-1}D(\mathbf{z}^* + \gamma v_i) - H^{-1}b.$$

This is true if and only if $\gamma H^{-1}Dv_i = 0$ that is, recalling that H is invertible, $v_i \in \ker(D)$. □

The following Proposition 4.11 finally proves the linear convergence of the ra-ADMM applied to strongly convex cost functions.

Proposition 4.11. *Assume functions f_i in* (3.4) *are strongly convex. Then, there exists a neighbourhood $\mathcal{N}_{\mathbf{x}^*}$ of the optimal point $\mathbf{x}^*$ such that, if $\mathbf{x}(0) \in \mathcal{N}_{\mathbf{x}^*}$, then Algorithm 5 converges in mean-square exponentially fast to the optimal solution, i.e.,*

$$\mathbb{E}\left[\|x_i(k) - x^*\|^2\right] \leq C\lambda^{-k}\|x_i(0) - x^*\|^2, \qquad \forall i \in \mathcal{V},$$

for suitable constants $C > 0$ and $0 \leq \lambda < 1$.

Proof. The probabilistic framework used to define the ra-ADMM requires that the expected error $\mathbb{E}[\|\mathbf{x}(k) - \mathbf{x}^*\|^2]$ be evaluated to prove the almost sure convergence with linear rate. Similarly to the deterministic case, the first step is to derive an upper bound to the expected error, and then to exploit it in order to prove the Proposition.

By following the steps used in the deterministic case, with the difference that expectation are involved and the new update (4.9) is used, it follows

$$\begin{aligned}\mathbb{E}[\|\mathbf{x}(k)-\mathbf{x}^*\|^2] \leq \mathbb{E}\left[\|DQ(k)(\mathbf{z}(0)-\mathbf{z}^*)\|^2\right] + \\ + \sum_{l=0}^{k-2}\mathbb{E}\left[\left\|DA(k-1)\cdots A(l+1)B(l)FE(\mathbf{x}(l)-\mathbf{x}^*)\right\|^2\right] + \\ + \mathbb{E}\left[\|DB(k-1)FE(\mathbf{x}(k-1)-\mathbf{x}^*)\|^2\right]\end{aligned}$$

and each of the three terms on the right-hand side will now be computed separately.

First of all consider the first term

$$\mathbb{E}\left[\|DQ(k)(\mathbf{z}(0)-\mathbf{z}^*)\|^2\right] = (\mathbf{z}(0)-\mathbf{z}^*)^\top \mathbb{E}\left[Q^\top(k))D^\top DQ(k)\right](\mathbf{z}(0)-\mathbf{z}^*).$$

Defining $\Delta(k) = \mathbb{E}[Q^\top(k)\Delta(0)Q(k)] = \mathbb{E}[A^\top(0)\cdots A^\top(k-1)\Delta(0)A(k-1)\cdots A(0)]$, with $\Delta(0) = D^\top D$, it is clear that the error depends on the dynamics of the following linear system:

$$\Delta(k+1) = \mathbb{E}[A^\top(0)\Delta(k)A(0)] = \mathcal{L}(\Delta(k)).$$

By an iterative argument it can be derived that $\Delta(k) = \mathcal{L}^k(\Delta(0))$ and therefore the goal is to prove linear convergence of this system.

By Proposition 4.9 it holds

$$\begin{aligned}\lim_{k\to\infty}\mathcal{L}^k(\Delta(0)) &= \lim_{k\to\infty}\mathbb{E}[Q^\top(k)\Delta(0)Q(k)] \\ &= \lim_{k\to\infty}\mathbb{E}\left[\left(\sum_{i=1}^m \rho_i v_i^\top\right)\Delta(0)\left(\sum_{i=1}^m v_i\rho_i^\top\right)\right] \\ &= \lim_{k\to\infty}\mathbb{E}\left[\sum_{i=1}^m\sum_{j=1}^m \rho_i v_i^\top\Delta(0)v_j\rho_j^\top\right].\end{aligned}$$

Recalling that $\Delta(0) = D^\top D$ and that by Lemma 4.10 it is $Dv_i = 0$ for any $i = 1,\dots,m$, it follows that

$$\lim_{k\to\infty}\mathcal{L}^k(\Delta(0)) = 0 \tag{4.10}$$

thus proving the linear convergence of the $\mathcal{L}(\cdot)$ system and therefore of the first term in the inequality.

Since $\Delta(k+1) = \mathbb{E}[A^\top(0)\Delta(k)A(0)]$, by the property of the vectorisation operator

$$\operatorname{vect}(ABC) = (C^\top \otimes A)\operatorname{vect}(B)$$

for matrices A, B and C of suitable dimensions, it holds

$$\begin{aligned}\operatorname{vect}(\Delta(k+1)) &= \operatorname{vect}\left(\mathbb{E}\left[A^\top(0)\Delta(k)A(0)\right]\right) \\ &= \mathbb{E}\left[\operatorname{vect}(A^\top(0)\Delta(k)A(0))\right] \\ &= \mathbb{E}\left[A^\top(0)\otimes A^\top(0)\right]\operatorname{vect}(\Delta(k)) \\ &= \Sigma^\top\operatorname{vect}(\Delta(k)).\end{aligned}$$

Therefore the convergence rate of the system $\Delta(k) = \mathcal{L}^k(\Delta(0))$ is given by the largest

eigenvalue of Σ[1] that is smaller than one. This eigenvalue will be indicated with λ, called the *effective spectral radius*. It follows that $\mathbb{E}\left[\|DQ(k)(\mathbf{z}(0)-\mathbf{z}^*)\|^2\right] \leq d\lambda^k$ with $d \in \mathbb{R}$.

By the properties of the norm and the expected value finally the error inequality can be rewritten as

$$\mathbb{E}\left[\|\mathbf{x}(k)-\mathbf{x}^*\|^2\right] \leq \bar{c}\lambda^k + \bar{c}\delta\mathbb{E}[\|\mathbf{x}(k-1)-\mathbf{x}^*\|^2] + \bar{c}\delta\sum_{l=0}^{k-2}\lambda^{k-l-1}\mathbb{E}[\|\mathbf{x}(l)-\mathbf{x}^*\|^2]$$

where $\bar{c} = \max\left\{d, d\mathbb{E}[\|B(l)F\|^2], \mathbb{E}[\|B(k-1)F\|^2]\right\}$ and the fact that $\|E(\mathbf{x}(l)-\mathbf{x}^*)\| \leq \delta\|\mathbf{x}(l)-\mathbf{x}^*\|$ was used.

The proof can be concluded following the steps described in the previous Chapter for the deterministic algorithm, with the difference that $w(k) = (\lambda+\epsilon)^{-k}\mathbb{E}[\|\mathbf{x}(k)-\mathbf{x}^*\|^2]$. □

The following Corollary of Proposition 4.11 gives an estimate of the convergence rate.

Corollary 4.12. *The expected convergence rate of the ra-ADMM depends on the effective spectral radius of the matrix $\Sigma = \mathbb{E}\left[A^\top(0)\otimes A^\top(0)\right]$, that is on the largest eigenvalue of Σ that is strictly smaller than one. Moreover, an explicit formula for computing Σ is as follows*

$$\Sigma = (1-2r)I\otimes I + rI\otimes A + R - R(I\otimes A) + rA\otimes I - R(A\otimes I) + R(A\otimes A) \quad (4.11)$$

with a suitable R matrix defined below.

Proof. The first claim of the Corollary follows from the proof of Proposition 4.11 reported above, while the explicit equation for computing Σ is described in the following.
Recall that $A(0) = I - B(0) + B(0)A$ with $B(0)$ a realisation of the matrix $B = \mathrm{diag}\{\boldsymbol{\beta}\}$, where $\boldsymbol{\beta}$ is the stacked vector of the random variables β_{ij} are Bernoulli with probability r but not all independent.
Therefore, omitting the time index which can be done since the matrix $B(0)$ appears in the expectation only, it is necessary to compute

$$\mathbb{E}[(I-B+BA)\otimes(I-B+BA)] = \mathbb{E}\Big[I\otimes I - I\otimes B + I\otimes BA - B\otimes I + B\otimes B + - B\otimes BA + BA\otimes I - BA\otimes B + BA\otimes BA\Big]$$

where each single term on the right-hand side can be computed separately by the linearity of the expectation.
The first term is clearly equal to itself, while the other are derived in the following. Note that where convenient the β_{ij} variables will be enumerated with the index $e = 1,\dots,M$, with M equal to twice the number of edges.

- First of all it holds

$$\mathbb{E}[I\otimes B] = \mathbb{E}[\mathrm{diag}\{\mathrm{diag}\{\boldsymbol{\beta}\},\dots,\mathrm{diag}\{\boldsymbol{\beta}\}\}] = rI\otimes I.$$

- $\mathbb{E}[B\otimes B]$ is a block diagonal matrix with the block relative to edge (i,j) equal to $\mathbb{E}[\beta_{ij}\,\mathrm{diag}\{\boldsymbol{\beta}\}]$. The diagonal elements of each block are of the type $\mathbb{E}[\beta_{ij}\beta_{lk}]$ where in general it might be $i = l$. In case $i \neq l$ then β_{ij} and β_{lk} are independent and $\mathbb{E}[\beta_{ij}\beta_{lk}] = r^2$.

[1] Recalling that spectrum of a square matrix corresponds to the spectrum of its transpose.

However if $ij = lk$ then it is $\mathbb{E}[\beta_{ij}^2] = r$, while if $i = l$ but $j \neq k$ it is $\mathbb{E}[\beta_{ij}\beta_{lk}] = r^2/q$. This last result can be derived by considering that the product $\beta_{ij}\beta_{lk}$ is equal to one if and only if $L_{ij} = L_{lk} = 0$ and $u_i = 1$; this event happens with probability $q(1-p)^2 = r^2/q$.
For simplicity in the following this matrix will be denoted with $R = \mathbb{E}[B \otimes B]$.

- $\mathbb{E}[B \otimes I]$ is a block diagonal matrix with the ij-th block equal to $\mathbb{E}[\beta_{ij} I] = rI$. Therefore $\mathbb{E}[B \otimes I] = rI \otimes I$.

- The diagonal blocks of $\mathbb{E}[I \otimes BA]$ are equal to $\mathbb{E}[BA] = \mathbb{E}[B]A = rA$, hence it can be written $\mathbb{E}[I \otimes BA] = rI \otimes A$.

- The matrix $\mathbb{E}[BA \otimes I]$ is equal to

$$\mathbb{E}\begin{bmatrix} \beta_1 a_{11} I & \cdots & \beta_1 a_{1M} I \\ \beta_2 a_{21} I & \cdots & \beta_2 a_{2M} I \\ \vdots & \ddots & \vdots \\ \beta_M a_{M1} I & \cdots & \beta_M a_{MM} I \end{bmatrix}$$

  which gives $\mathbb{E}[BA \otimes I] = rA \otimes I$.

- The ij-th block diagonal element of $\mathbb{E}[B \otimes BA]$ is equal to $\mathbb{E}[\beta_{ij} B]A = R_{ij}A$, with R_{ij} the ij-th diagonal block of R. Therefore it follows $\mathbb{E}[B \otimes BA] = R(I \otimes A)$.

- $\mathbb{E}[BA \otimes B]$ is equal to

$$\mathbb{E}\begin{bmatrix} \beta_1 a_{11} B & \cdots & \beta_1 a_{1M} B \\ \beta_2 a_{21} B & \cdots & \beta_2 a_{2M} B \\ \vdots & \ddots & \vdots \\ \beta_M a_{M1} B & \cdots & \beta_M a_{MM} B \end{bmatrix}$$

  that is, $\mathbb{E}[BA \otimes B] = R(A \otimes I)$.

- Finally $\mathbb{E}[BA \otimes BA]$ is

$$\mathbb{E}\begin{bmatrix} \beta_1 a_{11} BA & \cdots & \beta_1 a_{1M} BA \\ \beta_2 a_{21} BA & \cdots & \beta_2 a_{2M} BA \\ \vdots & \ddots & \vdots \\ \beta_M a_{M1} BA & \cdots & \beta_M a_{MM} BA \end{bmatrix}$$

  which yields $\mathbb{E}[BA \otimes BA] = R(A \otimes A)$.

Summing all terms the expression for Σ reported in the Corollary follows.

□

Remark 4.13. Note that a diagonal block R_{ij} of matrix R in practice has all the elements equal to r^2, except for $\mathbb{E}[\beta_{ij}\beta_{ik}] = r^2/q$ and $\mathbb{E}[\beta_{ij}^2] = r$.

Remark 4.14. Note that the complex structure of R is due to the fact that all the $\{\beta_{ij}\}_{j \in \mathcal{N}_i}$ coefficients depend on the same random variable u_i. Restricting to the case of synchronous updates the ij-th diagonal block of R reduces to the matrix $r^2\,\mathrm{diag}\{1, \ldots, 1, 1/r, 1, \ldots, 1\}$, the $1/r$ being in the ij-th position.

Corollary 4.15. *In case the cost functions are quadratic functions, then the linear convergence holds globally both for the ra-ADMM.*

Proof. This is a consequence of the fact that the Taylor expansion used to derive Equation (4.9) is not an approximation, and therefore $\mathbf{z}(k+1) = A(k)\mathbf{z}(k) + B(k)c$. □

4.4 Randomised ADMM and Final Considerations

Notice that both the r-ADMM and ra-ADMM conform to the randomised Krasnosel'skiĭ-Mann iteration of (4.5), and moreover that the convergence Proposition 4.3 does not depend on the actual probability space that selects the subset of co-ordinates to be updated, as long as condition (4.6) is satisfied.
Therefore it is possible to prove the convergence of the more general *randomised ADMM* (ran-ADMM) in which the co-ordinate selection can account for any number of reasons for a communication to take place or not.

In particular, the parallel with optimisation over graphs with time-varying topologies is clear. Indeed it is possible to adapt the ran-ADMM to account for the variation in the topology of the graph in the following way. Assume that the underlying connection graph is complete, that is, each node is connected by an edge to any other node. This would entail a number of edges equal to $N!$. Now, at any given time it is possible to require that only a very small subset of edges $\mathcal{A} \subset \mathcal{E}$ is active. Hence in the ran-ADMM collaboration between nodes i and j is possible if and only if $(i,j) \in \mathcal{A}$. Moreover, the packet loss and update probabilities can be included on top.
In most applications the topology of the graph would vary with low frequency (if at all), for example in the case of distributed Machine Learning problems solved with a network of CPUs, but in other it might vary quite often, being therefore an important factor, for example reconnaissance with Unmanned Aerial Vehicles (UAVs) or localisation in WSN.

Another interesting scenario arises if the packet loss between node i and its neighbour j is due to a failure of the link that bars both in-going and out-going transmissions to be delivered. In this case the r.v.s L_{ij} and L_{ji} are no longer independent, but instead can be unified in a single r.v. that accounts for the failure of the physical channel for instance.

Consider the r-ADMM, by Assumption 4.1 an auxiliary variable is not updated if $L_{ij} = 1$, which happens with a probability equal to p. Let $K > 0$ be the length of a time interval, therefore since the variable is updated according to a Bernoulli random variable, then the probability of updating k times during the interval K is equal to

$$\mathbb{P}[z_{ij} \text{ updated } k \text{ times}] = \binom{K}{k} p^{K-k}(1-p)^k,$$

that is, the number of updates performed in a given interval is a binomial random variable. Therefore the mean number of updates during the interval is equal to $(1-p)K$, and the probability of not updating z_{ij} is $\mathbb{P}[\text{no update}] = p^K$. Therefore there is an analogy between the results presented in this Chapter and the convergence theorem presented in [52] for a totally asynchronous version of the ADMM with a single co-ordinate update at each instant. However, the likelihood of never updating a particular co-ordinate converges exponentially fast to zero.

Chapter 5

Partition-Based Randomised ADMM

The previous Chapter described the randomised ADMM for consensus optimisation problems with local cost functions, that is, cost functions that depend only on the local state variable. However, it is possible that the problems to be solved have more complicated cost functions that, for example, depend on the local variable and the variables of a node's neighbours. This class of problem has been studied in the field of WSN and distributed Control. This Chapter therefore aims at extending the results of the previous one to this more complex scenario.

5.1 Problem Formulation

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes, each of which is assigned a local variable $x_i \in \mathbb{R}^n$, $i = 1, \dots, N$. The objective is to solve with a consensus procedure the following problem

$$\min_{x_i,\, i \in \mathcal{V}} \sum_{i=1}^{N} f_i(x_i, \{x_j\}_{j \in \mathcal{N}_i}) \tag{5.1}$$

where the scalar costs $f_i(x_i, \{x_j\}_{j \in \mathcal{N}_i})$ are convex functions of the local variable x_i and the variables of node i's neighbours $\{x_j\}_{j \in \mathcal{N}_i}$. Therefore differently from the previous problem (3.1) a node needs information from all its neighbours in order to compute the current cost.
In order to recast problem (5.1) into a formulation amenable to distributed solving procedures, the following steps are taken. Notice that this derivation follows essentially the one described in Section 3.1.1.

First of all, for each node i a new set of variables is defined $x_i^{(i)}$ and $\left\{x_j^{(i)}\right\}_{j \in \mathcal{N}_i}$ where the superscript (i) indicates that the variables are local copies of x_i and $\{x_j\}_{j \in \mathcal{N}_i}$ stored by node i.
The problem therefore becomes

$$\begin{aligned} \min_{x_i^{(i)}, \{x_j^{(i)}\}_{j \in \mathcal{N}_i},\, i \in \mathcal{V}} \quad & \sum_{i=1}^{N} f_i\left(x_i^{(i)}, \left\{x_j^{(i)}\right\}_{j \in \mathcal{N}_i}\right) \\ \text{s.t.} \quad & x_i^{(i)} = x_i^{(j)}, \\ & x_j^{(i)} = x_j^{(j)} \qquad \forall j \in \mathcal{N}_i \end{aligned} \tag{5.2}$$

where the constraint imposes that at the optimum consensus is reached.
The next step is to introduce the so-called *bridge variables*, which allow to rewrite the problem in a form suitable for application of the ADMM. In particular, the constraints in (5.2) are

rewritten as

$$x_i^{(i)} = y_i^{(i,j)} \qquad x_j^{(i)} = y_j^{(i,j)}$$
$$x_i^{(j)} = y_j^{(j,i)} \qquad x_j^{(j)} = y_j^{(j,i)}$$

and

$$y_i^{(i,j)} = y_i^{(j,i)} \qquad y_j^{(i,j)} = y_j^{(j,i)}$$

where the the superscript (i,j) indicates that the bridge variable is stored by node i and communicated to neighbour j.
Now, in order to simplify the notation, the following vectors are introduced to collect all the variables and bridge variables that node i stores

$$\mathbf{x}^{(i)} = \begin{bmatrix} x_i^{(i)} \\ \left\{x_j^{(i)}\right\}_{j\in\mathcal{N}_i} \end{bmatrix}, \qquad \mathbf{y}^{(i)} = \begin{bmatrix} \left\{y_i^{(i,j)}\right\}_{j\in\mathcal{N}_i} \\ \left\{y_j^{(i,j)}\right\}_{j\in\mathcal{N}_i} \end{bmatrix}.$$

Note that overall there are two sets of $|\mathcal{N}_i|$ bridge variables each,

$$\left\{y_i^{(i,j)}\right\}_{j\in\mathcal{N}_i} \quad \text{and} \quad \left\{y_j^{(i,j)}\right\}_{j\in\mathcal{N}_i},$$

corresponding to the constraints $x_i^{(i)} = y_i^{(i,j)}$ and $x_j^{(i)} = y_j^{(i,j)}$ respectively. Moreover, variable $x_i^{(i)}$ appears in $|\mathcal{N}_i|$ constraints, while the variables $x_j^{(i)}$ appear in one constraint each.

Finally problem (5.2) can be rewritten as

$$\begin{aligned} \min_{\mathbf{x}^{(i)},\, i\in\mathcal{V}} \ & \sum_{i=1}^{N} f_i\left(\mathbf{x}^{(i)}\right) \\ \text{s.t.} \quad & A_i\mathbf{x}^{(i)} + \mathbf{y}^{(i)} = 0, \\ & \mathbf{y}^{(i)} = P_i\mathbf{y}^{(j)} \qquad \forall j \in \mathcal{N}_i \end{aligned} \tag{5.3}$$

for a suitable matrix A_i and permutation matrix P_i, $i = 1, \dots, N$.
Stacking the constraints and defining

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{x}^{(N)} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} \mathbf{y}^{(1)} \\ \vdots \\ \mathbf{y}^{(N)} \end{bmatrix}$$

problem (5.3) can be therefore rewritten in the formulation of problem (3.4), which means that the ADMM described in the previous Chapters can be applied to solve it.

Remark 5.1. Throughout this Chapter it will be assumed that the variables $\{x_i\}_{i\in\mathcal{V}}$ are all vectors in $\mathbb{R}^n$ and hence so are the corresponding local copies. However it is possible to derive the results of the following Sections also in the case $x_i \in \mathbb{R}^{n_i}$ with in general the n_i's all different. This would lead to each node storing a series of local copies of vectors of different dimensions, *e.g.* $x_i^{(i)} \in \mathbb{R}^{n_i}$ but $x_j^{(i)} \in \mathbb{R}^{n_j}$ with $n_i \neq n_j$.

5.2 Partition-Based Randomised ADMM

Now that the problem has been cast in the right framework, that is, it can be rewritten as

$$\begin{aligned} \min_{\mathbf{x}^{(i)},\, i\in\mathcal{V}} \quad & \sum_{i=1}^{N} f_i\left(\mathbf{x}^{(i)}\right) \\ \text{s.t.} \quad & A\mathbf{x}+\mathbf{y}=0, \\ & \mathbf{y}=P\mathbf{y} \end{aligned}$$

for suitable A and P matrices, it is possible to apply the ran-ADMM studied in the previous Chapter.
Applying Equations (3.5) and (3.6) the ran-ADMM for the partition-based problem (5.1) is given by the following equations

$$\begin{aligned} \mathbf{x}^{(i)}(k) = \arg\min \Bigg\{ & f_i\left(x_i^{(i)}, \{x_j(i)\}_{j\in\mathcal{N}_i}\right) + \frac{\rho}{2}|\mathcal{N}_i|\left\|x_i^{(i)}\right\|^2 + \frac{\rho}{2}\sum_{j\in\mathcal{N}_i}\left\|x_j^{(i)}\right\|^2 + \\ & - \langle \sum_{j\in\mathcal{N}_i} z_i^{(j,i)}(k), x_i^{(i)}\rangle - \sum_{j\in\mathcal{N}_i}\langle z_j^{(j,i)}(k), x_j^{(i)}\rangle \Bigg\} \end{aligned} \tag{5.4}$$

and

$$\begin{cases} z_i^{(i,j)}(k+1) = (1-\alpha_k) z_i^{(i,j)}(k) - \alpha_k z_i^{(j,i)}(k) + 2\alpha_k\rho x_i^{(i)}(k) \\ z_j^{(i,j)}(k+1) = (1-\alpha_k) z_j^{(i,j)}(k) - \alpha_k z_j^{(j,i)}(k) + 2\alpha_k\rho x_j^{(i)}(k) \end{cases} \tag{5.5}$$

for each $j\in\mathcal{N}_i$.
Notice that the updates (5.5) for the auxiliary variables has the same form as (3.6) with the difference that they use a different primal variable. On the other hand, the update for the primal variables (5.4) depends now on all the components of $\mathbf{x}^{(i)}$ which means that in general it is not possible to compute each component separately, unless the cost function f_i is separable. However, the interest for the partition-based formulation stems exactly from the possibility of dealing with this type of costs.

Now that the update equations for the auxiliary variables have been established, it is possible to define the random variables $\beta_i^{(i,j)}$ and $\beta_j^{(i,j)}$ which are equal to one if respectively $z_i^{(i,j)}$ and $z_j^{(i,j)}$ have to be updated or not. Therefore similarly to the previous Chapter the ran-ADMM hinges on the following updates

$$\begin{aligned} z_i^{(i,j)}(k+1) = & (1-\beta_i^{(i,j)}(k)) z_i^{(i,j)}(k) + \\ & + \beta_i^{(i,j)}(k)\left((1-\alpha_k) z_i^{(i,j)}(k) - \alpha_k z_i^{(j,i)} + 2\alpha_k\rho x_i^{(i)}(k)\right) \end{aligned} \tag{5.6}$$

$$\begin{aligned} z_j^{(i,j)}(k+1) = & (1-\beta_j^{(i,j)}(k)) z_j^{(i,j)}(k) + \\ & + \beta_j^{(i,j)}(k)\left((1-\alpha_k) z_j^{(i,j)}(k) - \alpha_k z_j^{(j,i)} + 2\alpha_k\rho x_j^{(i)}(k)\right) \end{aligned} \tag{5.7}$$

for each $j\in\mathcal{N}_i$. For the general case of the ran-ADMM the β random variables can depend on any possible probability space.

Consider now the robust and asynchronous case, with the same reasoning applied in the previous Chapter to Equations (3.5) and (3.6) it is possible to assign to node i the storage and update of the variables $\left\{z_i^{(j,i)}, z_j^{(j,i)}\right\}_{j\in\mathcal{N}_i}$ so that the computation of $\mathbf{x}^{(i)}$ requires only local information. In order to update $z_i^{(j,i)}$ and $z_j^{(j,i)}$, node i needs to receive information from its

neighbour j, information that can be entirely stored in the temporary variables

$$\begin{aligned} q_i^{(i\to j)} &= -z_i^{(j,i)}(k) + 2\rho x_i^{(i)}(k) \\ q_j^{(i\to j)} &= -z_j^{(j,i)}(k) + 2\rho x_j^{(i)}(k). \end{aligned} \tag{5.8}$$

Before introducing the implementation of the partition-based ra-ADMM, the following Assumption will be made in order to simplify the algorithm, knowing that the convergence is guaranteed in the general case. Similarly to the case of Section 4.2 the random variables $u_i^{(i)}$, $\{u_j^{(i)}\}_{j\in\mathcal{N}_i}$ schedule the update of the primal variables, while the random variables $\{L_i^{(i,j)}\}_{j\in\mathcal{N}_i}$, $\{L_j^{(i,j)}\}_{j\in\mathcal{N}_i}$ indicate if a packet sent from i to j is lost.

Assumption 5.2.

- A node computes all the components of $\mathbf{x}^{(i)}$, if it is scheduled to update, or none at all. This means that at each $k \in \mathbb{N}$ it holds $u_i^{(i)}(k) = u_j^{(i)}(k) =: u^{(i)}(k)$ for any $j \in \mathcal{N}_i$, and $u^{(i)}(k)$ are independent when k varies.
- A packet is either completely received or not at all, there is no partial information loss. That is, at each $k \in \mathbb{N}$ it holds $L_i^{(i,j)}(k) = L_j^{(i,j)}(k) =: L^{(i,j)}(k)$, and $L^{(i,j)}(k)$ are independent when k varies.
- The families of random variables $\{u^{(i)}\}_{i\in\mathcal{V}}$ and $\{L^{(i,j)}\}_{(i,j)\in\mathcal{E}}$ are both i.i.d. Bernoulli with probability q and p respectively, and they are independent from each other at all times.

The partition-based ra-ADMM is therefore described by the following Algorithm.

Algorithm 6 Partition-based robust and asynchronous distributed ADMM.

Input: step-sizes $\{\alpha_k\}_{k\in\mathbb{N}}$, penalty ρ, termination condition K.

Initialise: $x_i^{(i)}(0)$, $x_j^{(i)}(0)$, $z_i^{(j,i)}(0)$ and $z_j^{(j,i)}(0)$ for each node i and neighbour $j \in \mathcal{N}_i$.

$k \leftarrow 0$
while $k < K$ each agent i **do**
  if update scheduled **then**
    compute $x_i^{(i)}(k)$ and $\{x_j^{(i)}(k)\}_{j\in\mathcal{N}_i}$ according to (5.4)
    for all $j \in \mathcal{N}_i$ compute $q_i^{(i\to j)}$ and $q_j^{(i\to j)}$ as in (5.8)
    transmit $\left\{q_i^{(i\to j)}, q_j^{(i\to j)}\right\}$ to node j
  end if
  for $j \in \mathcal{N}_i$ **if** $\{q_j^{(j\to i)}, q_i^{(j\to i)}\}$ is received **do**
    update $z_i^{(j,i)}$ and $z_j^{(j,i)}$ as

$$\begin{aligned} z_i^{(j,i)}(k+1) &= (1-\alpha_k) z_i^{(j,i)}(k) + \alpha_k q_i^{(j\to i)} \\ z_j^{(j,i)}(k+1) &= (1-\alpha_k) z_j^{(j,i)}(k) + \alpha_k q_j^{(j\to i)} \end{aligned}$$

  end for
  $k \leftarrow k+1$
end while

5.3 Quadratic Cost Functions

This Section will describe the partition-based ra-ADMM in the case of quadratic cost functions, which is used in the simulations reported in Section 6.4.

The cost functions are as follows

$$f_i(x_i^{(i)}, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}) = \left(A_{ii}x_i^{(i)} + \sum_{j\in\mathcal{N}_i} A_{ij}x_j^{(i)} - b_i\right)^\top Q_i \left(A_{ii}x_i^{(i)} + \sum_{j\in\mathcal{N}_i} A_{ij}x_j^{(i)} - b_i\right)$$

where $A_{ii}, \{A_{ij}\}_{j\in\mathcal{N}_i} \in \mathbb{R}^{r_i\times n}$, $b_i \in \mathbb{R}^{r_i}$, and with $Q_i \in \mathbb{R}^{r_i\times r_i}$ a positive definite symmetric matrix. Hereafter it will be assumed that $r_i = r$ for all $i \in \mathcal{V}$, but the results hold equally for different sizes of the Q_i matrices.
In this case it is possible to find a closed-form solution for the minimisation in (5.4), which will be derived in the following. In order to simplify the derivation, assume that

$$\mathbf{x}^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_N^{(i)} \end{bmatrix}$$

where the components of $\mathbf{x}^{(i)}$ have been re-ordered by subscript and a $x_j^{(i)}$ variable has been added for each $j \in \mathcal{V}$ even though $(i,j) \notin \mathcal{E}$. Define now

$$A_i = \begin{bmatrix} A_{i1} & A_{i2} & \cdots & A_{iN} \end{bmatrix}$$

where $A_{ij} = \mathbb{0}_{r\times n}$ if $(i,j) \notin \mathcal{E}$. The cost functions can be rewritten as

$$f_i\left(\mathbf{x}^{(i)}\right) = \left(A_i\mathbf{x}^{(i)} - b_i\right)^\top Q_i \left(A_i\mathbf{x}^{(i)} - b_i\right) \tag{5.9}$$

and by the definition of the A_i matrix it is clear that any component $x_j^{(i)}$ with $(i,j) \notin \mathcal{E}$ does not appear in the cost.
The update equation (5.4) becomes therefore

$$\mathbf{x}^{(i)}(k) = \underset{\mathbf{x}^{(i)}}{\arg\min} \left\{ \left(A_i\mathbf{x}^{(i)} - b_i\right)^\top Q_i \left(A_i\mathbf{x}^{(i)} - b_i\right) + \frac{\rho}{2}\mathbf{x}^{(i)\top}N_i\mathbf{x}^{(i)} - \langle \mathbf{z}_i, \mathbf{x}^{(i)}\rangle \right\}$$

where

$$N_i = \operatorname{diag}\{1, \ldots, 1, |\mathcal{N}_i|, 1, \ldots, 1\} \otimes I_n$$

and

$$\mathbf{z}_i = \begin{bmatrix} z_1^{(1,i)} & \cdots & z_{i-1}^{(i-1,i)} & \sum_{j\in\mathcal{N}_i} z_i^{(j,i)} & z_{i+1}^{(i+1,i)} & \cdots & z_N^{(N,i)} \end{bmatrix}$$

with $z_i^{(j,i)} = z_j^{(j,i)} = \mathbb{0}_n$ if $(i,j) \notin \mathcal{E}$.

The solution can now be found imposing the first-order optimality condition which yields

$$\mathbf{x}^{(i)} = \left(2A_i^\top Q_i A_i + \rho N_i\right)^{-1} \left(\mathbf{z}_i + 2A_i^\top Q_i b_i\right) \tag{5.10}$$

where $\left(2A_i^\top Q_i A_i + \rho N_i\right)$ is non-singular because Q_i is positive definite and N_i is clearly non-singular.

5.4 Local Variables of Different Sizes

As noted in Remark 5.1, the results presented in the previous Sections hold even though the local vectors $\{x_i\}_{i\in\mathcal{V}}$ are of different sizes. If now it is assumed that $x_i \in \mathbb{R}^{n_i}$ with n_i in general different from n_j, $j \in \mathcal{N}_i$, an interesting scenario occurs if the local costs are a function of the local variables only.
Therefore the partition-based problem (5.2) is equivalent to the problem of finding the minimum of the sum of functions that depend on variables of different sizes, and doing so in a distributed set-up in which neighbours exchange information in order to reach the solution. In a sense, it is the extension of the consensus optimisation problem treated in Chapters 3 and 4 for the case of local variables with different sizes.

In the scenario described above the local cost function becomes $f_i(x_i^{(i)}, \{x_j^{(i)}\}_{j\in\mathcal{N}_i}) = f_i(x_i^{(i)})$, and therefore the update problem (5.4) requires the minimisation of an objective function that is separable in $x_i^{(i)}$ and in $x_j^{(i)}$, $j \in \mathcal{N}_i$. Then Equation (5.4) is equivalent to

$$x_i^{(i)}(k) = \arg\min \left\{ f_i\left(x_i^{(i)}\right) + \frac{\rho}{2}|\mathcal{N}_i| \left\|x_i^{(i)}\right\|^2 - \langle \sum_{j\in\mathcal{N}_i} z_i^{(j,i)}(k), x_i^{(i)} \rangle \right\}$$

$$\left\{x_j^{(i)}(k)\right\}_{j\in\mathcal{N}_i} = \arg\min \sum_{j\in\mathcal{N}_i} \left\{ \frac{\rho}{2} \left\|x_j^{(i)}\right\|^2 - \langle z_j^{(j,i)}(k), x_j^{(i)} \rangle \right\}$$

and, imposing the first-order condition, it follows that

$$x_j^{(i)}(k) = \frac{1}{\rho} z_j^{(j,i)}(k).$$

Therefore it is clear that storing and updating local copies on agent i's neighbours is no longer needed, only the corresponding auxiliary variables are required. Indeed it holds

$$\begin{aligned} q_i^{(i\to j)} &= -z_i^{(j,i)}(k) + 2\rho x_i^{(i)}(k) \\ q_j^{(i\to j)} &= z_j^{(j,i)}(k) \end{aligned}$$

in which the variables $x_j^{(i)}(k)$ do not appear and node i needs only to compute $z_j^{(j,i)}$, $j \in \mathcal{N}_i$, which are only used to reach consensus and do not bear information that has been locally computed by solving an optimisation problem as is the case for $z_i^{(j,i)}$.
In this scenario the packets that need to be transmitted by node i to each of its neighbours are of the type $\left\{q_i^{(i\to j)},\ z_j^{(j,i)}(k)\right\}$.

Remark 5.3. The number of variables that a node needs to store and update goes from the $3|\mathcal{N}_i| + 1$ of the general case to $2|\mathcal{N}_i| + 1$ in this particular scenario.

Chapter 6

Simulations Results

The current Chapter presents the results of an extensive campaign of simulations carried out to evaluate the performance of the algorithms proposed in this Thesis.

6.1 Note on the Methodology

The simulations presented in this Chapter have been performed with the following set-up and methodology.

The graphs used to represent the underlying communication network at each run of the simulations are *random geometric graphs* with communication radius $r = 0.1$[p.u.], in which two nodes are connected by an edge if and only if their relative distance is smaller than r. Figure 6.1 depicts an example of this type of graphs.

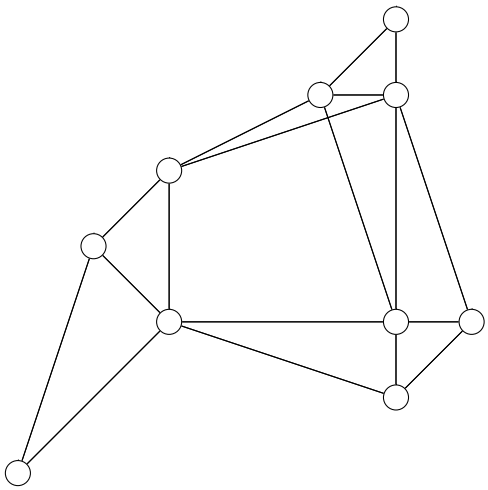


FIGURE 6.1: Example of random geometric graph with $N = 10$ nodes and communication radius $r = 0.1$[p.u.].

In order to derive from the simulations the characteristics of the proposed Algorithms and reject all artifices due to the particular set-up of each run, all the results presented have been derived averaging over a set of 50 to 100 Monte Carlo runs. For each run the set-up of the simulation has been drawn independently from the other runs, *e.g.* the communication graph is randomly generated at the beginning of each run.

The main metric for the evaluation of the proposed algorithms is the (evolution of the) *relative error*. Let $\mathbf{x}^* = \mathbb{1} \otimes x^*$ be the optimum of the consensus problem (3.4) where x^* is the minimum of $\sum_{i=1}^{N} f_i(x)$, then the relative error in logarithmic scale at the k-th iteration is defined as

$$\log \frac{\|\mathbf{x}(k) - \mathbf{x}^*\|}{\|\mathbf{x}^*\|} = \log \frac{\sum_{i=1}^{N} \|x_i(k) - x^*\|}{\sqrt{N}\,\|x^*\|}. \tag{6.1}$$

This error will be employed to evaluate all simulations except for those relative to the ADMM for partition-based problems, in which case the metric will be modified as follows

$$\log \sum_{i=1}^{N} \frac{\left\| \mathbf{x}^{(i)}(k) - \mathbf{x}^*_{(i)} \right\|}{\left\| \mathbf{x}^*_{(i)} \right\|} \tag{6.2}$$

where

$$\mathbf{x}^*_{(i)} = \begin{bmatrix} x_i^* \\ \{x_j^*\}_{j \in \mathcal{N}_i} \end{bmatrix}.$$

Finally, the termination condition used during the simulations was two-fold: a maximum number of iterations, usually 10^3, was imposed, and an early termination was performed whenever the relative error fell under a certain threshold (10^{-7} for quadratic costs and 10^{-3} for the Lasso and partition-based costs).

6.2 Robust and Asynchronous ADMM

In the simulations presented in this Section Algorithm 5 has been applied to two different problems, changing the values of the tunable parameters, step-size α (assumed constant) and penalty ρ, and the values of packet loss and update probabilities p and q.

6.2.1 Quadratic cost functions

The first benchmark functions used are quadratic costs of the type

$$f_i(x) = a_i x^2 + b_i x + c_i \tag{6.3}$$

with $a_i \in \mathbb{R}/\{0\}$ and $b_i, c_i \in \mathbb{R}$. In general the parameters of the local cost functions are different from each other, and randomly drawn at the beginning of each run.
Since the costs are differentiable, a closed-form solution can be derived for the problem that needs solving in the update equation (3.5), which is

$$x_i(k) = \frac{\sum_{j \in \mathcal{N}_i} z_{ji}(k) - b_i}{2a_i + \rho|\mathcal{N}_i|}. \tag{6.4}$$

Figure 6.2 depicts the evolution of the relative error when different values of the step-size α are used, with a fixed penalty $\rho = 3$, packet loss $p = 0.2$ and for simplicity synchronous updates. Recalling that the classic ADMM is characterised by $\alpha = 1/2$, then as the Figure shows the use of a relaxed ADMM can improve the convergence rate of the algorithm. In particular for values larger than $1/2$ that are inside the convergence region the r-ADMM is faster than it is with $\alpha \leq 1/2$.
Therefore these results justify the use of the somewhat more complicated relaxed ADMM instead of the classic version.
The effect of the second tunable parameter, the penalty ρ, is depicted in Figure 6.3 with a step-size of $1/2$, $p = 0.4$ and synchronous updates. Overall, the penalty that leads to the faster convergence is $\rho = 1$, but this might not be the case when the algorithm is applied to different problems. Therefore it is important to keep in mind that this second parameter can make the difference as well.

The following results analyse the effect that different packet loss probabilities have on the performance of the r-ADMM. First of all, Figure 6.4 reports the error evolution with $\alpha = 3/4$ and $\rho = 3/2$ for different values of p. As expected, the speed of convergence is

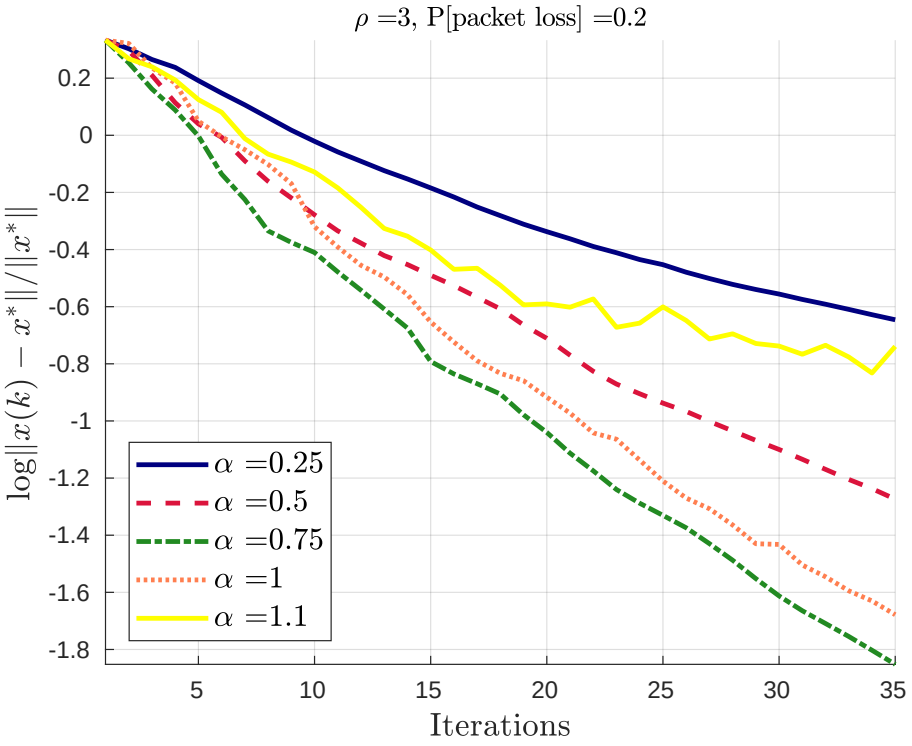


FIGURE 6.2: Error evolution of the r-ADMM for different step-sizes with quadratic costs.

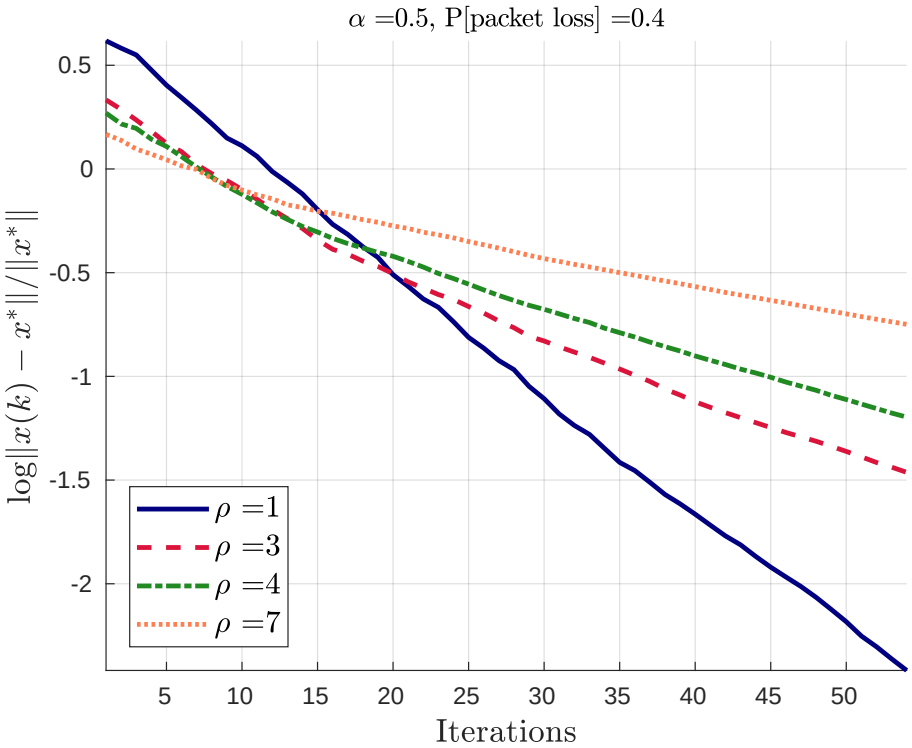


FIGURE 6.3: Error evolution of the r-ADMM for different penalty parameters.

negatively impacted by a larger probability of packet losses occurring, since each node on average performs less updates of the auxiliary variables in a given interval.
On the other hand, Figure 6.5 reports an interesting and counter-intuitive result. The Figure

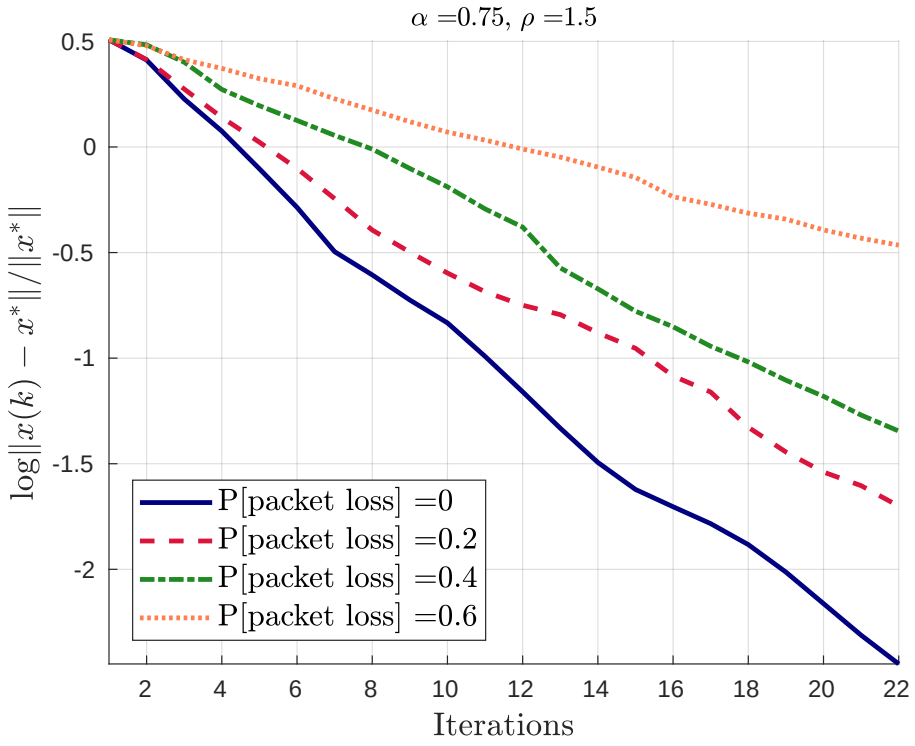


FIGURE 6.4: Error evolution of the r-ADMM for different packet loss probabilities with quadratic costs.

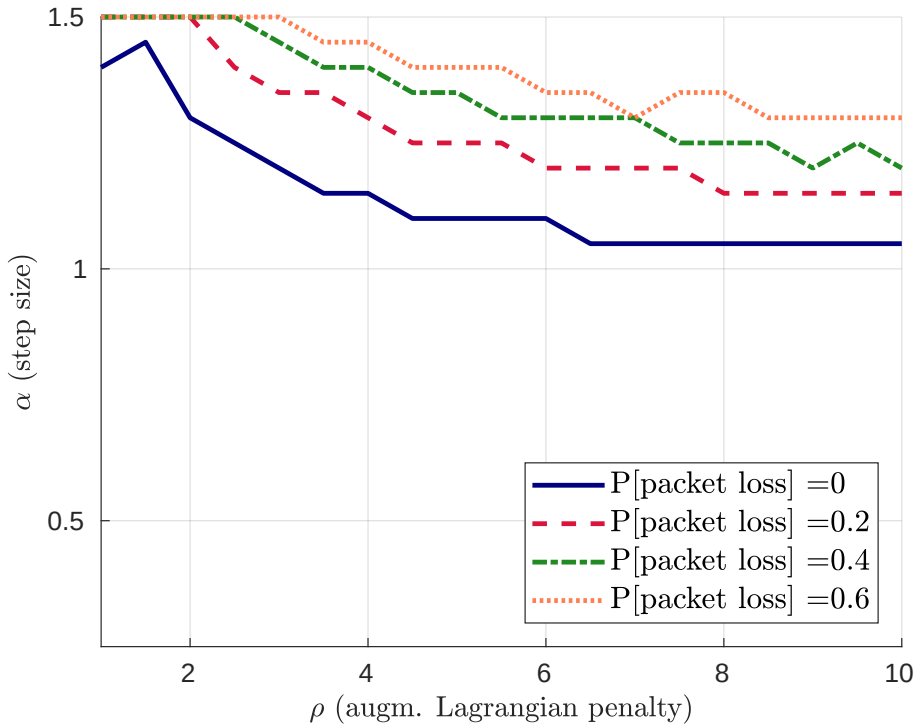


FIGURE 6.5: Stability boundaries of the r-ADMM for different packet loss probabilities with quadratic costs.

represents the empirical *stability regions* of the r-ADMM for different values of the packet loss probability. Proposition 4.2 guarantees the convergence if the free parameters are chosen

such that $0 < \alpha < 1$ and $\rho > 0$ but this is only a sufficient condition, and indeed the simulations results show that convergence is possible outside this region. The curves depicted in Figure 6.5 represent the boundary below which the r-ADMM converges and above which it diverges[1]. Clearly, for any of the packet loss probabilities the convergence region is larger then that guaranteed by Proposition 4.2. Moreover, the interesting phenomenon of the stability regions growing larger the larger p is can be observed. However as presented in Figure 6.4 this is counterbalanced by a slower rate of convergence.

The previous results were all obtained assuming that the updates are carried out synchronously, however the case of asynchronous updates is far more interesting for practical applications. By the framework described in Section 4.2, the introduction of asynchronous updates has the consequence of decreasing the probability of performing the update of an auxiliary variable from $1 - p$ to $q(1 - p)$, due to the fact that updated primal variables are not available at each time instant. Therefore the results obtained from the simulations of the ra-ADMM are very similar to those obtained for the r-ADMM even though the cause of a missed auxiliary update is in part different.
For example, similarly to Figure 6.4, the following Figure 6.6 depicts the evolution of the relative error for different values of the update probability q. Clearly the larger is the average number of updates carried out by a node, the faster is the convergence rate of the algorithm.

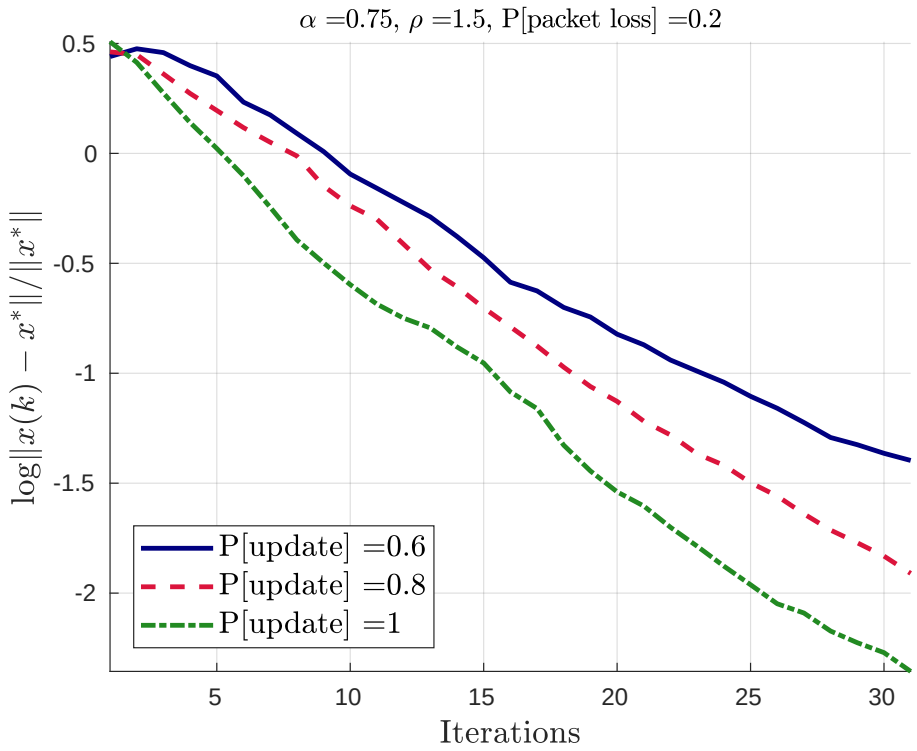


FIGURE 6.6: Error evolution of the ra-ADMM for different update probabilities with quadratic costs.

Finally, Section 4.3 proved that the convergence of the ra-ADMM is linear (in logarithmic scale, *i.e.* exponential) whenever the cost functions are strongly convex in the sense that there exists a constant $c > 0$ such that $\nabla^2 f_i(x_i) \succ cI_n$. This can be clearly observed in Figures 6.2, 6.3, 6.4, 6.6 where the evolution of the logarithm of the relative error is characterised by a line, since quadratic cost functions are indeed strongly convex.

[1]Since these results are averaged over 100 Monte Carlo runs, the r-ADMM is said to converge for a particular combination of α and ρ if it converges for a majority of the simulation repetitions.

Moreover, Corollary 4.12 states that an upper bound to the convergence rate of the ra-ADMM is given by the absolute value of the largest eigenvalue of the matrix Σ given by Equation (4.11). Figure 6.7 depicts the convergence rate measured empirically and the theoretical upper bound described by Corollary 4.12 in the case of $p = 0.4$, $q = 0.8$ and for different values of the free parameters α and ρ.

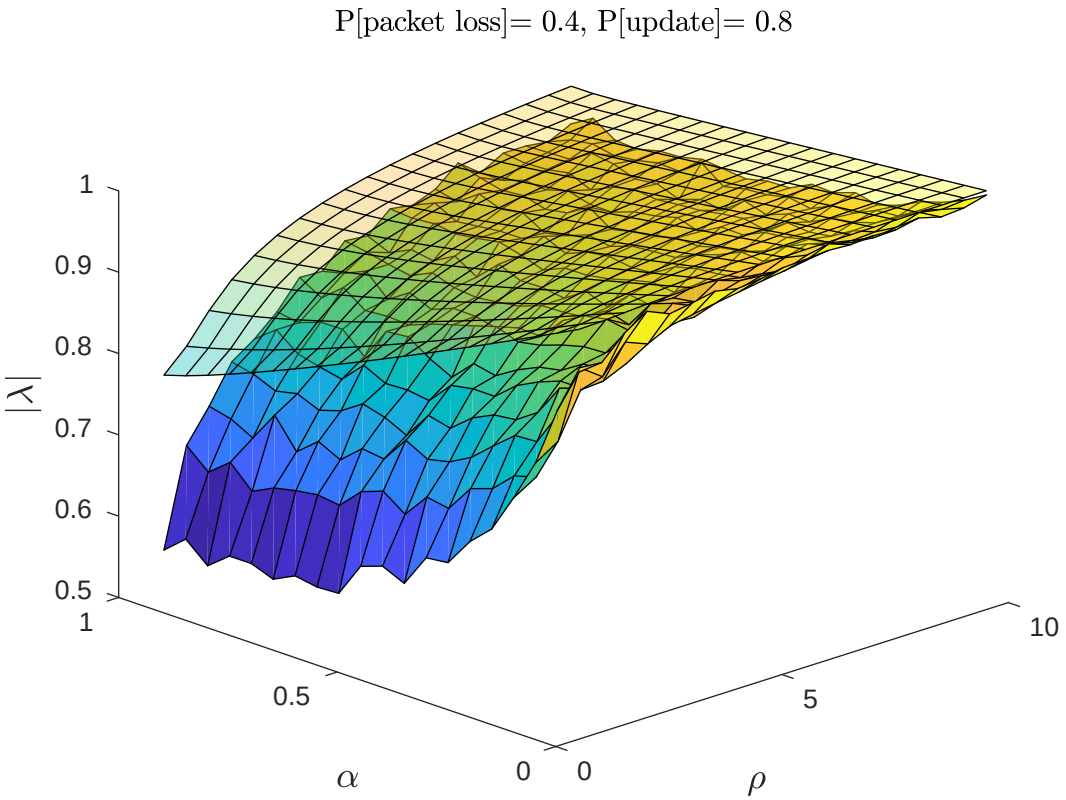


FIGURE 6.7: Empirical convergence rate and theoretical bound with quadratic costs.

Clearly the (semi-transparent) upper bound is tighter for large values of the penalty, but looser for smaller values of ρ.

6.2.2 Lasso

The following paragraphs introduce the *distributed Lasso* problem, then how the robust and asynchronous ADMM can be applied to solve it, and finally the results of the simulations performed on a synthetic data-set.

Problem formulation

The Lasso (from *least absolute shrinkage and selection operator*) problem, introduced by Tibshirani in [98], is a prominent method for *linear regression* in Machine Learning applications. The problem is defined as

$$\min_{x,x_0} \frac{1}{2D} \sum_{d=1}^{D} \|a_d x + x_0 - b_d\|_2^2 + \lambda \|x\|_1 \tag{6.5}$$

where the data are the input-output pairs (a_d, b_d), x is the vector of parameters of the model and x_0 the intercept that need to be found, and the variable $\lambda > 0$ is called the regularisation parameter. Notice that the objective function of (6.5) consists of two terms, the first that depends on the accuracy of the parameters estimated and can be summarised with $\|Ax + \mathbb{1}x_0 - b\|_2^2$ for suitable matrix $A \in \mathbb{R}^{D\times n}$ and vector $b \in \mathbb{R}^D$, and the second that penalises the size of the parameters, and is therefore called a *regularisation term*. Therefore the joint action of these two terms ensures that the solution with smallest 1-norm be found.
The remainder of this paragraph will present the distributed formulation of the Lasso. The possibility of solving the Lasso in a distributed fashion is of interest in many data-rich applications from Machine Learning to Smart Grids to healthcare [66, 95].

With a simple division of the regularisation term, problem (6.5) can be rewritten as

$$\min_{x,x_0} \sum_{i=1}^{N} \left\{ \frac{1}{2D} \|A_i x + x_0 - b_i\|_2^2 + \frac{\lambda}{N} \|x\|_1 \right\}$$

where the data have been divided in N blocks (A_i, b_i), not necessarily of the same size.
If now each block of data is assigned to a single node, the distributed Lasso can be straightforwardly defined as

$$\begin{aligned} &\min_{x,x_0} \sum_{i=1}^{N} \left\{ \frac{1}{2D} \|A_i x_i + x_{0,i} - b_i\|_2^2 + \frac{\lambda}{N} \|x_i\|_1 \right\} \\ &\text{s.t.} \quad x_i = x_j \quad \text{and} \quad x_{0,i} = x_{0,j} \qquad \forall (i,j) \in \mathcal{E}. \end{aligned} \tag{6.6}$$

Clearly problem (6.6) conforms to the problem (3.2) with the choice of local cost functions

$$f_i(x_i, x_{0,i}) = \frac{1}{2D} \|A_i x_i + x_{0,i} - b_i\|_2^2 + \frac{\lambda}{N} \|x_i\|_1 . \tag{6.7}$$

Notice that the f_i's are convex but not differentiable due to the presence of the non-smooth 1-norm regularisation term. Moreover, with this reformulation of the problem each node has to store locally the data (A_i, b_i) without the need to share them with any other node. This is particularly appealing in healthcare applications for instance, in which privacy policies dictate that the data from patients be protected.

Applying the ADMM

Applying the Algorithm 5 requires now that at each iteration in which node i performs an update, the following problem be solved

$$\begin{aligned} \begin{bmatrix} x_i \\ x_{0,i} \end{bmatrix}(k) = \arg\min_{x_i, x_{0,i}} \Bigg\{ &\frac{1}{2D} \|A_i x_i + x_{0,i} - b_i\|_2^2 + \frac{\lambda}{N} \|x_i\|_1 + \\ &- \langle \sum_{j\in\mathcal{N}_i} z_{ji}(k), \begin{bmatrix} x_i \\ x_{0,i} \end{bmatrix} \rangle + \frac{\rho}{2} |\mathcal{N}_i| \left\| \begin{bmatrix} x_i \\ x_{0,i} \end{bmatrix} \right\|^2 \Bigg\} \end{aligned} \tag{6.8}$$

However since f_i is not differentiable there is no closed-form solution to (6.8) and an iterative algorithm has to be applied. Since (6.8) conforms to the problem (2.14) then it is possible to apply the relaxed Peaceman-Rachford to solve it efficiently.
By choosing f in (2.14) to be the sum of all differentiable terms in problem (6.8), and g to be simply the 1-norm term, it is possible to leverage the following property of the proximal

operator to simplify the R-PRS iterates

$$\left[\text{prox}_{\rho\gamma\|\cdot\|_1}(u)\right]_i = \text{sign}(u_i)\max\{|u_i| - \rho\gamma, 0\}. \tag{6.9}$$

Therefore a closed-form solution can be found for both the updates (2.17) and (2.18). Notice that the R-PRS requires a step-size and a penalty parameter to be defined, and they can be selected with considerations similar to those described above for the ra-ADMM.

Simulations results

For simplicity, the following results were obtained by applying assuming that the updates are carried out synchronously.

First of all, the effect of varying the tunable parameters α and ρ is depicted in Figures 6.8 and 6.9, respectively.

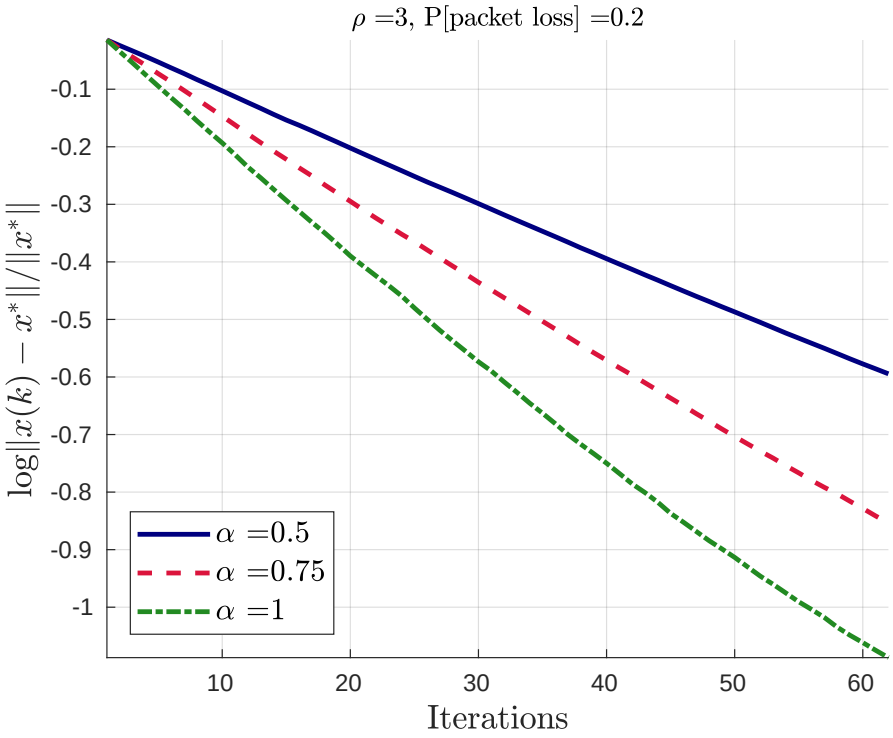


FIGURE 6.8: Error evolution of the r-ADMM for different step-sizes applied to the Lasso problem.

Clearly the results resemble very closely those obtained for the quadratic case, with step-sizes larger than $1/2$ guaranteeing faster convergence, and with a unitary penalty resulting in a smaller convergence rate.

Secondly, the effect of different packet loss probabilities is depicted in Figure 6.10, in which the larger p is, the slower the convergence of the r-ADMM results.

Notice that as in the case of quadratic cost functions, also for the Lasso problem the algorithm exhibits linear convergence in logarithmic scale. However, the results presented in Section 4.3 hold only if the cost functions are smooth, while the 1-norm is not differentiable in the origin. However, an alternative definition of strong convexity that can be applied to non-smooth functions states that $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is strongly convex if and only if $f(\cdot) - (\beta/2)\|\cdot\|^2$ is convex for some $\beta > 0$. Therefore, since the cost functions of the Lasso problem are strongly convex under this definition, the following Conjecture is proposed that will be the focus of further studies.

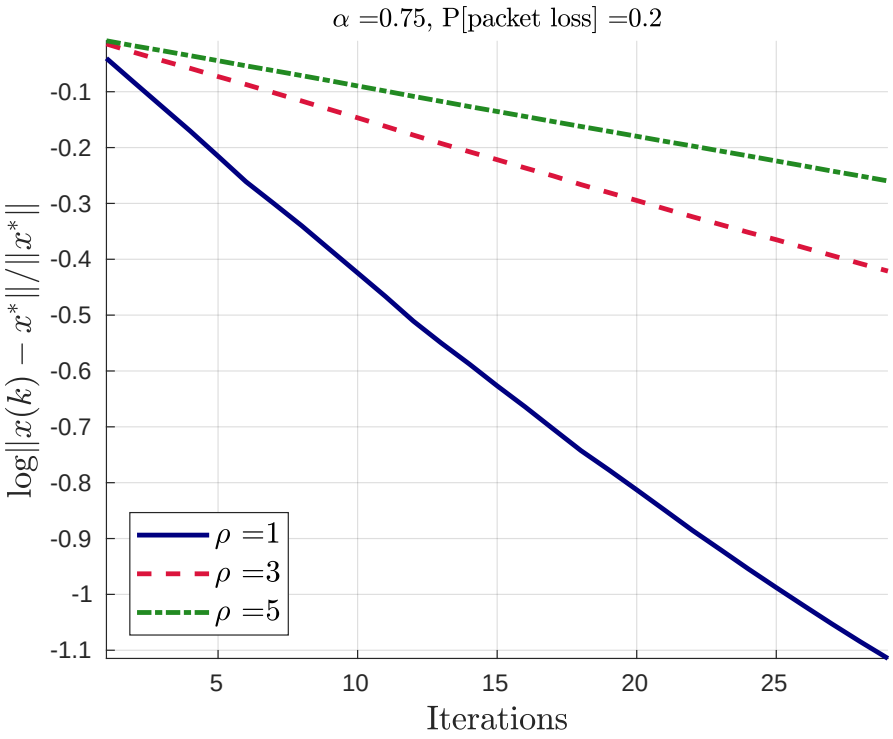


FIGURE 6.9: Error evolution of the r-ADMM for different penalty parameters applied to the Lasso problem.

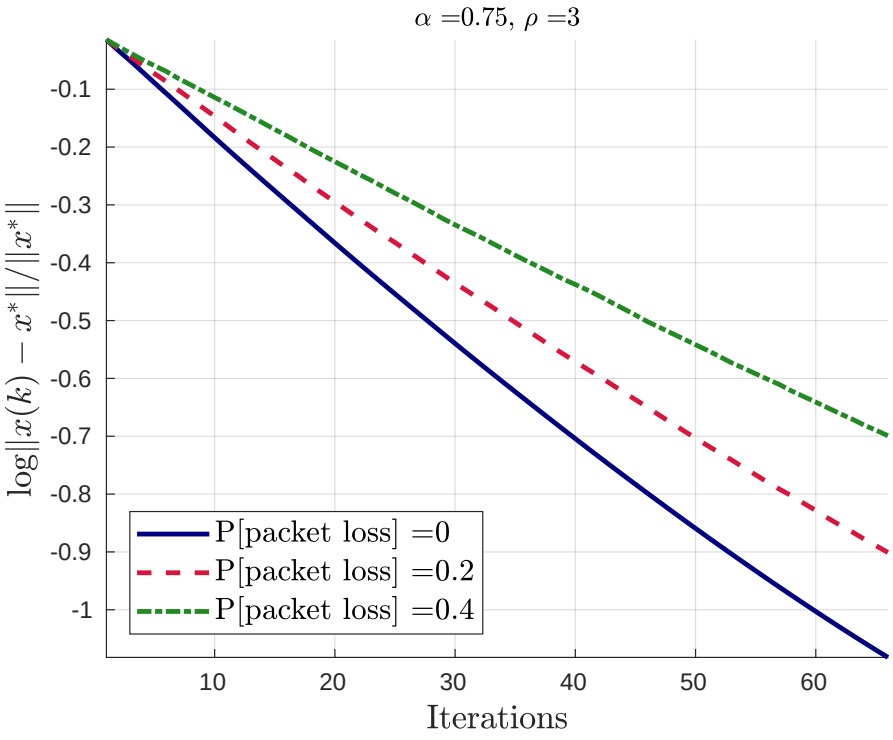


FIGURE 6.10: Error evolution of the r-ADMM for different packet losses applied to the Lasso problem.

Conjecture 6.1. The robust and asynchronous ADMM converges exponentially fast if the cost functions f_i are strongly convex in the sense that there exists a coefficient $\beta > 0$ such that $f_i(\cdot) - (\beta/2)\,\|\cdot\|^2$ is a convex function.

In conclusion, usually the performance of an algorithm applied to Machine Learning problems is evaluated by the fitness of the solution that it obtains when evaluated on the test set, a subset of the available data that is not used to compute the parameters x and intercept x_0. A particular metric well suited to this task is the *coefficient of determination* defined as

$$R^2 = 1 - \frac{\left\| b_t - \hat{b}_t \right\|^2}{\left\| b_t - \bar{b}_t \right\|^2} \tag{6.10}$$

where b_t is the output vector of the test data, $\hat{b}_t$ is the predicted output computed as $\hat{b}_t = A_t\hat{x} + \mathbb{1}\hat{x}_0$ with $\hat{x}$ and $\hat{x}_0$ the solution found by the ADMM, and $\bar{b}_t$ the mean of b_t. R^2 is a number always smaller than one and the closer it is to one the better the parameters fit the data.
Figure 6.11 reports the R^2 averaged over 100 Monte Carlo runs obtained varying both the regularisation parameter λ and the tolerance in the solution T_x, that is the threshold for the error below which the simulation terminates.

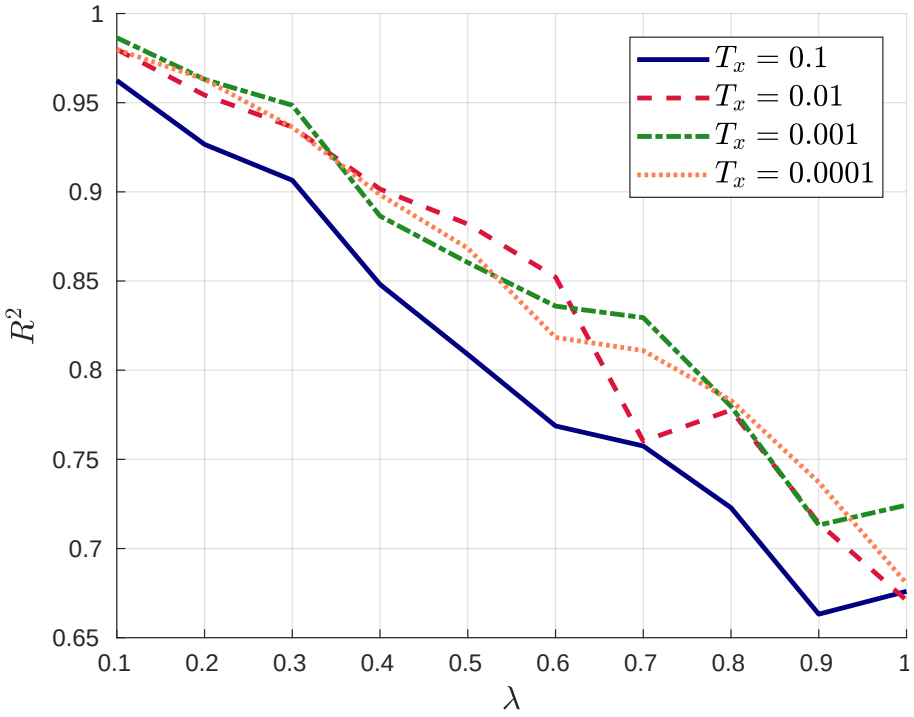


FIGURE 6.11: Coefficient of determination for different regularisation parameters and tolerances.

It is clear that the larger λ is, the less the solution fits the data, since the regularisation term imposes the parameters to be more sparser. On the other hand, for tolerances smaller than 10^{-1} the results are quite similar, suggesting that the algorithm has very good performances even though it has a looser early termination condition.

6.2.3 Considerations

In conclusion of the present Section some considerations are reported about the proposed ra-ADMM.

Synchronous vs asynchronous updates

As adduced in Chapter 4, the use of asynchronous updates can lead to better performance of the proposed algorithm as it results in a more efficient utilisation of the agents' time. However the results presented above do not highlight this effect since the asynchrony of updates is merely treated as a cause of missed auxiliary variables updates. In the following a simple example is presented that instead emphasises the positive aspects of asynchrony.

Consider a graph with three nodes all connected among each other, and suppose that one has a fixed period of $T = 5$ steps, in the sense that it needs T instants in order to solve the problem for updating the primal variable. The other two have periods T' and T'' that vary between one and T. Suppose that the algorithm is evaluated by the number of updates that the nodes can carry out with new information $U(T', T'')$. This is to take into account the fact that if a node updates without any neighbour having sent new information then the primal variable computed is the same as before.

The following Table 6.1 reports the percentage of time instants in which an update with new information can be carried out for different values of the periods T' and T''. Clearly if the nodes are allowed to update asynchronously, then it is possible to achieve a greater density of meaningful updates than with synchronous updates in most of the cases. Due to some overlapping of the periods, however, an improvement is not necessarily guaranteed.

TABLE 6.1: Percentage of time instants in which a primal variable is updated with new information.

	1	2	3	4	5
1	100	59.09	45.45	38.64	27.27
2		50	25.45	29.55	22.73
3			33.18	16.36	21.36
4				25	20.45
5					27.27

The larger number of updates with new information that the nodes are able to carry out asynchronously leads, intuitively, to a larger convergence rate. Assuming the absence of packet losses, the algorithm was tested with quadratic costs and with the same range of periods used above. Table 6.2 reports the number of steps that the algorithm required to converge, averaged over 100 Monte Carlo runs[2].

TABLE 6.2: Number of iterations for the ADMM varying the update periods with quadratic costs.

	1	2	3	4	5
1	112	115	105	158	757
2		105	130	141	758
3			169	167	758
4				558	759
5					551

The naïve criterion

In the proposed implementation of the ra-ADMM of Algorithm 5, a node updates the auxiliary variable relative to an edge (i, j) $j \in \mathcal{N}_i$ only if a new information $q_{j\to i}$ has been

[2]In general the number of iterations is not symmetric as the Table suggests, since the cost functions of a node might be more or less informative than the others', but for simplicity the cost were chosen to be equal.

received. That is, if node i does not receive a new packet from neighbour j then it does nothing.
However, a simpler criterion for dealing with the absence of new information would be for node i to perform an update of z_{ji} using the last packet $q'_{j\to i}$ that it received. Algorithm 7 represents an implementation of the ra-ADMM that employs such criterion, in which node i stores in memory the variable $q_{j\to i}$ which is overwritten with a new packet $q^{new}_{j\to i}$ whenever it arrives. The choice of always updating, possibly with the last packet received, is referred to as the *naïve criterion* since it is a simple and straightforward idea.

Algorithm 7 Naïve robust and asynchronous distributed ADMM.

Input: step-sizes $\{\alpha_k\}_{k\in\mathbb{N}}$, penalty ρ, termination condition K.
Initialise: $x_i(0)$ and $z_{ji}(0)$ for each node i and neighbour $j \in \mathcal{N}_i$.

$k \leftarrow 0$
while $k < K$ each agent i **do**
  if update scheduled **then**
    compute $x_i(k)$ according to (3.5)
    for all $j \in \mathcal{N}_i$ compute $q_{i\to j}$ as in (3.14) and transmit it to node j
  end if
  for $j \in \mathcal{N}_i$ **do**
    if packet $q^{new}_{j\to i}$ received **then**
      $q_{j\to i} \leftarrow q^{new}_{j\to i}$
    end if
    update z_{ji} as in (3.15)
  end for
  $k \leftarrow k + 1$
end while

Both the ra-ADMM and the naïve ra-ADMM were tested with a set of simulations changing the step-size, the penalty and the packet loss probability, with synchronous updates for simplicity. In order to compare the two algorithms the difference of iterations required by the ra-ADMM and the naïve was computed for any choice of the parameters. Clearly if there is no packet loss, the two algorithms are equivalent. However if $p > 0$ then the naïve ra-ADMM requires on average ~ 16 iterations less than the ra-ADMM, with the relatively high standard deviation of ~ 26 iterations.
There are two issues that suggest the use of the ra-ADMM instead of the naïve version, besides the fact that the speed-up with the latter is not very large. First of all, in Algorithm 7 each node needs to store $q_{j\to i}$ locally, hence taking the total number of variables stored by the single node from $|\mathcal{N}_i| + 1$ to $2|\mathcal{N}_i| + 1$, see Table 3.1. Secondly, even though in practice the naïve ra-ADMM is observed to converge it is not possible, to the best of the author's knowledge, to prove the convergence rigorously.

Consider now the Algorithm 2 characterised by the implementation of the ADMM described by the three Equations (2.39)–(2.41), and suppose to apply the naïve criterion in order to handle the loss of a packet. The resulting algorithm, referred to as naïve ADMM, appears to be faster than the proposed ra-ADMM, as illustrated by Figures 6.12 and 6.13.
However the naïve ADMM presents some problems that justify the choice of the proposed ra-ADMM. Indeed as highlighted by Table 3.1 the number of variables that each node needs to store using the naïve ADMM is $2|\mathcal{N}_i| + 1$, larger than the $|\mathcal{N}_i| + 1$ required by the ra-ADMM. Moreover as was the case for the naïve ra-ADMM described in Algorithm 7, there is no theoretical guarantee of convergence for the naïve ADMM.

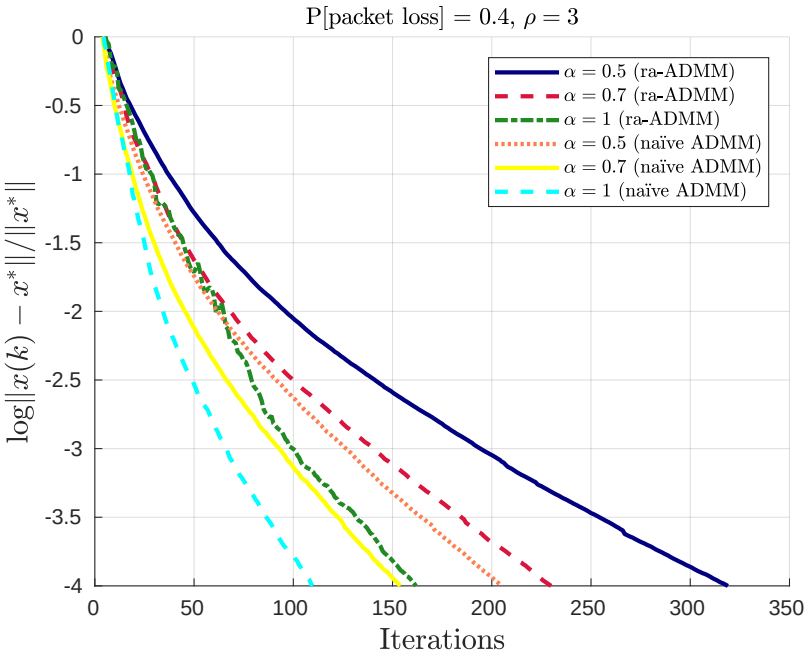


FIGURE 6.12: Error evolution of the ra-ADMM and naïve ADMM for different step-sizes with quadratic costs.

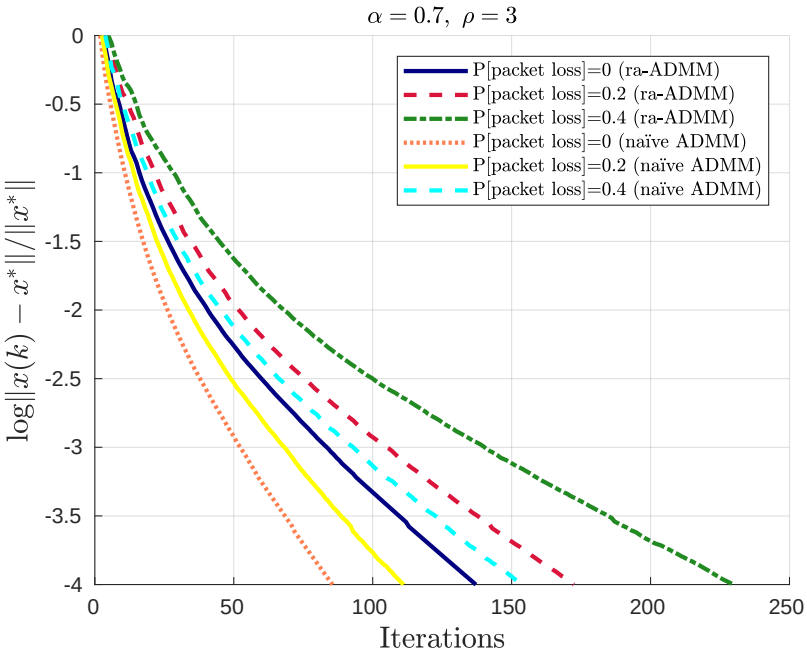


FIGURE 6.13: Error evolution of the ra-ADMM and naïve ADMM for different packet loss probabilities with quadratic costs.

6.3 Comparison with Newton-Raphson

The robust and asynchronous Newton-Raphson

The *robust and asynchronous Newton-Raphson* (ra-NR) developed in [15, 14, 8] addresses the problem of solving convex distributed optimisation problems. The algorithm is based on

the Newton-Raphson method, characterised by the iterate

$$x(k+1) = x(k) - \epsilon \left(\nabla^2 f(x(k))\right)^{-1} \nabla f(x(k)), \tag{6.11}$$

which is integrated with a *push-sum* and a *robust ratio consensus algorithms* in order to account for packet losses and asynchronous updates.
The ra-NR is guaranteed to converge if the cost functions are $\mathcal{C}^3$ and strongly convex. This is indeed the main difference with the ra-ADMM, which does not require smooth cost functions, and therefore is more widely applicable. Another difference is that the ra-NR requires to store and update more variables than the ra-ADMM, but needs to transmit less variables and, moreover, the transmission can be a broadcast to all neighbours since the same information needs to be sent to all neighbours. Table 6.3 reports a comparison of the exact counts of variables involved in the two algorithms.

TABLE 6.3: Comparison of ra-ADMM and ra-NR.

	ra-ADMM	ra-NR
Update and store	$\lvert\mathcal{N}_i\rvert + 1$	$2\lvert\mathcal{N}_i\rvert + 7$
Temporary	$\lvert\mathcal{N}_i\rvert$	$-$
Send	$\lvert\mathcal{N}_i\rvert$	2

Finally, notice that the ra-NR has a single tunable parameter, the step-size ϵ, while the ra-ADMM has more degrees of freedom, the penalty ρ and the sequence of step-sizes $\{\alpha_k\}_{k\in\mathbb{N}}$.

Comparison

Due to the fact that the ra-NR requires differentiable costs, the comparison simulations with the proposed ra-ADMM were performed using quadratic costs.

Figure 6.14 reports the error evolution for the ra-ADMM (above) with fixed step-size $\alpha = 0.6$ and penalty $\rho = 4$, and the ra-NR (below) with step-size $\epsilon = 0.9$, both evaluated with an update probability equal to 0.8.
The ra-ADMM has a faster convergence rate than the ra-NR, indeed for instance it reaches an error of 10^{-5} approximately 150 iterations before the latter.

Similar results are obtained in Figure 6.15 in which all the values of α evaluated for the ra-ADMM except for $1/2$ lead to a faster convergence. And again in Figure 6.16 the ra-ADMM shows better performances for almost any choice of the penalty.

6.4 Partition-Based ADMM

The partition-based costs defined in 5.3 were used to assess the proposed Algorithm 6. In particular it was chosen $n = 2$ and $r = 5$, the matrices $\{A_{ij}\}_{j\in\mathcal{N}_i\cup\{i\}}$ were independently generated with elements from a uniform distribution $\mathcal{U}(0,1)$, the components of vector b_i were drawn from the normal distribution $\mathcal{N}(0,1)$, and matrix Q_i was computed as

$$Q_i = \frac{1}{2}(R + R^\top) + rI_r$$

with $R \in \mathbb{R}^{r\times r}$ a matrix drawn from the uniform $\mathcal{U}(0,1)$.

Figure 6.17 shows that, similarly to the case analysed in 6.2, the larger the packet loss probability is, the more negative an effect it has on the convergence speed of the proposed algorithm.

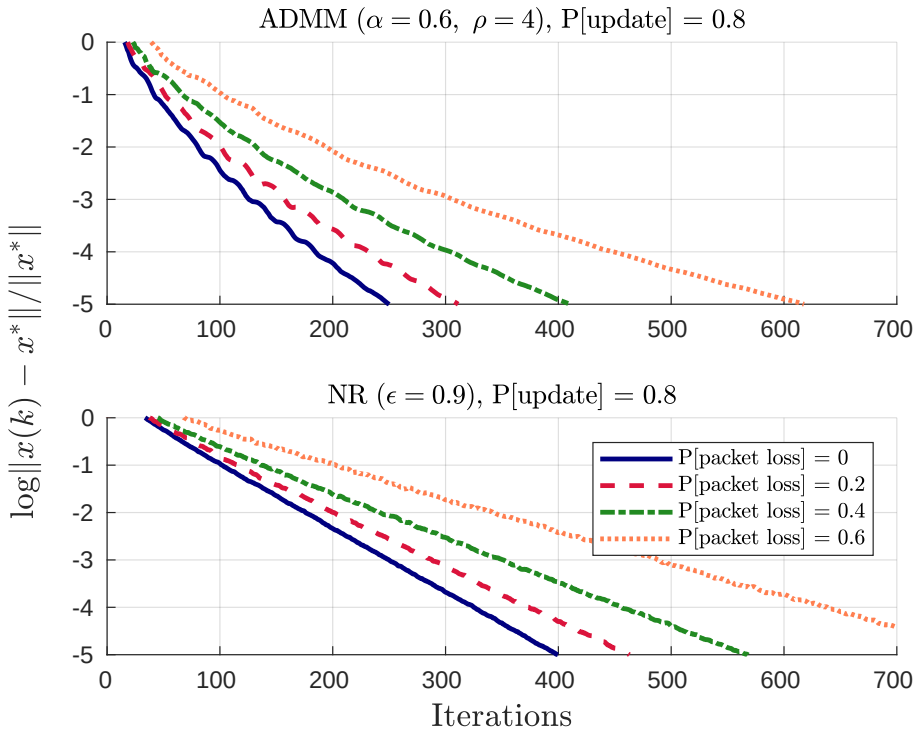


FIGURE 6.14: Error evolution of the ra-ADMM and the ra-NR for different packet losses.

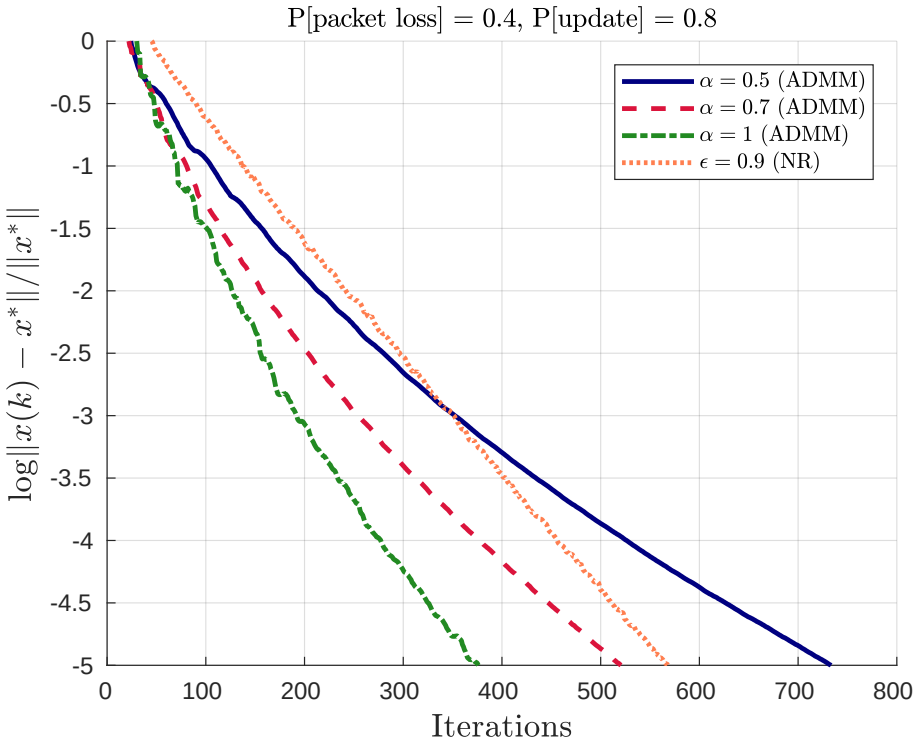


FIGURE 6.15: Error evolution of the ra-ADMM for different step-sizes and the ra-NR with $\epsilon = 0.9$.

However, this is counterbalanced by the fact that the stability regions in the parameters space, depicted in Figure 6.18, are larger the more likely the loss of a packet becomes. This result resembles very closely the one obtained for the simple ra-ADMM of Section 6.2.

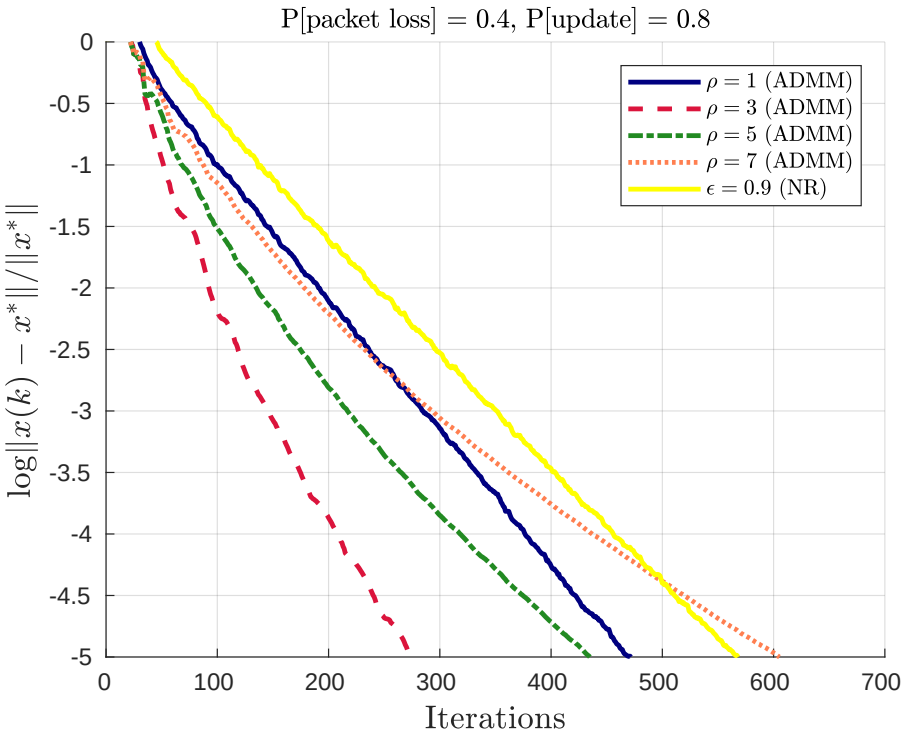


FIGURE 6.16: Error evolution of the ra-ADMM for different penalties and the ra-NR with $\epsilon = 0.9$.

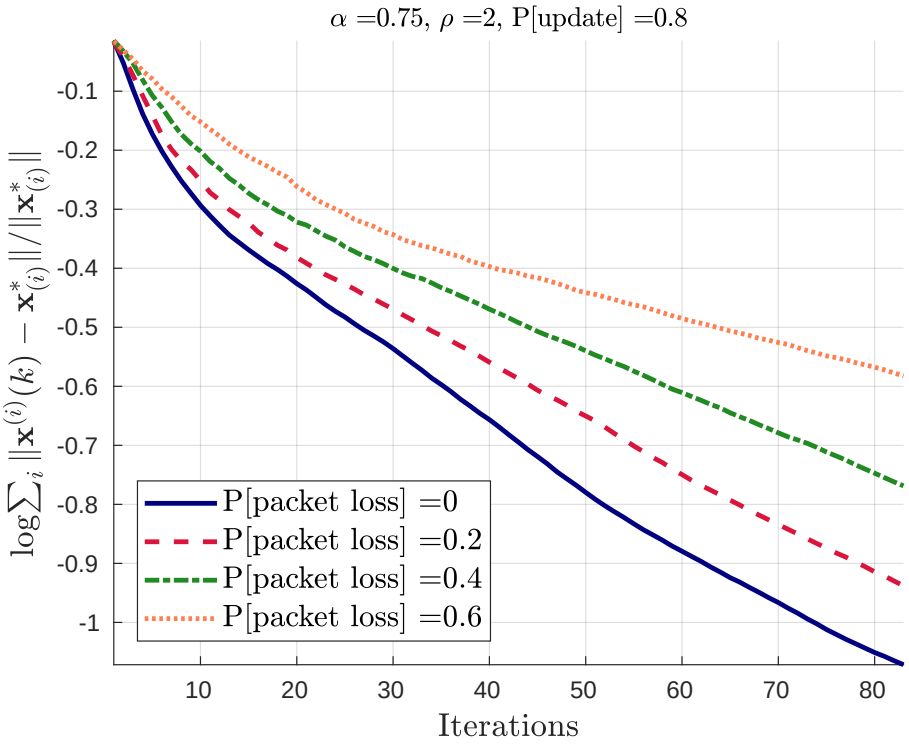


FIGURE 6.17: Error evolution of the partition-based ra-ADMM for different packet loss probabilities.

Finally, Figure 6.19 presents the error evolution of the partition-based ra-ADMM for three different values of the step-size α. Clearly the relaxation can be leveraged in order

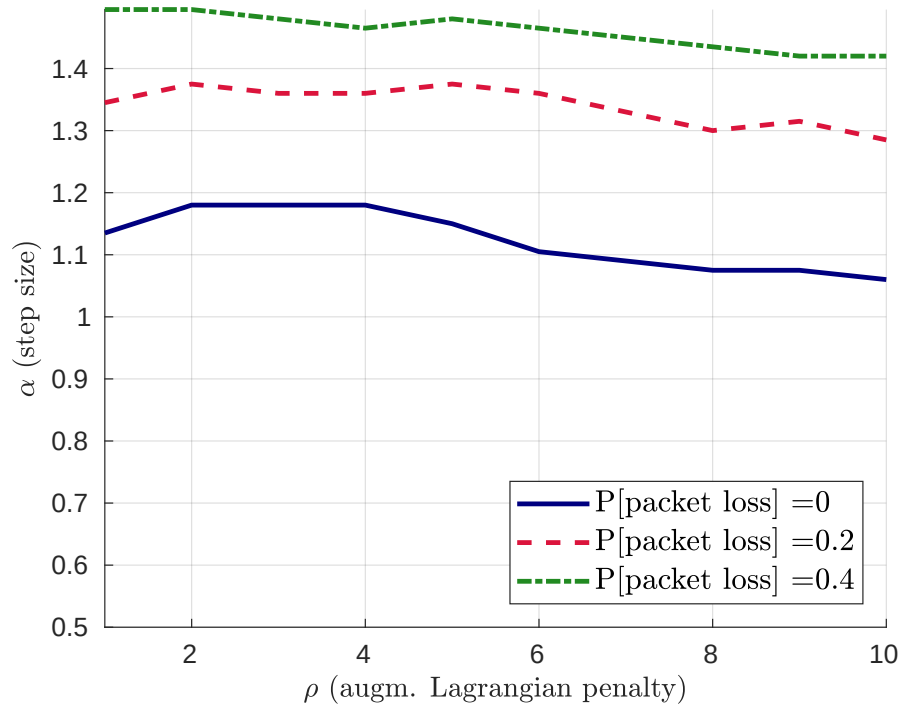


FIGURE 6.18: Stability boundaries of the partition-based ra-ADMM for different packet loss probabilities.

to improve the rate of convergence of the proposed algorithm as was the case with the ra-ADMM.

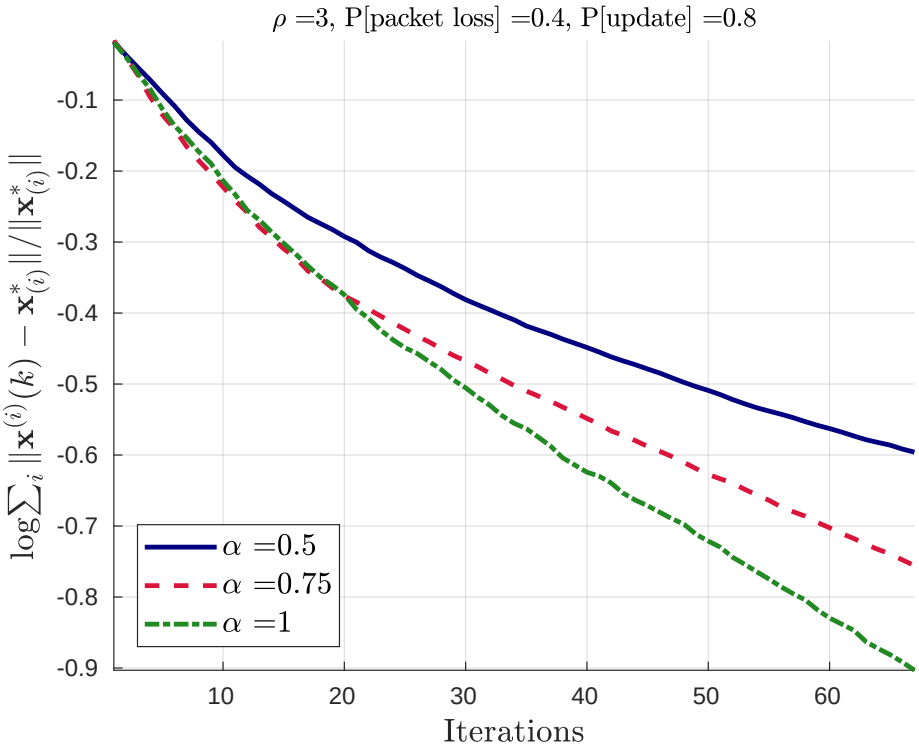


FIGURE 6.19: Error evolution of the partition-based ra-ADMM for different step-sizes.

Remark 6.2. In some of the results presented above, particularly those relative to partition-based problems, the error evolution presents two distinct phases. The first is a transitory that reduces the error with relative large rate. For instance in the case of the normal ra-ADMM of Figures 6.12 and 6.13 during this phase the error reaches values in the order of 10^{-2}. The second phase is instead characterised by a linear convergence with slower rate. This *long-tail* behaviour of the ADMM has been remarked upon for instance in [96]. Notice that the upper bound presented in Figure 6.7 refers to the slower convergence rate of the second phase.

Chapter 7

Conclusions and Further Work

Conclusions The aim of this Thesis has been to propose a formulation of the notorious Alternating Direction Method of Multipliers for solving distributed convex problem that is robust to faulty communications among agents and that can be applied with asynchronous updates. The convergence of the algorithm has been proved in a probabilistic framework and moreover the linear convergence has been proved in the case of strongly convex costs. The proposed algorithm has then been extended to solve partition-based problems, which are characterised by the dependence of the costs on the local variable of an agent but also on the local variables of its neighbours.
Extensive Monte Carlo simulations have been performed and the results analysed in detail. In particular the effect of the two tunable parameters of the algorithm has been investigated, alongside the performance reduction due to packet losses and asynchronous updates. The algorithm has been evaluated solving quadratic and Lasso problems, also comparing it with the robust and asynchronous Newton-Raphson.

Further Work The linear convergence of the proposed algorithm has been characterised by an upper bound for which an explicit formula is given. Moreover the convergence is guaranteed with a time-varying step-size. Therefore a first possible direction for further studies is to exploit these results in order to design an adaptive step-size scheme that can speed up the convergence of the algorithm.
An interesting phenomenon has been highlighted by the simulations results, namely the fact that the convergence in practice is guaranteed for values of the tunable parameters outside the region prescribed by the convergence Proposition. This behaviour is therefore a possible object for further studies.
Finally, the simulations carried out for the Lasso problems have shown a linear convergence rate. However the theory developed in this Thesis guarantees a linear rate of convergence only in the case of twice differentiable costs with Hessian bounded below. On the other hand a more general definition of strong convexity states that f has this property if there exists $\beta > 0$ such that $f(\cdot) - (\beta/2)\left\|\cdot\right\|^2$ is convex. Therefore studying the convergence rate of the proposed algorithm for this class of functions is certainly of interest.

Appendix A

Mathematical Background

A.1 Convex Analysis

This Section contains miscellaneous notions regarding convex functions and convex optimisation problems.

A.1.1 Convex functions

Definition A.1 (Hilbert space)**.** An *Hilbert space* $\mathcal{X}$ is a real or complex space with an inner product and such that all Cauchy sequences converge to points inside $\mathcal{X}$ with respect to the norm defined on the inner product.

The following definitions and properties all refer to the scalar function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ defined over the Hilbert space $\mathcal{X}$.

Definition A.2 (Closed function)**.** A function f is said to be *closed* if for any $a \in \mathbb{R}$ the set

$$\{x \in \mathrm{dom}(f) \mid f(x) \leq a\}$$

is closed.

Definition A.3 (Proper function)**.** A function f is said to be *proper* if it does not attain $-\infty$.

Definition A.4 (Convex and concave functions)**.** A function f is said to be *convex* if for any $x, y \in \mathcal{X}$ and scalar $\lambda \in [0, 1]$ it holds

$$f\left(\lambda x + (1-\lambda)y\right) \leq \lambda f(x) + (1-\lambda) f(y). \tag{A.1}$$

A function f is said to be *concave* if $-f$ is convex.

A function is said *strictly convex* if A.1 holds with a strict inequality. Now some properties and examples of convex functions follow.

Proposition A.5. *The domain of a convex function f, defined as*

$$\mathrm{dom}(f) = \{x \in \mathcal{X} \big| f(x) < +\infty\}$$

is a convex set.

Proposition A.6. *Let $\mathcal{Y}$ be a real Hilbert space, let $L : \mathcal{X} \to \mathcal{Y}$ be an affine function*[1] *and let $f : \mathcal{Y} \to \mathbb{R} \cup \{+\infty\}$. Then $f \circ L$ is a convex function.*

Example A.1. *The following are examples of convex functions.*

- *The indicator function ι_C of set C is a convex function if and only if C is a convex set.*

[1] A function $L : \mathcal{X} \to \mathcal{Y}$ is *affine* if $L(\lambda x + (1-\lambda)y) = \lambda L(x) + (1-\lambda)L(y)$ for any $x, y \in \mathcal{X}$ and $\lambda \in \mathbb{R}$.

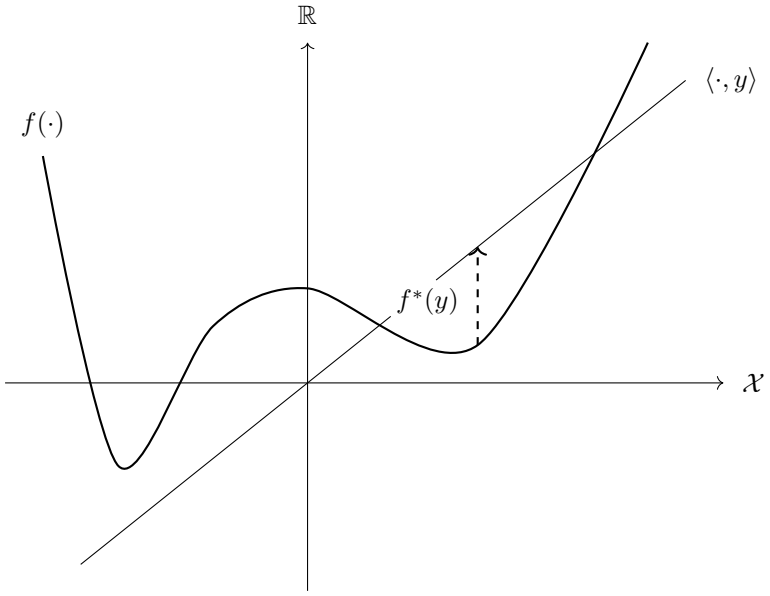


FIGURE A.1: Representation of the convex conjugate for a generic function $f : \mathbb{R} \to \mathbb{R}$.

- *The norm $\|\cdot\|$ and the squared norm $\|\cdot\|^2$ are convex functions.*

Definition A.7 (Strongly convex function)**.** A function f is said to be *strongly convex* if it is twice differentiable, *i.e.* $f \in \mathcal{C}^2$, and its Hessian is bounded from below, that is $\nabla^2 f(x) \succ cI$ for any $x \in \mathcal{X}$ and some positive scalar c.

The following alternative definition of strong convexity applies to non-smooth functions as well.

Definition A.8 (Strongly convex - alternative)**.** A possibly non-smooth proper function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ is *strongly convex* with constant $\beta > 0$ if and only if $f - \frac{\beta}{2} \|\cdot\|^2$ is convex.

A.1.2 Conjugation

Definition A.9 (Convex conjugate)**.** The *convex conjugate*, or *Legendre transform*, or *Legendre-Fenchel transform*, or *Fenchel conjugate*, of a convex function f is defined as

$$f^*(x) = \sup_{y \in \mathcal{X}} \{\langle x, y \rangle - f(y)\} = - \inf_{y \in \mathcal{X}} \{f(y) - \langle x, y \rangle\}. \tag{A.2}$$

In other words, the convex conjugate is the supremum of the signed vertical distance between the function f and the continuous linear functional $\langle \cdot, y \rangle$. Figure A.1 depicts with the dashed arrow the convex conjugate of a scalar function for a fixed y, which is indeed the distance between f and $\langle \cdot, y \rangle$.

Proposition A.10 ([88, Theorem 12.2])**.** *The convex conjugate f^* of a function f is always a closed and convex function. Moreover it is proper if and only if f is proper.*

Proposition A.11 (Fenchel-Young inequality [3, Proposition 13.13])**.** *Let f be proper, then*

$$f(x) + f^*(y) \geq \langle x, y \rangle \tag{A.3}$$

for any $x, y \in \mathcal{X}$.

Example A.2. *Let $f : \mathcal{X} \to [-\infty, +\infty]$, then $f = f^*$ if and only if $f = \frac{1}{2}\|\cdot\|^2$.*

Definition A.12 (Fenchel-Moreau [3, Theorem 13.32])**.** Let f be closed, proper and convex, then $f = f^{**}$.

An important corollary of the Fenchel-Moreau theorem follows.

Corollary A.13. *The convex conjugate of a closed, proper and convex function is closed, proper and convex.*

Let $f, g : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be proper, a frequent problem to be solved, for instance in Machine Learning applications, is

$$\min_{x \in \mathcal{X}} \{f(x) + g(x)\}. \tag{A.4}$$

The dual problem of (A.4) is defined as

$$\min_{y \in \mathcal{X}} \{f^*(y) + g^*(-y)\}$$

and this result goes under the name of *Fenchel duality*. The primal and dual optimal values are given by $p = \inf_{x \in \mathcal{X}} \{f(x) + g(x)\}$ and $q = \inf_{y \in \mathcal{X}} \{f^*(y) + g^*(-y)\}$ respectively, therefore the duality gap is

$$\Delta = \begin{cases} 0, & \text{if } p = -q \in \{-\infty, +\infty\}, \\ p + q, & \text{otherwise.} \end{cases}$$

Proposition A.14 ([3, Proposition 15.12])**.** *Under the assumptions described above, the following conditions are equivalent*

(i). $p \geq q$,

(ii). $\Delta \in [0, +\infty]$,

(iii). $p = q$ *if and only if* $\Delta = 0$.

Finally, in many applications the functions f and g are actually closed, proper and convex, in which case the following result holds.

Proposition A.15 ([3, Proposition 15.13])**.** *Let the closed, proper and convex functions f and g be such that the null vector belongs to the interior of* $\operatorname{dom}(f) - \operatorname{dom}(g) = \{x - y,\ x \in \operatorname{dom}(f),\ y \in \operatorname{dom}(g)\}$. *Then the duality gap is zero, that is $p = -q$.*

A.1.3 Convex optimisation

This Section will review some basic concepts in convex optimisation. Notice that the main part of the results reported below are taken from [11] and this reference will be omitted. Moreover, the attention is narrowed to functions defined on the real space $\mathbb{R}^n$.

The prototypical problem that has to be solved is of the form

$$\begin{aligned} \min_{x} \ & f_0(x) \\ \text{s.t.} \ & f_i(x) \leq 0 \quad i = 1, \dots, m \\ & h_i(x) = 0 \quad i = 1, \dots, p \end{aligned} \tag{A.5}$$

with $x \in \mathbb{R}^n$ and $f_0, \{f_i\}_{i=1}^m, \{h_i\}_{i=1}^p : \mathbb{R}^n \to \mathbb{R}$.
The *domain* is the subset of $\mathbb{R}^n$ defined as

$$\mathcal{D} = \bigcap_{i=0}^{m} \mathrm{dom}(f_i) \cap \bigcap_{i=1}^{p} \mathrm{dom}(h_i) \tag{A.6}$$

and a point is said to be *feasible* if it belongs to $\mathcal{D}$. In the following it is assumed $\mathcal{D} \neq \emptyset$, and the optimal value of (A.5) will be denoted by p^*.

Definition A.16 (Lagrangian and dual function)**.** The *Lagrangian* of problem (A.5) is the functional $\mathcal{L} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ defined as

$$\mathcal{L}(x; \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \tag{A.7}$$

where λ and ν are called the *dual variables* or *Lagrange multipliers*[2].
The *Lagrange dual function* relative to problem (A.5) is now defined as

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} \mathcal{L}(x; \lambda, \nu). \tag{A.8}$$

The dual function is, by definition, always a lower bound to the optimal value p^*, that is, for any $\lambda \geq 0$ and ν it holds $g(\lambda, \nu) \leq p^*$. Therefore it is of interest finding which is the best among these possible lower bounds, that will be the solution of the *Lagrange dual problem* (whereas (A.5) is referred to as the *primal problem*)

$$\begin{aligned} &\max_{\lambda,\nu} \; g(\lambda, \nu) \\ &\text{s.t.} \quad \lambda \geq 0. \end{aligned} \tag{A.9}$$

A point (λ, ν) is said to be *dual feasible* if $\lambda \geq 0$ and $g(\lambda, \nu) > -\infty$. Notice that the dual problem is convex even though the primal is not.

Denoting with d^* the optimal value of the dual (A.9), then the following inequality holds, which is called *weak duality*

$$d^* \leq p^*$$

and the value $p^* - d^*$ is named *duality gap*. If $p^* = d^*$ than *strong duality* holds and the duality gap is zero.
Conditions for strong duality to hold are very important for the purpose of solving (A.5). Suppose that the equality constraints in (A.5) are linear and that $f_0, \ldots, f_m$ are convex, then the problem becomes

$$\begin{aligned} &\min_{x} \; f_0(x) \\ &\text{s.t.} \quad f_i(x) \leq 0 \quad i = 1, \ldots, m \\ &\qquad Ax = b. \end{aligned} \tag{A.10}$$

Proposition A.17 (Slater's conditions)**.** *If there exists a point* $x \in \mathrm{rel\,int}(\mathcal{D})$ *such that*

$$f_i(x) < 0, \; i = 1, \ldots, m \qquad \textit{and} \qquad Ax = b \tag{A.11}$$

[2]Some texts use the inverse of the Lagrange multipliers as they were defined here, therefore subtracting the constraints in the Lagrangian. The two conventions are clearly equivalent.

then strong duality holds[3] *This is called Slater's condition.*
Moreover, if f_i $i = 1, \dots, l < m$ *are affine functions, then Slater's condition can be refined as*

$$f_i(x) \leq 0,\ i = 1, \dots, l, \qquad f_i(x) < 0,\ i = l+1, \dots, m \qquad \textit{and} \qquad Ax = b. \tag{A.12}$$

Remark A.18. Slater's conditions reduce to the feasibility of a point in $\operatorname{rel\,int}(\mathcal{D})$, that is the existence of a solution, if all constraints are linear equalities and inequalities and $\operatorname{dom}(f_0)$ is open.

If in problem (A.5) the functions $f_0, \dots, f_m$ are convex and the functions $h_1, \dots, h_p$ are affine, then it possible to define the following sufficient condition for optimality of a triple x^*, λ^* and ν^*.

Proposition A.19 (Karush-Kuhn-Tucker conditions)**.** *If a triple* x^*, λ^* *and* ν^* *satisfies the Karush-Kuhn-Tucker (KKT) conditions*

$$\begin{aligned} f_i(x^*) &\leq 0 \quad i = 1, \dots, m \\ h_i(x^*) &= 0 \quad i = 1, \dots, p \\ \lambda_i^* &\geq 0 \quad i = 1, \dots, m \\ \lambda_i^* f_i(x^*) &= 0 \quad i = 1, \dots, m \\ \nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i^* \nabla h_i(X^*) &= 0 \end{aligned} \tag{A.13}$$

then x^* *and* (λ^*, ν^*) *are the primal and dual optimums respectively and the duality gap is zero.*

Remark A.20. If the problem satisfies the Slater's conditions, then the KKT conditions become necessary and sufficient optimality conditions.

A.1.4 Precursors of the ADMM

This final part of the Section will briefly introduce two algorithms that are precursors of the ADMM, more details can be found in [10].

Consider problem

$$\begin{aligned} \min_x \ & f(x) \\ \text{s.t.} \ & Ax = b \end{aligned} \tag{A.14}$$

with $f : \mathbb{R}^n \to \mathbb{R}$ a convex function.

The *dual ascent* algorithm is characterised by the two iterates

$$\begin{aligned} x(k+1) &= \arg\min_x \mathcal{L}(x; \lambda(k)) \\ \lambda(k+1) &= \lambda(k) + \alpha_k (Ax(k+1) - b) \end{aligned}$$

where λ is the vector of Lagrange multipliers and $\{\alpha_k\}_{k \in \mathbb{N}}$, $\alpha_k > 0$, a sequence of step-sizes. If the cost function is separable, *i.e.* $f(x) = \sum_{i=1}^{N} f_i(x_i)$ then the dual ascent can be

[3] The relative interior of $\mathcal{D}$, denoted with $\operatorname{rel\,int}(\mathcal{D})$ are the points in the interior of $\mathcal{D}$ that lie within the affine hull $\operatorname{aff}(\mathcal{D}) = \left\{ \alpha_1 x_1 + \ldots + \alpha_l x_l \;\middle|\; l > 0,\ x_i \in \mathcal{D},\ \alpha_i \in \mathbb{R} \text{ s.t. } \sum_{i=1}^{l} \alpha_i = 1 \right\}$.

reformulated as

$$x_i(k+1) = \arg\min_{x_i} \mathcal{L}_i(x_i; \lambda(k))$$
$$\lambda(k+1) = \lambda(k) + \alpha_k(Ax(k+1) - b)$$

which is called the *dual decomposition* algorithm, where

$$\mathcal{L}_i(x_i; \lambda) = f_i(x_i) + \lambda^\top A_i x_i - \frac{1}{N}\lambda^\top b.$$

Notice that an implementation of the dual decomposition cannot be perfectly decentralised as the vector λ needs to be updated using information from all the agents that compute the x_i's.

A generalisation of the Lagrangian for problem (A.14) is the so-called *augmented Lagrangian* defined as

$$\mathcal{L}_\rho(x; \lambda) = f(x) + \lambda^\top(Ax - b) + \frac{\rho}{2}\|Ax - b\|^2 \tag{A.15}$$

where $\rho > 0$ is called *penalty parameter*.
The *method of multipliers* is now defined applying the dual ascent algorithm to problem (A.14) but using the augmented Lagrangian

$$x(k+1) = \arg\min_{x} \mathcal{L}_\rho(x; \lambda(k))$$
$$\lambda(k+1) = \lambda(k) + \rho(Ax(k+1) - b).$$

To conclude, recall that the classic ADMM (or the R-ADMM with $\alpha_k = 1/2$ for all $k \in \mathbb{N}$, solves problem

$$\min_{x,y} \; f(x) + g(y)$$
$$\text{s.t.} \quad Ax + By = c$$

with the iterates

$$x(k+1) = \arg\min_{x} \mathcal{L}_\rho(x, y(k); \lambda(k))$$
$$y(k+1) = \arg\min_{y} \mathcal{L}_\rho(x(k+1), y; \lambda(k))$$
$$\lambda(k+1) = \lambda(k) + \rho(Ax(k+1) + By(k+1) - c).$$

Then the resemblance with the dual ascent is clear, since there is a primal variable minimisation step (divided in two steps, one for the x and one for the y) and a dual variable update step. Moreover, as in the method of multipliers the step-size used is equal to the penalty parameter ρ, but in this case the primal update is divided in two steps instead of the single one

$$(x(k+1), y(k+1)) = \arg\min_{x,y} \lambda_\rho(x, y; \lambda(k)), \tag{A.16}$$

hence the name of *alternating direction* method.

A.2 Operator Theory

This Section collects various notions in Operator Theory, some of which have been already introduced in the main text.

A.2.1 Non-expansive operators

Definition A.21 (Operator)**.** An *operator* T on the Hilbert space $\mathcal{X}$ is a mapping $T : \mathcal{X} \to \mathcal{X}$ that assigns to each point $x \in \mathcal{X}$ the corresponding point $Tx \in \mathcal{X}$.

Definition A.22 (Non-expansive operators)**.** Let $\mathcal{X}$ be a Hilbert space, an operator $T : \mathcal{X} \to \mathcal{X}$ is said to be *non-expansive* if it has unitary Lipschitz constant, that is it verifies

$$\|Tx - Ty\| \leq \|x - y\|$$

for any two $x, y \in \mathcal{X}$.

Definition A.23 (Contractive operators)**.** Let $\mathcal{X}$ be a Hilbert space, an operator $T : \mathcal{X} \to \mathcal{X}$ is said to be *contractive* if it has Lipschitz constant strictly smaller than one, that is it verifies

$$\|Tx - Ty\| \leq L \|x - y\| < \|x - y\|$$

with $0 < L < 1$ for any two $x, y \in \mathcal{X}$.

Clearly contractiveness implies non-expansiveness, but not vice-versa. Note also that some authors use the term non-expansive to denote in general the class of operators with Lipschitz constant in $(0, 1]$.

Proposition A.24 ([4, Lemma 4.10])**.** *The composition of two non-expansive operators is a non-expansive operator itself.*

The following class of operators is characterised by even more restrictive properties.

Definition A.25 (Firmly non-expansive operators)**.** Let $\mathcal{X}$ be a Hilbert space, an operator $T : \mathcal{X} \to \mathcal{X}$ is said to be *firmly non-expansive* if for any $x, y \in \mathcal{X}$ it satisfies

$$\|Tx - Ty\|^2 + \|(I - T)x - (I - T)y\|^2 \leq \|x - y\|^2$$

where I is the identity operator on the Hilbert space $\mathcal{X}$.

Clearly firm non-expansiveness implies non-expansiveness, and moreover the following result holds.

Proposition A.26 ([3, Proposition 4.2])**.** *Let $\mathcal{X}$ be a Hilbert space and let $T : \mathcal{X} \to \mathcal{X}$. Then the following are equivalent*

(i). T is firmly non-expansive,

(ii). $I - T$ is firmly non-expansive,

(iii). $2T - I$ is non-expansive,

(iv). $\|Tx - Ty\|^2 \leq \langle x - y, Tx - Ty \rangle$ for any $x, y \in \mathcal{X}$.

Definition A.27 (Averaged operators)**.** Let $\mathcal{X}$ be a Hilbert space, let the operator $T : \mathcal{X} \to \mathcal{X}$ be non-expansive and let $\alpha \in (0, 1]$. Then the operator $T_\alpha : \mathcal{X} \to \mathcal{X}$ defined as $T_\alpha = (1 - \alpha)I + \alpha T$ is said to be *averaged* or *α-averaged*.

Remark A.28. Note that a non-expansive operator is averaged with $\alpha = 1$, while an operator is firmly non-expansive if and only if it is $1/2$-averaged.

Remark A.29. The α-averaging of a non-expansive operator is sometimes referred to as *relaxing*.

A useful characterisation of α-averaged operators is given by the following Proposition.

Proposition A.30 ([3, Proposition 4.25])**.** *Let $T : \mathcal{X} \to \mathcal{X}$ be a non-expansive operator, then it is α-averaged if and only if the following inequality holds*

$$\|Tx - Ty\|^2 \leq \|x - y\|^2 - \frac{1-\alpha}{\alpha} \|(I - T)x - (I - T)y\|^2 \tag{A.17}$$

for any pair of $x, y \in \mathcal{X}$.

A.2.2 Fixed point algorithms

Working with non-expansive operators the objective is often to find their fixed points, particularly when they are applied to convex optimisation problems.

Definition A.31 (Fixed points of an operator)**.** Let $T : \mathcal{X} \to \mathcal{X}$ be an operator on the Hilbert space $\mathcal{X}$, then the set of *fixed points* of T is defined as the set

$$\mathrm{fix}(T) = \left\{ x \in \mathcal{X} \;\middle|\; Tx = x \right\}.$$

The set of fixed points satisfies the following property.

Proposition A.32. *Let $\mathcal{D}$ be a non-empty, closed, and convex subset of $\mathcal{X}$ and let $T : \mathcal{D} \to \mathcal{X}$ be non-expansive. Then $\mathrm{fix}(T)$ is closed and convex.*

Before describing some possible *fixed point algorithms*, *i.e.* algorithm designed to find the fixed points of an operator, the following definitions are introduced.

Definition A.33 (Weak and strong convergence)**.** Let $\{x(k)\}_{k\in\mathbb{N}}$ be a sequence of points in the Hilbert space $\mathcal{X}$. The sequence is said to *converge weakly* to a point $\bar{x} \in \mathcal{X}$ if

$$\langle x(k), y\rangle \to \langle \bar{x}, y\rangle \; \forall y \in \mathcal{X}$$

for k that tends to infinity. Weak convergence will be denoted as $x(k) \rightharpoonup \bar{x}$.
Moreover, the sequence is said to *converge strongly* if

$$\|x(k) - \bar{x}\| \to 0$$

for $k \to \infty$, and this fact will be denoted with $x(k) \to \bar{x}$.

Definition A.34 (Fejér monotonicity)**.** Let $\mathcal{D}$ be a non-empty subset of the Hilbert space $\mathcal{X}$ and let $\{x(k)\}_{k\in\mathbb{N}}$ be a sequence in $\mathcal{X}$. Then the sequence is Fejér monotone with respect to $\mathcal{D}$ if

$$\|x(k+1) - x\| \leq \|x(k) - x\|$$

for all $x \in \mathcal{X}$ and for all $k \in \mathbb{N}$.

Now, the simplest algorithm that can be applied to find the fixed points of an operator is the *Banach-Picard iteration* [3, Theorem 1.48], defined as

$$x(k+1) = Tx(k). \tag{A.18}$$

The sequence $\{x(k)\}_{k\in\mathbb{N}}$ generated by this algorithm can be proved to converge weakly to a fixed point of the non-expansive operator T if the *asymptotic regularity* property $x(k) - Tx(k) \to 0$ holds [3, Theorem 5.13]. Moreover, if the operator is firmly non-expansive then the weak convergence to a point in $\mathrm{fix}(T)$ is guaranteed [3, Example 5.17].
However if this is not the case, the algorithm might not converge. Therefore the Krasnosel'skiĭ-Mann iteration [65, 62] is introduced, which enjoys more robust convergence properties that impose only the non-expansiveness on the operator. The Krasnosel'skiĭ-Mann iteration hinges on the following property of the averaged operators.

Proposition A.35. *Let $T : \mathcal{X} \to \mathcal{X}$ be a non-expansive operator on the Hilbert space $\mathcal{X}$ and $T_\alpha : \mathcal{X} \to \mathcal{X}$ be the corresponding averaged operator $T_\alpha = (1-\alpha)I + \alpha T$. Then it is* $\mathrm{fix}(T) = \mathrm{fix}(T_\alpha)$.

And the definition is given by the following Theorem.

Theorem A.36 (Krasnosel'skiĭ-Mann iteration [3, Theorem 5.14])**.** *Let $\mathcal{D}$ be a non-empty, closed, convex subset of the Hilbert space $\mathcal{X}$, let $T : \mathcal{D} \to \mathcal{D}$ be a non-expansive operator such that* $\mathrm{fix}(T) \neq \emptyset$*. Moreover, let $\{\alpha_k\}_{k\in\mathbb{N}}$ be a sequence in* $[0, 1]$ *such that*

$$\sum_{k\in\mathbb{N}} \alpha_k(1-\alpha_k) = +\infty. \tag{A.19}$$

The Krasnosel'skiĭ-Mann iteration is defined as

$$x(k+1) = (1-\alpha_k)x(k) + \alpha_k Tx(k) \tag{A.20}$$

with initial condition $x(0) \in \mathcal{D}$.
The following results hold for the sequence of points generated by the Krasnosel'skiĭ-Mann iteration.

(i). $\{x(k)\}_{k\in\mathbb{N}}$ *is Fejér monotone with respect to* $\mathrm{fix}(T)$*,*

(ii). $\{Tx(k) - x(k)\}_{k\in\mathbb{N}}$ *converges strongly to 0,*

(iii). $\{x(k)\}_{k\in\mathbb{N}}$ *converges weakly to a point in* $\mathrm{fix}(T)$*.*

This result therefore ensures that the Krasnosel'skiĭ-Mann iteration of a non-expansive operator converges (weakly) to a fixed point of said operator, given a suitable sequence of *step-sizes* $\{\alpha_k\}_{k\in\mathbb{N}}$.

Remark A.37. The Krasnosel'skiĭ-Mann iteration (A.20) at each time instant k essentially evaluates the α_k-averaged operator of T in the current state $x(k)$.
Moreover, the convergence result is satisfied in particular by a sequence of step-sizes $\alpha_k = \alpha$ for a constant α. In this case the Krasnosel'skiĭ-Mann iteration of operator T coincides with the Banach-Picard iteration of the α-averaged operator T_α.

Remark A.38. Result *(ii).* implies that the asymptotic regularity property required for the Banach-Picard iteration to converge is satisfied by the Krasnosel'skiĭ-Mann iteration.

A.2.3 Proximal algorithms

Definition A.39 (Proximal and reflective operators)**.** Let $\mathcal{X}$ be a Hilbert space, let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be a closed, proper, convex function and let $\rho > 0$. The *proximal operator*, or *proximity operator*, of f with penalty ρ is the operator $\mathrm{prox}_{\rho f} : \mathcal{X} \to \mathcal{X}$ defined as

$$\mathrm{prox}_{\rho f}(x) = \arg\min_{y\in\mathcal{X}} \left\{ f(y) + \frac{1}{2\rho}\|x-y\|^2 \right\}. \tag{A.21}$$

Moreover, the *reflective operator* of f is defined as

$$\operatorname{refl}_{\rho f} = 2 \operatorname{prox}_{\rho f} - I. \tag{A.22}$$

Proposition A.40. *The proximal operator is firmly non-expansive and the reflective operator is non-expansive.*

Proof. The first result is proved in [3, Proposition 12.27] and the second is a consequence of Proposition A.26. □

In the following are reported some properties of and results about the proximal operator, for the proofs and more details see [76, 27, 24].

Proposition A.41 (Translation, scaling, reflection)**.** *Let f be closed, proper, and convex, and let $y \in \mathcal{X}$, then the proximal operator of $f(x - y)$ is*

$$y + \operatorname{prox}_{\rho f}(x - y).$$

Let $\beta \in \mathbb{R} \setminus \{0\}$, then the proximal of $f(x/\beta)$ is

$$\beta \operatorname{prox}_{\rho f/\beta^2}\left(\frac{x}{\beta}\right).$$

Finally, the proximal operator of $f(-x)$ is

$$-\operatorname{prox}_{\rho f}(-x).$$

Proposition A.42 (Separable functions)**.** *If the closed, proper, convex function f is separable across two variables, i.e. $f(x, y) = \varphi(x) + \psi(y)$, then it holds*

$$\operatorname{prox}_{\rho f}(x, y) = \left(\operatorname{prox}_{\rho\varphi}(x), \operatorname{prox}_{\rho\psi}(y)\right).$$

Moreover, if it is fully separable, that is $f(x) = \sum_{i=1}^{n} f_i(x_i)$, then

$$[\operatorname{prox}_{\rho f}(x)]_i = \operatorname{prox}_{\rho f_i}(x_i).$$

Proposition A.43 (Affine transformation)**.** *Let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be a closed, proper, convex function. The proximal of the affine transformation $a f(x) + b$ with $a > 0$ is*

$$\operatorname{prox}_{a\rho f}(x).$$

Proposition A.44 (Quadratic perturbation)**.** *Let f be closed, proper and convex, let $a \geq 0$ and $\beta \in \mathbb{R}$. Then the proximal operator of the quadratic perturbation $f(x) + a \|x\|^2 / 2 + u^\top x + \beta$ is*

$$\operatorname{prox}_{\rho f/(a+1)}\left(\frac{x - u}{a + 1}\right). \tag{A.23}$$

In order to gain a better understanding of proximal operators, in the following the main two aspects of their effect are discussed.
First of all, the proximal operator maps the points that are outside the domain of the function $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ it is applied to onto the closest point on the boundary of $\operatorname{dom}(f)$. This is a consequence of the fact that points outside the domain of f will be penalised with an infinite cost by f, and therefore the proximal operator will minimise this cost by selecting a point that does not violate the domain, and this point will be the closest possible because of

the norm $\|x-y\|^2/(2\rho)$ that is part of the cost function. The following Example clarifies this description by looking at the proximal operator of the indicator function of four convex sets.

Example A.3. *Let $\mathcal{C}$ be a convex set and define the corresponding indicator function as*

$$\iota_{\mathcal{C}}(x) = \begin{cases} 0 & \text{if } x \in \mathcal{C}, \\ +\infty & \text{otherwise}. \end{cases} \tag{A.24}$$

The proximal operator of $\iota_{\mathcal{C}}(x)$, indicated for short as $\operatorname{prox}_{\mathcal{C}}(x)$, is the projection operator on the set $\mathcal{C}$ defined as

$$\operatorname{prox}_{\mathcal{C}}(x) = \mathrm{P}_{\mathcal{C}}(x) = \arg\min_{y \in \mathcal{C}} \left\{ \frac{1}{2} \|y-x\|^2 \right\}. \tag{A.25}$$

Some examples of projection operators follow, which are depicted in Figure A.2. In the figures, the thick lines represent the boundaries of each set $\mathcal{C}$, while the arrows map points outside the set $\mathcal{C}$ onto points on its boundary.

- $\mathcal{C} = \{x \in \mathcal{X} \mid \|x\|_2 \leq 1\}$, *see Fig. A.2a:*

$$\mathrm{P}_2(x) = \begin{cases} x & \|x\|_2 \leq 1 \\ \frac{x}{\|x\|_2} & \text{otherwise}. \end{cases}$$

- $\mathcal{C} = \{x \in \mathcal{X} \mid \|x\|_\infty \leq 1\}$, *see Fig. A.2b:*

$$[\mathrm{P}_\infty(x)]_i = \min\{\max\{x_i, -1\}, 1\}.$$

- $\mathcal{C} = \{x \in \mathcal{X} \mid \|x\|_1 \leq 1\}$, *see Fig. A.2c:*

$$[\mathrm{P}_1(x)]_i = \begin{cases} x_i & \|x\|_1 \leq 1 \\ \operatorname{sign}(x_i)\max\{|x_i| - \lambda, 0\} & \text{otherwise} \end{cases}$$

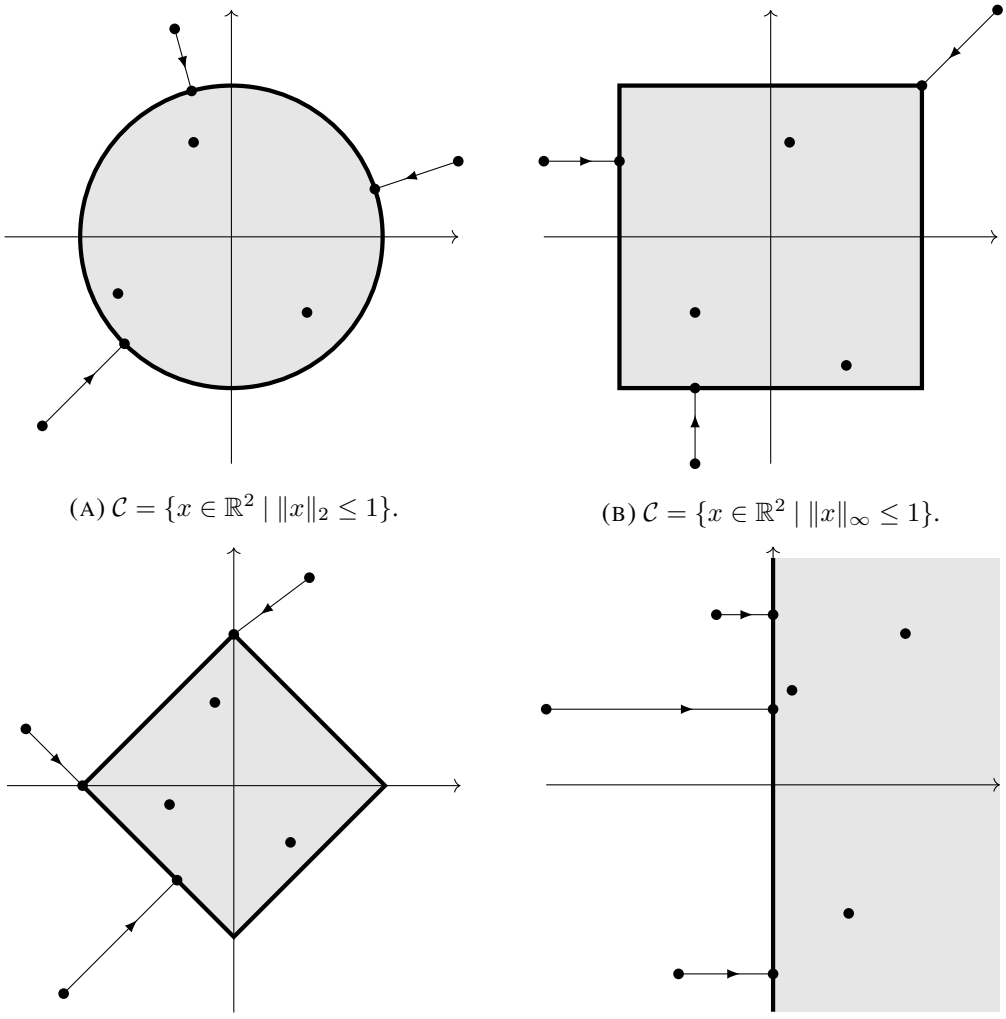
  where λ is the solution of $\sum_{i=1}^n \max\{|x_i| - \lambda\, 0\} = 1$.

- $\mathcal{C} = \{x \in \mathcal{X} \mid x \geq 0\}$, *see Fig. A.2d:*

$$\mathrm{P}_+(x) = \max\{x, 0\}.$$

Secondly, suppose that the proximal operator is applied to a point inside the domain of f. Then the operator will draw this point closer to the minimum of f, since it is effectively minimising f, but because of the 2-norm term the amount by which the point is shifted will depend on the penalty parameter ρ. Indeed the larger the penalty parameter is, the more the point will be drawn closer to the minimum since the penalty term weights increasingly less. The following Example represents this effect of the proximal operator.

Example A.4. *Consider the simple quadratic cost $f(x) = ax^2$, it is easily verified that $\operatorname{prox}_{\rho f}(x) = x/(2\rho+1)$, which indeed draws x closer to the origin each time it is applied. Figure A.3 depicts the repeated application of the proximal operator for the quadratic function. The extent by which the point is drawn closer to the minimum depends on the penalty ρ, with larger values that correspond to points mapped closer, which in this example is an effect of prox_f depending on the inverse of ρ.*



(A) $\mathcal{C} = \{x \in \mathbb{R}^2 \mid \|x\|_2 \leq 1\}$.

(B) $\mathcal{C} = \{x \in \mathbb{R}^2 \mid \|x\|_\infty \leq 1\}$.

(C) $\mathcal{C} = \{x \in \mathbb{R}^2 \mid \|x\|_1 \leq 1\}$.

(D) $\mathcal{C} = \{x \in \mathbb{R}^2 \mid x_1 \geq 0\}$.

FIGURE A.2: Action of the proximal operator for the indicator function of some particular sets.

In case the proximal of the convex conjugate f^* is easier to compute than that of f, the Moreau decomposition is a very useful result.

Definition A.45 (Moreau decomposition [76])**.** Let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be a closed, proper, convex function, then the *Moreau decomposition* holds for any point $x \in \mathcal{X}$

$$x = \operatorname{prox}_{\rho f}(x) + \rho \operatorname{prox}_{f^*/\rho}(x/\rho). \tag{A.26}$$

Remark A.46. Notice that the Moreau decomposition generalises the orthogonal decomposition induced by a subspace, which states that a vector can be expressed as the sum of its projection onto a subset and onto the corresponding orthogonal subset.

The following Proposition links proximal operators and convex optimisation problems, hence highlighting the importance of this class of non-expansive operators.

Proposition A.47. *Let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be a closed, proper, convex function. Then a point x^* is a minimiser of f if and only if it is a fixed point of* $\operatorname{prox}_{\rho f}$.

Proposition A.47 suggests that in order to solve a convex optimisation problem with cost function f it is sufficient to find the fixed points of the corresponding proximal operator.

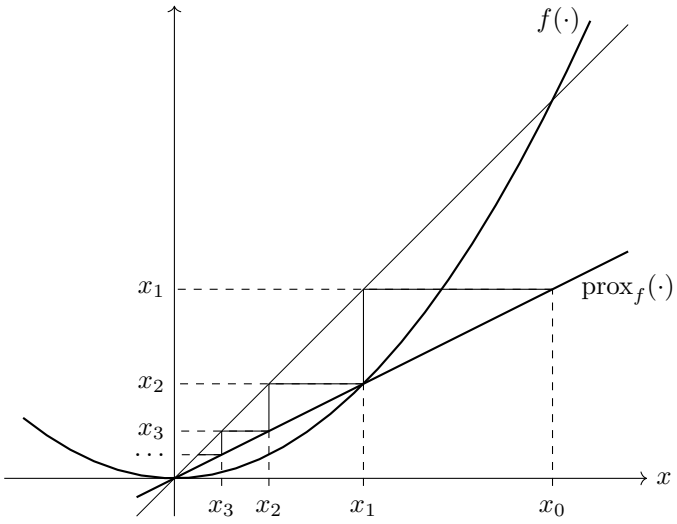


FIGURE A.3: Proximal operator for a scalar quadratic function.

Hence the focus is shifted from an optimisation problem to a fixed point problem, which can be solved with the algorithms described in A.2.2. And since the operator is a proximal operator, these algorithms go under the name of *proximal algorithms*.

To conclude this section the *proximal point algorithm* (PPA) [89] is introduced, which historically is the first proximal point to have been proposed.

Theorem A.48 (Proximal point algorithm [3, Theorem 27.1])**.** *Let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be closed, proper and convex. Let $\{\alpha_k\}_{k\in\mathbb{N}}$ be a sequence in $\mathbb{R}_+$ such that $\sum_{k\in\mathbb{N}} \alpha_k = +\infty$ and let $x(0) \in \mathcal{X}$. Then the proximal point algorithm is defined as*

$$x(k+1) = \mathrm{prox}_{\alpha_k f}(x(k)) \tag{A.27}$$

and is guaranteed to converge weakly to a minimiser of f.

Remark A.49. Notice that if the sequence $\{\alpha_k\}_{k\in\mathbb{N}}$ is such that $\alpha_k = \alpha$ for any $k \in \mathbb{N}$, then the PPA corresponds to the Banach-Picard iteration (A.18) applied to the proximal operator $\mathrm{prox}_{\alpha f}$.

A.2.4 Splitting operators

The main issue with the PPA is that at each iteration it requires to compute the proximal operator of f, which might be quite difficult and time consuming, depending on the structure of the function.

Many convex optimisation problems, however, have a structure amenable to the implementation of more efficient algorithms. Indeed, in particular in Machine Learning applications [80], the problems that need solving are characterised by an objective function that can be split into the sum of two terms

$$\min_{x\in\mathcal{X}} \{f(x) + g(x)\} \tag{A.28}$$

with f and g closed, proper and convex. A first approach to finding the solution might be to simply apply the PPA to the function $F(x) = f(x) + g(x)$. However this is not always an efficient solution, because as noted above the proximal operator of function $F(x)$ might be very difficult to compute.

Therefore a different approach relies instead on the so-called *splitting operators*, which leverage the separability of the cost function in order to define a series of smaller and simpler steps to be performed at each iteration.

Peaceman-Rachford splitting

An important class of splitting operators, particularly for their relationship with the ADMM, is centred on the *Peaceman-Rachford* operator [78] defined as

$$T_{PR} = \text{refl}_{\rho f} \circ \text{refl}_{\rho g} \,. \tag{A.29}$$

Consider now the sequence of step-sizes $\{\alpha_k\}_{k\in\mathbb{N}}$ satisfying property (A.19), applying the Krasnosel'skiĭ-Mann iteration to the PR operator yields the so-called *Relaxed Peaceman-Rachford Splitting* (R-PRS) scheme

$$z(k+1) = (1-\alpha_k)z(k) + \alpha_k T_{PR}(z(k)) \tag{A.30}$$

where z is an auxiliary variable from which the optimum of problem (A.28) can be computed according to $x^* = \text{prox}_{\rho g}(z^*)$ with z^* a fixed point of T_{PR}.
A possible implementation of the R-PRS is given by the following equations[4]

$$\psi(k) = \text{prox}_{\rho g}(z(k)) \tag{A.31}$$

$$\xi(k) = \text{prox}_{\rho f}\left(2\psi(k) - z(k)\right) \tag{A.32}$$

$$z(k+1) = z(k) + 2\alpha_k\left(\xi(k) - \psi(k)\right) \tag{A.33}$$

This scheme splits the computational effort between the two steps (A.31) and (A.32), unlike the PPA (A.27), taking advantage of the particular structure of (A.28).

Finally, there are two more splitting schemes based on the PR operator: the *Douglas-Rachford Splitting* (DRS) [36] and the *Peaceman-Rachford Splitting* (PRS) [78]. Both can be derived from the R-PRS with the particular choice of step-sizes $\alpha_k = 1/2$ and $\alpha_k = 1$, for all $k \in \mathbb{N}$, respectively.

Remark A.50. The DRS therefore can be defined as the Banach-Picard iteration of the operator $T_{DR} = (I + T_{PR})/2$ while the PRS as the Banach-Picard iteration of T_{PR}.

Forward-backward splitting

A splitting scheme that is not based on the Peaceman-Rachford operator is the so-called *forward-backward splitting* (FBS).
Suppose that in problem (A.28) the function f is continuously differentiable and that its gradient is Lipschitz continuous with Lipschitz constant γ[5]. Then the *forward-backward operator* is defined as

$$T_{FB} = \text{prox}_{\eta g} \circ (I - \eta\nabla f) \tag{A.34}$$

where $\eta \in (0, 2/\gamma]$, and can be shown to be α_{FB}-averaged [31] with

$$\alpha_{FB} = \frac{1}{2 - \eta\gamma/2}.$$

[4] See Section 2.2 for the proof.

[5] A function $h : \mathcal{X} \to \mathcal{X}$ is said to be *Lipschitz continuous* with constant γ if it verifies $\|h(x) - h(y)\| \leq \gamma \|x - y\|$ for any $x, y \in \mathcal{X}$.

The FBS is therefore characterised by the Krasnosel'skiĭ-Mann iterate of T_{FB} and can be shown to converge to a minimum of problem (A.28) if the step-sizes are a sequence in $[0, \delta]$ with $\delta = \min\{1, 1/(\gamma\eta)\} + 1/2$ such that $\sum_{k\in\mathbb{N}} \alpha_k(\delta - \alpha_k) = +\infty$ [3].

Remark A.51. The FBS is sometimes defined as the Banach-Picard iterate of the T_{FB} operator, and the Krasnosel'skiĭ-Mann iterate of T_{FB} is referred to as *generalised FBS*.
Moreover, it is possible to vary the parameter η at each instant, provided that each element in the sequence $\{\eta_k\}_{k\in\mathbb{N}}$ lies in $(0, 2/\gamma]$.

From the FBS it is possible to derive the proximal point algorithm in the particular case of $f = 0$, and also the *gradient method* [24]

$$x(k+1) = x(k) - \eta\nabla f(x(k)) \tag{A.35}$$

in the case of $g = 0$. For this reason the FBS is sometimes referred to as the *proximal gradient method*.

Figure A.4 depicts the web of connections that links the fixed point algorithms that were introduced in this Section, highlighting the conditions to derive one from the other.

A.3 Subdifferentiability

Many notions and results in convex optimisation with non-smooth functions and operator theory are tied to the concept of subdifferential, which is introduced in this Section.

Definition A.52 (Subdifferential)**.** Let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be proper. The *subdifferential* of f is the set-valued operator

$$\partial f : \mathcal{X} \to 2^{\mathcal{X}} : x \mapsto \{u \in \mathcal{X} \mid \langle y - x, u\rangle + f(x) \leq f(y)\ \forall y \in \mathcal{X}\}\,. \tag{A.36}$$

Let $x \in \mathcal{X}$, then f is *subdifferentiable* at x if $\partial f(x) \neq \emptyset$. The elements of $\partial f(x)$ are the *subgradients* of f at x.

By the definition it follows that a vector $u \in \mathcal{X}$ is a subgradient of the proper function f at $x \in \mathrm{dom}(f)$ if the continuous affine functional $y \mapsto \langle y - x, u\rangle + f(x)$, which coincides with $f(x)$ at x, minorises f. That is, u is the "slope" of a continuous affine minorant of f that coincides with f at x. Figure A.5 gives a graphical representation of this interpretation of subdifferential.
Note that in general there can be more than one subgradient at a point $x \in \mathrm{dom}(f)$, and indeed in the Figure there are many possible lines that underestimate f with different slopes.

The importance of subdifferentiation in convex analysis is highlighted by the following Theorem.

Theorem A.53 (Fermat's rule [3, Theorem 16.2])**.** *Let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be proper. Then*

$$\arg\min f = \mathrm{zer}\,\partial f = \{x \in \mathcal{X} \mid 0 \in \partial f(x)\}.$$

Therefore finding the minimisers of a proper function f is equivalent to finding the zeros of the subdifferential ∂f of f. There is an analogy with the case of differentiable functions, in which a necessary condition for a point to be a minimiser of f is that the gradient be zero in that point.

In the following some properties of the subdifferential are reported that hold in general for convex and non-convex functions.

Proposition A.54 ([3, Proposition 16.3])**.** *Let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ be proper and let $x \in \mathrm{dom}(f)$. Then the following hold*

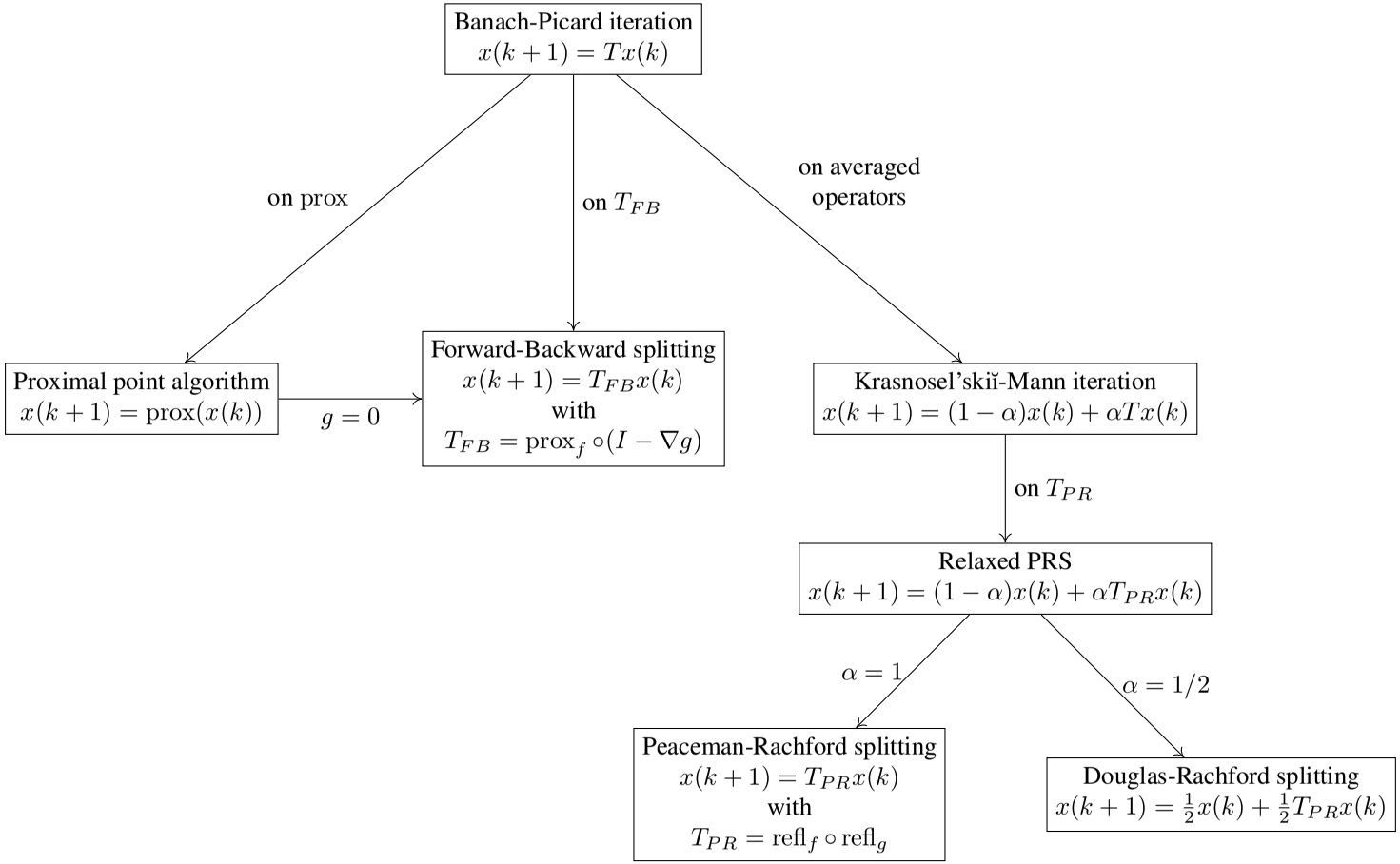


FIGURE A.4: Relationships between fixed point algorithms.

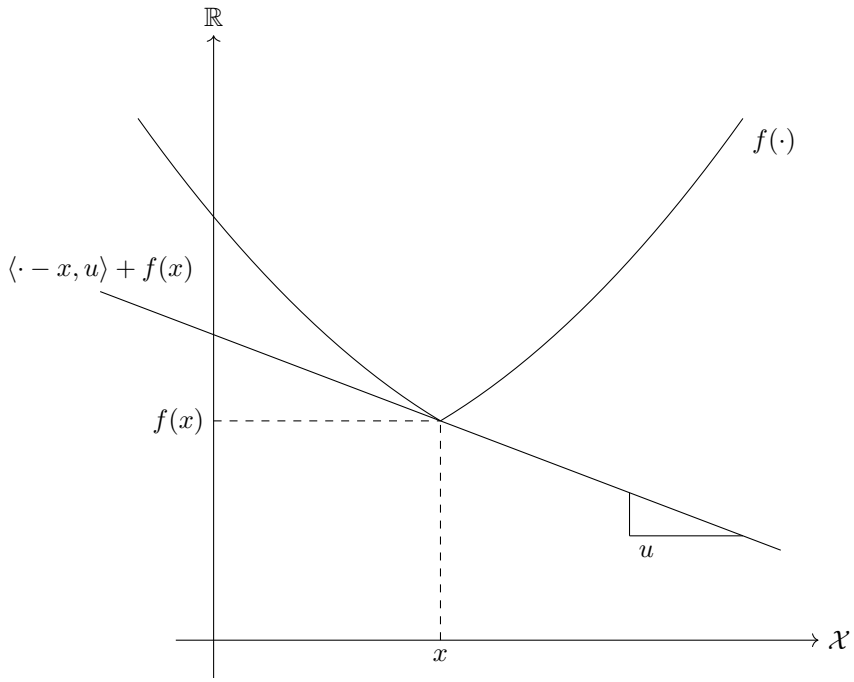


FIGURE A.5: Graphical representation of subdifferential.

(i). $\mathrm{dom}(\partial f) \subset \mathrm{dom}(f)$,

(ii). $\partial f(x)$ *is closed and convex.*

Proposition A.55. *Let* $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ *be proper, let* $x, u \in \mathcal{X}$*. Then* $u \in \partial f(x)$ *if and only if* $f(x) + f^*(u) = \langle x, u \rangle$*, which in turn implies that* $x \in \partial f^*(u)$*.*

Which means that the subgradients of f at x are those points u for which the Fenchel-Young inequality (A.3) becomes an equality.

Example A.5. *The following properties hold.*

- *Let* $f = \frac{1}{2} \|\cdot\|^2$ *then* $\partial f = I$*.*
- *Let* $f = \|\cdot\|_1$ *then*

$$\partial \|\cdot\|_1 (x) = \begin{cases} \frac{x}{\|x\|_1}, & \text{if } x \neq 0 \\ B(0;1), & \text{if } x = 0 \end{cases}$$

  where $B(0;1)$ *is the ball of unitary radius centred in the origin.*

If f is closed, proper and convex, then stronger results hold for the subdifferential.

Proposition A.56. *Let* f *be a closed, proper and convex function. Then* $(\partial f)^{-1} = \partial f^*$*.*

Example A.6. *Let* f *be a closed, proper and convex function and let* $\gamma > 0$*. Then*

$$\partial \left(f + (\gamma/2) \|\cdot\|^2 \right) = \partial f + \gamma I.$$

The following important Proposition shows the interpretation of the proximal operator in terms of the subdifferential of a convex function.

Proposition A.57. *Let f be closed, proper and convex, let $x, u \in \mathcal{X}$ and let $\rho > 0$. Then*

$$u = \text{prox}_{\rho f}(x) \quad \Leftrightarrow \quad \frac{1}{\rho}(x - u) \in \partial f(u) \tag{A.37}$$

or equivalently

$$\text{prox}_{\rho f} = (I + \rho \partial f)^{-1}. \tag{A.38}$$

As mentioned already, often it is required to minimise the sum of two convex functions, which can be simplified by the following property.

Proposition A.58. *Let f, g be closed, proper and convex functions such that one of the following holds*

(i). $\text{dom}(f) \cap \text{int dom}(g) \neq \emptyset$,

(ii). $\text{dom}(g) = \mathcal{X}$.

Then $\partial(f + g) = \partial f + \partial g$.

Remark A.59. Notice that if g is continuously differentiable, then Proposition A.58 states that

$$\partial(f + g)(x) = \partial f(x) + \nabla g(x) = \{y + \nabla g(x) \mid y \in \partial f(x)\}$$

that is, the subgradients of f at each point x are translated by a quantity equal to $\nabla g(x)$.

Also very useful in practice is the next Proposition.

Proposition A.60. *Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be closed, proper and convex and let matrix $A \in \mathbb{R}^{m \times n}$. Then $\partial(f \circ A) = A^\top (\partial f \circ A)$.*

Turning now to convex optimisation, the gradient method defined in (A.35) above requires the objective function to be smooth. If this is not the case, subdifferential theory has been leveraged in order to design the so-called *sub-gradient method*, which substitutes the necessity of computing the gradient with that of computing a subgradient.

A.4 Monotone Operators

The previous Sections highlighted the relationship between convex optimisation, fixed point algorithms and subdifferentiability. A different framework in which these problems and many of the results reported in A.2.4 can be recast is the theory of monotone operators.

Definition A.61 (Set-valued operator)**.** Let $\mathcal{X}$ be a real Hilbert space, a *set-valued operator* A maps each element of $\mathcal{X}$ into a set of elements in $\mathcal{X}$, and is indicated with $A : \mathcal{X} \rightrightarrows \mathcal{X}$ or sometimes $A : \mathcal{X} \to 2^{\mathcal{X}}$.

Remark A.62. Note that set-valued operators are different from the operators defined in A.21 because the latter are maps from one point to another of the Hilbert space, while the former map in principle to a set of points.

Remark A.63 (Nomenclature). Some authors call set-valued operators simply operators and the one-to-one operators of Definition A.21 *mappings*. In the following when clear from the context the 'set-valued' modifier will be dropped.

Definition A.64 (Graph)**.** A set-valued operator $A : \mathcal{X} \rightrightarrows \mathcal{X}$ on the Hilbert space $\mathcal{X}$ is uniquely described by its *graph*, defined as

$$\text{gr}(A) = \{(x, u) \in \mathcal{X} \times \mathcal{X} \mid u \in A(x)\}. \tag{A.39}$$

Definition A.65 (Domain)**.** The *domain* of a set-valued operator $A : \mathcal{X} \rightrightarrows \mathcal{X}$ on the Hilbert space $\mathcal{X}$ is defined as

$$\mathrm{dom}(A) = \{x \in \mathcal{X} \mid A(x) \neq \emptyset\}. \tag{A.40}$$

Definition A.66 (Zeros)**.** The *zeros* of a set-valued operator $A : \mathcal{X} \rightrightarrows \mathcal{X}$ are defined as

$$\mathrm{zer}(A) = \{x \in \mathcal{X} \mid 0 \in A(x)\}. \tag{A.41}$$

The following Definitions will be important for drawing a parallel between set-valued operator theory and convex optimisation.

Definition A.67 (Monotone set-valued operators)**.** A set-valued operator $A : \mathcal{X} \rightrightarrows \mathcal{X}$ is said to be *monotone* if for any $(x, u), (y, v) \in \mathrm{gr}(A)$ it holds that

$$\langle x - y, u - v \rangle \geq 0. \tag{A.42}$$

Definition A.68 (Maximal monotone set-valued operators)**.** Formally a monotone set-valued operator $A : \mathcal{X} \rightrightarrows \mathcal{X}$ is said to be *maximal monotone*, or *maximally monotone*, if $\mathrm{gr}(A)$ is not a proper subset of the graph of any other monotone operator. This definition means that there does not exist another monotone operator $B : \mathcal{X} \rightrightarrows \mathcal{X}$ such that $\mathrm{gr}(B)$ properly contains $\mathrm{gr}(A)$, that is for any $(x, u) \in \mathcal{X} \times \mathcal{X}$

$$(x, u) \in \mathrm{gr}(A) \;\Leftrightarrow\; \langle x - y, u - v \rangle \geq 0 \;\text{ for any }\; (y, v) \in \mathcal{X} \times \mathcal{X}. \tag{A.43}$$

In other words it is not possible to find a point $(x, u) \in \mathcal{X} \times \mathcal{X}$ for which is verified the condition of A.67 that does not belong to the graph of A.

Figure A.6 gives a graphical representation of monotone and maximally monotone set-valued operators with $\mathcal{X} = \mathbb{R}$.

Example A.7. *The subdifferential of a proper function f is monotone, and it is maximally monotone if f is also closed and convex.*

Definition A.69 (Inverse of set-valued operator)**.** Let $A : \mathcal{X} \rightrightarrows \mathcal{X}$ be a set-valued operator, the inverse operator is the operator A^{-1} such that $x \in A^{-1}(u)$ if and only if $u \in A(x)$ for any pair $(x, u) \in \mathcal{X} \times \mathcal{X}$.

Definition A.70 (Resolvent and reflected resolvent)**.** The *resolvent* of a monotone operator $A : \mathcal{X} \rightrightarrows \mathcal{X}$ is defined as

$$J_A = (I + A)^{-1}, \tag{A.44}$$

and the *reflected resolvent* as $R_A = 2J_A - I$.

An important example of resolvent operator is

$$J_{\rho \partial f} = \mathrm{prox}_{\rho f} \tag{A.45}$$

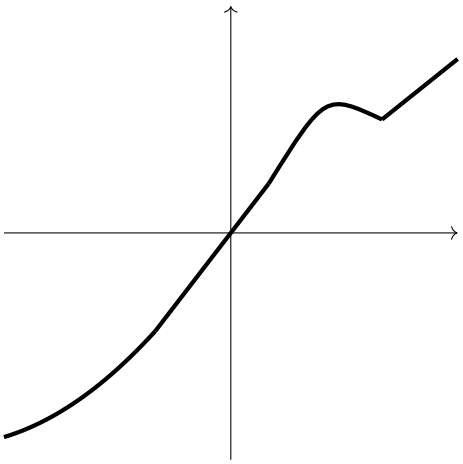
for a closed, proper and convex function $f : \mathcal{X} \to \mathbb{R}$. In this case therefore $R_{\rho \partial f} = \mathrm{refl}_{\rho f}$.

In the following some properties of the resolvent operator are reported.

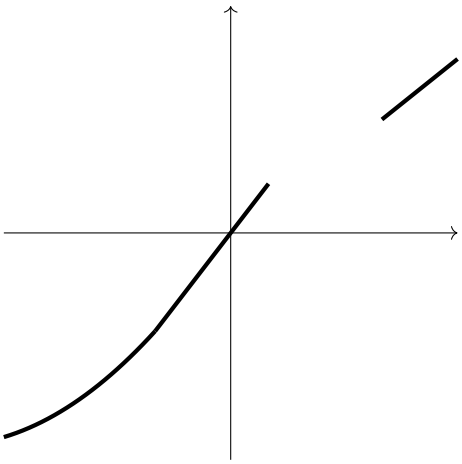
Proposition A.71. *Let $A : \mathcal{X} \rightrightarrows \mathcal{X}$ be a monotone set-valued operator, then J_A is single-valued, meaning that it is a mapping, and non-expansive. Moreover if A is maximal then*

(i). $\mathrm{dom}(J_A) = \mathcal{X}$,

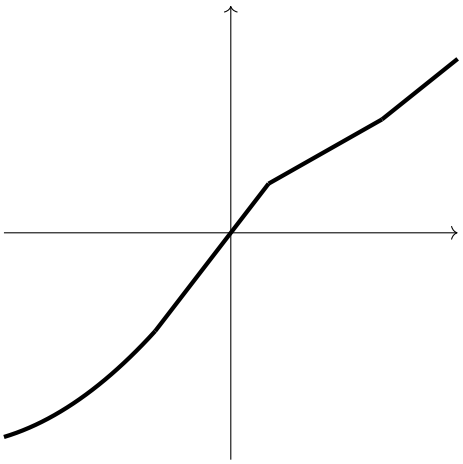
(ii). J_A and $I - J_A$ are firmly non-expansive and maximal monotone both,



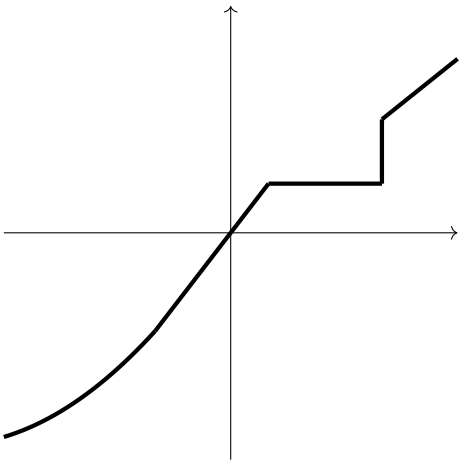
(A) Non-monotone operator.



(B) Monotone but not maximal.



(C) Maximal monotone operator, example 1.



(D) Maximal monotone operator, example 2.

FIGURE A.6: Some examples of set-valued operators with different properties.

(iii). the reflected resolvent R_A is non-expansive.

Proposition A.72 (Resolvent identity)**.** *The resolvent of a monotone set-valued operator satisfies the identity $J_A + J_{A^{-1}} = I$. If A is the subdifferential of a closed, proper and convex function then this identity implies the Moreau decomposition A.45.*

Minty's theorem traces now a first parallel between monotone set-valued operators and firmly non-expansive operators.

Theorem A.73 (Minty [4, Fact 1.2])**.** *Let $T : \mathcal{X} \to \mathcal{X}$ be a firmly non-expansive operator (or mapping), $A : \mathcal{X} \rightrightarrows \mathcal{X}$ a maximally monotone set-valued operator. Then*

(i). $B = T^{-1} - I$ is maximally monotone and $J_B = T$,

(ii). J_A is firmly non-expansive and $A = J_A^{-1} - I$.

Corollary A.74. *An operator $T : \mathcal{X} \to \mathcal{X}$ is firmly non-expansive if and only if it is the resolvent of a maximally monotone set-valued operator $A : \mathcal{X} \rightrightarrows \mathcal{X}$.*

Proposition A.75. *Let $T : \mathcal{X} \to \mathcal{X}$ be an α-averaged operator, then it is also monotone. Moreover, if $\alpha \in (0, 1/2]$ then it is maximally monotone.*

The following result draw a parallel between fixed point algorithms and monotone operator theory.

Proposition A.76. *Let $A : \mathcal{X} \rightrightarrows \mathcal{X}$ be monotone, then* $\mathrm{fix}(J_A) = \mathrm{zer}(A)$.

Proposition A.77. *Let $A, B : \mathcal{X} \rightrightarrows \mathcal{X}$ be monotone set-valued operators, then*

$$\mathrm{zer}(A + B) = J_{\rho B}\left(\mathrm{fix}(R_{\rho A} R_{\rho B})\right) \tag{A.46}$$

where the "relaxed" resolvent is defined as $J_{\rho A} = (I + \rho A)^{-1}$.

Finally the Peaceman-Rachford splitting defined above can be formulated in term of monotone operators.

Proposition A.78 (Peaceman-Rachford splitting for monotone operators [3, Theorem 25.6])**.** *Let $A, B : \mathcal{X} \rightrightarrows \mathcal{X}$ be maximally monotone operators such that* $\mathrm{zer}(A + B) \neq \emptyset$. *Let $\{\alpha_k\}_{k\in\mathbb{N}}$ be a sequence in $[0, 1]$ such that $\sum_{k\in\mathbb{N}} \alpha_k(1-\alpha_k) = +\infty$, let $\rho > 0$ and $x(0) \in \mathcal{X}$. The Peaceman-Rachford splitting is characterised by the iterates*

$$\begin{aligned} \psi(k) &= J_{\rho B}(z(k)) \\ \xi(k) &= J_{\rho A}(2\psi(k) - z(k)) \\ z(k+1) &= z(k) + 2\alpha_k(\xi(k) - \psi(k)) \end{aligned}$$

and there exists $z^* \in \mathrm{fix}(R_{\rho A} R_{\rho B})$ such that

(i). $J_{\rho B}(z^*) \in \mathrm{zer}(A + B)$,

(ii). $\{z(k)\}_{k\in\mathbb{N}}$ weakly converges to z^*,

(iii). $\{\xi(k)\}_{k\in\mathbb{N}}$ and $\{\psi(k)\}_{k\in\mathbb{N}}$ weakly converge to $J_{\rho B}(z^*)$.

Appendix B

Stochastic Krasnosel'skiĭ-Mann Iteration

The following Appendix will review in detail the *stochastic Krasnosel'skiĭ-Mann iteration* (s-KM) employed in Chapter 4 to prove the convergence of the randomised ADMM. The s-KM consists in a Krasnosel'skiĭ-Mann iteration in which the subset of co-ordinates to be updated is randomly chosen at each step according to some probability space, and the Theorem stated and proved below guarantees that this fixed point algorithm indeed converges to a fixed point of the operator to which it is applied.
A first formulation of the s-KM that performs the update of a single random co-ordinate at each instant was presented in [57], and then further extended to the update of a random subset of co-ordinates in [7]. Note that similar results are derived in [25, 26] making use of the quasi-Fejér monotonicity concept, and allowing for each update to be inexact due to random errors.

Let $T : \mathcal{X} \to \mathcal{X}$ be a γ-averaged operator, with $\mathcal{X}$ a finite real Hilbert space and $\gamma \in (0, 1]$. Let $\mathcal{M} = \{1, \dots, M\}$ be the set of indices denoting the co-ordinates of each element in the Hilbert space $\mathcal{X}$, and $2^{\mathcal{M}}$ be the set of all possible combinations of elements in $\mathcal{M}$.
The first step is to define a new operator that allows for the averaged operator T to be applied to only a subset of co-ordinates. With $\mu \in 2^{\mathcal{M}}$ the collection of indices in $\mathcal{M}$ denoting which co-ordinates to updated, define the operator $T^{(\mu)} : \mathcal{X} \to \mathcal{X}$ such that

$$[T^{(\mu)}x]_i = \begin{cases} T_i x & \text{if } i \in \mu \\ x_i & \text{otherwise.} \end{cases} \tag{B.1}$$

Moreover, let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and define over it the random i.i.d. sequence $\{\mu_k\}_{k\in\mathbb{N}}$ such that $\mu_k : \Omega \to 2^{\mathcal{M}}$. Suppose that the following assumption holds.

Assumption B.1. For any $i \in \mathcal{M}$ there exists a set $\mu \in 2^{\mathcal{M}}$ such that $\mathbb{P}[\mu_1 = \mu] > 0$ holds for the sequence $\{\mu_k\}_{k\in\mathbb{N}}$.

Then the s-KM iteration is defined as

$$x(k+1) = (1 - \alpha_k)x(k) + \alpha_k T^{(\mu_{k+1})}x(k) \tag{B.2}$$

and the following Theorem guarantees its probabilistic convergence to a fixed point of the operator T.

Theorem B.2 ([7, Theorem 3])**.** *Let $T : \mathcal{X} \to \mathcal{X}$ be a γ-averaged operator with $\mathrm{fix}(T) \neq \emptyset$. Assume that for all $k \in \mathbb{N}$ the sequence of step-sizes $\{\alpha_k\}_{k\in\mathbb{N}}$ satisfies*

$$0 < \liminf_k \alpha_k \leq \limsup_k \alpha_k < \frac{1}{\gamma}. \tag{B.3}$$

Let $\{\mu_k\}_{k\in\mathbb{N}}$ be a random i.i.d. sequence on $2^{\mathcal{M}}$ for which Assumption B.1 holds. Then for any deterministic initial condition $x(0) \in \mathcal{X}$ the stochastic KM iteration (B.2) *converges almost surely to a random vector with support in* $\mathrm{fix}(T)$.

Remark B.3. Assumption B.1 requires that the random sequence of subsets of co-ordinates to be updated guarantees that any co-ordinate can be selected with a non-zero probability at the first iteration of the s-KM. The need for this rather technical condition will be clear in the following proof.

Remark B.4. The Theorem states that the s-KM produces with probability one a point that belongs to the set $\mathrm{fix}(T)$ therefore solving the fixed point problem. The probability distribution describing the likelihood of one fixed point to be the result of the s-KM depends on the particular structure of the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and the initial condition $x(0)$.

Proof. Let $U = (1-\alpha_k)I + \alpha_k T$, by the properties of averaged operators [3] and the fact that T is γ-averaged, then U is $(\alpha_k\gamma)$-averaged. Defining $U^{(\mu)} = (1-\alpha_k)I + \alpha_k T^{(\mu)}$, where the time index of μ_{k+1} is omitted for simplicity, the s-KM can be rewritten as $x(k+1) = U^{(\mu)}x(k)$.
Let $p_\mu = \mathbb{P}[\mu_1 = \mu]$ for any $\mu \in 2^{\mathcal{M}}$ and define the new inner product $x \bullet y = \sum_{i=1}^{M} q_i \langle x_i, y_i\rangle$ where $q_i^{-1} = \sum_{\mu\in 2^{\mathcal{M}}} p_\mu \iota_{i\in\mu}$ with $\iota_{i\in\mu} = 1$ if $i \in \mu$, 0 otherwise. The corresponding norm will be denoted as $\|\!|x|\!\|^2 = x \bullet x$.

Conditionally to the sigma-field $\mathcal{F}^k = \sigma(\mu_1, \dots, \mu_k)$, that is, on the past history of the selected co-ordinates subsets, the following chain of equality holds.

$$
\begin{aligned}
\mathbb{E}\Big[\|\!|x(k+1) - x^*|\!\|^2 \big| \mathcal{F}^k\Big] &= \sum_{\mu\in 2^{\mathcal{M}}} p_\mu \left\|\!\left| U^{(\mu)}x(k) - x^* \right|\!\right\|^2 \\
&= \sum_{\mu\in 2^{\mathcal{M}}} p_\mu \sum_{i\in\mu} q_i \left\|U_i x(k) - x_i^*\right\|^2 + \sum_{\mu\in 2^{\mathcal{M}}} p_\mu \sum_{i\notin\mu} q_i \left\|x_i(k) - x_i^*\right\|^2 \\
&= \sum_{\mu\in 2^{\mathcal{M}}} p_\mu \sum_{i\in\mu} q_i \left(\left\|U_i x(k) - x_i^*\right\|^2 - \left\|x_i(k) - x_i^*\right\|^2\right) + \|\!|x(k) - x^*|\!\|^2 \\
&= \sum_{i=1}^{M} \left(\left\|U_i x(k) - x_i^*\right\|^2 - \left\|x_i(k) - x_i^*\right\|^2\right) + \|\!|x(k) - x^*|\!\|^2 \\
&= \left(\left\|Ux(k) - x^*\right\|^2 - \left\|x(k) - x^*\right\|^2\right) + \|\!|x(k) - x^*|\!\|^2
\end{aligned}
$$

where the definition of $\|\!|\cdot|\!\|^2$ was used to derive the second equality and to derive the third the term $\sum_{\mu\in 2^{\mathcal{M}}} p_\mu \sum_{i\in\mu} q_i \left\|x_i(k) - x_i^*\right\|^2$ was added and subtracted. Moreover, the second to last equation was derived applying the definition of the coefficients q_i.

Remark B.5. Note that the technical condition of Assumption B.1 ensures that each coefficient q_i^{-1} be strictly positive, and thus to avoid a division by zero.

Since U is an $(\alpha_k\gamma)$-averaged operator obtained by averaging the operator T, then the set of fixed points of U coincides with that of T, see Proposition A.35, and so $x^* \mathrm{fix}(U)$. Moreover, by Proposition A.30 it holds that

$$
\left\|Ux(k) - x^*\right\|^2 - \left\|x(k) - x^*\right\|^2 \leq -\frac{1-\alpha_k\gamma}{\alpha_k\gamma} \left\|(I-U)x(k)\right\|^2
$$

and by the fact that $I - U = \alpha_k(I - T)$ it follows

$$
\mathbb{E}\left[\|\!|x(k+1) - x^*|\!\|^2 \big| \mathcal{F}^k\right] \leq \|\!|x(k) - x^*|\!\|^2 - \alpha_k(1-\alpha_k\gamma) \left\|(I-T)x(k)\right\|^2 \tag{B.4}
$$

which proves that the sequence of distances from the fixed point $\|\|x(k) - x^*\|\|^2$ is a *non-negative supermartingale* with respect to the filtration $\mathcal{F}^k$. Therefore it has been proved that the sequence $\|\|x(k) - x^*\|\|^2$ converges with probability one towards a random vector that is finite almost everywhere.
In practice this result implies that the expected value of the error at time $k+1$ is smaller than the value of the error at the previous step, which means that the error is decreasing almost surely and moreover, by [86, Theorem 1], that it converges to a finite value[1].

The following part of the proof aims now to exploit inequality (B.4) to show that indeed the sequence $\{x(k)\}_{k\in\mathbb{N}}$ converges to a single fixed point of T with probability one.
Consider a countable dense subset X of $\mathrm{fix}(T)$, then by the result obtained above, for any $\bar{x} \in X$ there is a probability one set on which $\|\|x(k) - \bar{x}\|\| \to d_{\bar{x}} \in [0, \infty)$.
Let $\epsilon > 0$ and pick $\bar{x} \in X$ such that $\|\|x - \bar{x}\|\| \leq \epsilon$, with probability one therefore it follows that

$$\|\|x(k) - x^*\|\| \leq \|\|x(k) - \bar{x}\|\| + \|\|\bar{x} - x^*\|\| \leq d_{\bar{x}} + 2\epsilon \tag{B.5}$$

for k sufficiently large. Similarly, $\|\|x(k) - x^*\|\| \geq d_{\bar{x}} - 2\epsilon$ for k large enough. Since these results hold for any $x^* \in \mathrm{fix}(T)$ and by the fact that $d_{\bar{x}} \in [0, \infty)$, it follows that

R1 there is a probability one set on which $\|\|x(k) - x^*\|\|$ converges for every $x^* \in \mathrm{fix}(T)$.

This is a consequence of the fact that the error $\|\|x(k) - x^*\|\|$ has an upper and lower bounds almost everywhere finite and arbitrarily close to $d_{\bar{x}}$, since ϵ can be any infinitesimally small positive number.

Taking the expectation on both sides of inequality (B.4) and iterating it in time for k that goes to infinity, the following inequality can be derived

$$\sum_{k=0}^{\infty} \alpha_k(1 - \alpha_k\gamma)\mathbb{E}[\|(I - T)x(k)\|^2] \leq \|\|x(0) - x^*\|\|^2. \tag{B.6}$$

By condition (B.3) the step-sizes are positive numbers strictly smaller than one and hence $\sum_{k=0}^{\infty} \alpha_k(1 - \alpha_k\gamma) = +\infty$. This in turn implies that $\mathbb{E}[\|(I - T)x(k)\|^2]$ is finite since in the inequality (B.6) the right-hand side is finite. Furthermore, by Markov's inequality and Borel-Cantelli's lemma the following result is proved

R2 $(I - T)x(k) \to 0$ almost surely.

So far result **R1** proves that the error $\|\|x(k) - x^*\|\|$ is bounded, while result **R2** that the accumulation points of the sequence $\{x(k)\}_{k\in\mathbb{N}}$ are in the set $\mathrm{fix}(T)$, since T is continuous because it is averaged. The last step is then to prove that the accumulation points of $\{x(k)\}_{k\in\mathbb{N}}$ reduce to one single point to prove that indeed the sequence converges with probability one to a fixed point of T.
But assume that $\bar{x}^*$ is an accumulation point, then by **R1** the error $\|\|x(k) - \bar{x}^*\|\|$ converges, which means that $\lim \|\|x(k) - \bar{x}^*\|\| = \liminf \|\|x(k) - \bar{x}^*\|\| = 0$ proving that $\bar{x}^*$ is unique. □

[1]Note that this theorem is central in the proof of convergence of the asynchronous optimisation algorithm ARock [79].

Appendix C

Security Analysis of ADMM

This Appendix briefly outlines a possible exploit that could be used to compromise the privacy of the data held by one node at the hand of a malicious node or man-in-the-middle attacker. It has no pretence of being a thorough investigation, but only a demonstration that the ADMM in the formulation of Algorithm 1 might have some security flaws.
Hereafter the attention is restricted to the case of quadratic cost functions with scalar primal variables x_i, that is

$$f_i(x_i) = a_i x_i^2 + b_i x_i + c_i, \qquad a_i, b_i, c_i \in \mathbb{R}. \tag{C.1}$$

Moreover, the step-size is assumed to be constant throughout all computations.
The objective of an attacker is therefore to reconstruct the data a_i and b_i by using the (possibly intercepted) transmissions $q_{i\to j}(k)$ sent by node i to each of its neighbours $j \in \mathcal{N}_i$.

Recalling Algorithm 1 and using the costs (C.1), the iterates of the Algorithm become

$$x_i(k) = \frac{1}{2a_i + \rho|\mathcal{N}_i|}\left(\sum_{l\in\mathcal{N}_i} z_{li}(k) - b_i\right) \tag{C.2}$$

$$z_{ji}(k+1) = (1-\alpha)z_{ji}(k) + \alpha q_{j\to i}(k) \tag{C.3}$$

where

$$q_{i\to j}(k) = -z_{ji}(k) + 2\rho x_i(k). \tag{C.4}$$

Notice that the data a_i and b_i play a role only in the update (C.2), while the an attacker can get hold of the packets $q_{i\to j}$'s only. Moreover it is assumed that the attacker knows the $q_{j\to i}$ packets sent by the neighbours of i, either because the attacker is a neighbour or because tapping the channel between i and j gives access to both ingoing and outgoing data.

Plugging (C.2) into (C.4) and rearranging the terms yields

$$2a_i q_{i\to j}(k) + z_{ji}(k) - 2\rho\sum_{l\in\mathcal{N}_i} z_{li}(k) + 2\rho b_i = -\rho|\mathcal{N}_i|q_{i\to j}(k) = c(k) \tag{C.5}$$

where $c(k)$ is known to the attacker and the terms on the left-hand side are not.

Reconstruction of a_i In order to reconstruct a_i the attacker needs only to control two of i's neighbours, thanks to the following scheme.
Consider (C.5) from the point of view of neighbours j and m

$$2a_i q_{i\to j}(k) + z_{ji}(k) - 2\rho\sum_{l\in\mathcal{N}_i} z_{li}(k) + 2\rho b_i = c_j(k)$$

$$2a_i q_{i\to m}(k) + z_{mi}(k) - 2\rho\sum_{l\in\mathcal{N}_i} z_{li}(k) + 2\rho b_i = c_m(k)$$

and subtracting them it follows

$$2a_i(q_{i\to j}(k) - q_{i\to m}(k)) + (z_{ji}(k) - z_{mi}(k)) = c_j(k) - c_m(k). \tag{C.6}$$

Now consider that (C.3) holds and substitute it into (C.6) evaluated at time $k+1$, which yields

$$\begin{aligned} 2a_i(q_{i\to j}(k+1) - q_{i\to m}(k+1)) + (1-\alpha)(z_{ji}(k) - z_{mi}(k)) = \\ = c_j(k) - c_m(k) - \alpha(q_{j\to i}(k) - q_{m\to i}(k)) \end{aligned} \tag{C.7}$$

where by the assumptions the right-hand side is known.

Finally, (C.6) and (C.7) are two equations in the two unknowns a_i and $(z_{ji}(k) - z_{mi}(k))$ and the objective of reconstructing a_i is achieved.

Reconstruction of b_i As in the case of a_i, the attacker needs only to hijack, or intercept transmission to and from, two neighbours of i.
Plugging (C.3) into (C.5) evaluated at time $k+1$, and assuming that a_i has already been computed, yields

$$(1-\alpha)z_{ji}(k) - 2\rho \sum_{l\in\mathcal{N}_i} z_{li}(k+1) + 2\rho b_i = d_j(k+1). \tag{C.8}$$

Moreover $(z_{ji}(k) - z_{mi}(k)) = e(k)$ is known, by the procedure above. Hence evaluating equation (C.8) for neighbours j and m gives the following system of two equations

$$\begin{aligned} (1-\alpha)z_{mi}(k) - 2\rho \sum_{l\in\mathcal{N}_i} z_{li}(k+1) + 2\rho b_i = d_j(k+1) - (1-\alpha)e(k) \\ (1-\alpha)z_{mi}(k) - 2\rho \sum_{l\in\mathcal{N}_i} z_{li}(k+1) + 2\rho b_i = d_m(k+1) \end{aligned}$$

in the two unknowns b_i and $(1-\alpha)z_{mi}(k) - 2\rho\sum_{l\in\mathcal{N}_i} z_{li}(k+1)$, making it possible to reconstruct b_i.

To conclude, in order to compromise the privacy of the data stored by node i a malicious agent need only get access to the packets $q_{i\to j}$ and $q_{j\to i}$ exchanged by i and two of its neighbours over at least two consecutive time instants. This could be accomplished for instance using a man-in-the-middle attack.

Bibliography

[1] Nicola Bastianello, Ruggero Carli, Luca Schenato, and Marco Todescato. A partition-based implementation of the relaxed admm for distributed convex optimization over lossy networks. In *2018 IEEE 57th Annual Conference on Decision and Control (CDC) [submitted]*. IEEE, 2018.

[2] Nicola Bastianello, Marco Todescato, Ruggero Carli, and Luca Schenato. Distributed optimization over lossy networks via relaxed peaceman-rachford splitting: a robust admm approach. In *European Control Conference (ECC) 2018*. IEEE, 2018.

[3] Heinz H. Bauschke and Patrick L. Combettes. *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. Springer Publishing Company, Incorporated, 1st edition, 2011.

[4] Heinz H. Bauschke, Sarah M. Moffat, and Xianfu Wang. Firmly nonexpansive mappings and maximally monotone operators: Correspondence and duality. *Set-Valued and Variational Analysis*, 20(1):131–153, 2012.

[5] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[6] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*, volume 23. Prentice hall Englewood Cliffs, NJ, 1989.

[7] Pascal Bianchi, Walid Hachem, and Franck Iutzeler. A coordinate descent primal-dual algorithm and application to distributed asynchronous optimization. *IEEE Transactions on Automatic Control*, 61(10):2947–2957, 2016.

[8] Nicoletta Bof, Ruggero Carli, Giuseppe Notarstefano, Luca Schenato, and Damiano Varagnolo. Newton-raphson consensus under asynchronous and lossy communications for peer-to-peer networks. *arXiv preprint arXiv:1707.09178*, 2017.

[9] Daniel Boley. Local linear convergence of the alternating direction method of multipliers on quadratic or linear programs. *SIAM Journal on Optimization*, 23(4):2183–2207, 2013.

[10] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122, 2011.

[11] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[12] James H. Bramble, Joseph E. Pasciak, and Apostol T. Vassilev. Analysis of the inexact uzawa algorithm for saddle point problems. *SIAM Journal on Numerical Analysis*, 34(3):1072–1092, 1997.

[13] Ruggero Carli and Giuseppe Notarstefano. Distributed partition-based optimization via dual decomposition. In *2013 IEEE 52nd Annual Conference on Decision and Control (CDC)*, pages 2979–2984. IEEE, 2013.

[14] Ruggero Carli, Giuseppe Notarstefano, Luca Schenato, and Damiano Varagnolo. Analysis of newton-raphson consensus for multi-agent convex optimization under asynchronous and lossy communications. In *2015 IEEE 54th Annual Conference on Decision and Control (CDC)*, pages 418–424. IEEE, 2015.

[15] Ruggero Carli, Giuseppe Notarstefano, Luca Schenato, and Damiano Varagnolo. Distributed quadratic programming under asynchronous and lossy communications via newton-raphson consensus. In *European Control Conference (ECC) 2015*, pages 2514–2520. IEEE, 2015.

[16] Tsung-Hui Chang, Mingyi Hong, Wei-Cheng Liao, and Xiangfeng Wang. Asynchronous distributed admm for large-scale optimization—part i: Algorithm and convergence analysis. *IEEE Transactions on Signal Processing*, 64(12):3118–3130, 2016.

[17] Tsung-Hui Chang, Mingyi Hong, and Xiangfeng Wang. Multi-agent distributed large-scale optimization by inexact consensus alternating direction method of multipliers. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6137–6141. IEEE, 2014.

[18] Tsung-Hui Chang, Mingyi Hong, and Xiangfeng Wang. Multi-agent distributed optimization via inexact consensus admm. *IEEE Transactions on Signal Processing*, 63(2):482–497, 2015.

[19] Tsung-Hui Chang, Wei-Cheng Liao, Mingyi Hong, and Xiangfeng Wang. Asynchronous distributed admm for large-scale optimization—part ii: Linear convergence analysis and numerical performance. *IEEE Transactions on Signal Processing*, 64(12):3131–3144, 2016.

[20] Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan. The direct extension of admm for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, 155(1-2):57–79, 2016.

[21] Caihua Chen, Min Li, Xin Liu, and Yinyu Ye. On the convergence of multi-block alternating direction method of multipliers and block coordinate descent method. *arXiv preprint arXiv:1508.00193*, 2015.

[22] Caihua Chen, Yuan Shen, and Yanfei You. On the convergence analysis of the alternating direction method of multipliers with three blocks. In *Abstract and Applied Analysis*, volume 2013. Hindawi, 2013.

[23] Dai-Qiang Chen. Inexact alternating direction method based on newton descent algorithm with application to poisson image deblurring. *Signal, Image and Video Processing*, 11(1):89–96, 2017.

[24] Patrick L. Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point algorithms for inverse problems in science and engineering*, pages 185–212. Springer, 2011.

[25] Patrick L. Combettes and Jean-Christophe Pesquet. Stochastic quasi-fejér block-coordinate fixed point iterations with random sweeping. *SIAM Journal on Optimization*, 25(2):1221–1248, 2015.

[26] Patrick L. Combettes and Jean-Christophe Pesquet. Stochastic quasi-fejér block-coordinate fixed point iterations with random sweeping ii: Mean-square and linear convergence. *arXiv preprint arXiv:1704.08083*, 2017.

[27] Patrick L. Combettes and Valérie R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[28] Laurent Condat. A primal–dual splitting method for convex optimization involving lipschitzian, proximable and linear composite terms. *Journal of Optimization Theory and Applications*, 158(2):460–479, 2013.

[29] Christian Conte, Tyler Summers, Melanie N. Zeilinger, Manfred Morari, and Colin N. Jones. Computational aspects of distributed optimization in model predictive control. In *2012 IEEE 51st Annual Conference on Decision and Control (CDC)*, pages 6819–6824. IEEE, 2012.

[30] Etienne Corman and Xiaoming Yuan. A generalized proximal point algorithm and its convergence rate. *SIAM Journal on Optimization*, 24(4):1614–1638, 2014.

[31] Damek Davis and Wotao Yin. Convergence rate analysis of several splitting schemes. In *Splitting Methods in Communication, Imaging, Science, and Engineering*, pages 115–163. Springer, 2016.

[32] Damek Davis and Wotao Yin. Faster convergence rates of relaxed peaceman-rachford and admm under regularity assumptions. *Mathematics of Operations Research*, 42(3):783–805, 2017.

[33] Damek Davis and Wotao Yin. A three-operator splitting scheme and its optimization applications. *Set-valued and Variational Analysis*, 25(4):829–858, 2017.

[34] Wei Deng and Wotao Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing*, 66(3):889–916, 2016.

[35] Paolo Di Lorenzo and Gesualdo Scutari. Next: In-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

[36] Jim Douglas and Henry H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, 82(2):421–439, 1956.

[37] Jonathan Eckstein. Augmented lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. *RUTCOR Research Reports*, 32:3, 2012.

[38] Jonathan Eckstein and Dimitri P. Bertsekas. On the douglas—rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55(1-3):293–318, 1992.

[39] Tomaso Erseghe. A distributed and scalable processing method based upon admm. *IEEE Signal Processing Letters*, 19(9):563–566, 2012.

[40] Qiang Fu, Huahua Wang, and Arindam Banerjee. Bethe-admm for tree decomposition based parallel map inference. *arXiv preprint arXiv:1309.6829*, 2013.

[41] Daniel Gabay. Chapter ix applications of the method of multipliers to variational inequalities. In *Studies in Mathematics and its Applications*, volume 15, pages 299–331. Elsevier, 1983.

[42] Daniel Gabay and Bertrand Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 2(1):17–40, 1976.

[43] Euhanna Ghadimi, André Teixeira, Iman Shames, and Mikael Johansson. Optimal parameter selection for the alternating direction method of multipliers (admm): quadratic problems. *IEEE Transactions on Automatic Control*, 60(3):644–658, 2015.

[44] Pontus Giselsson and Stephen Boyd. Diagonal scaling in douglas-rachford splitting and admm. In *2014 IEEE 53rd Annual Conference on Decision and Control (CDC)*, pages 5033–5039. IEEE, 2014.

[45] Pontus Giselsson and Stephen Boyd. Metric selection in fast dual forward–backward splitting. *Automatica*, 62:1–10, 2015.

[46] Pontus Giselsson and Stephen Boyd. Linear convergence and metric selection for douglas-rachford splitting and admm. *IEEE Transactions on Automatic Control*, 62(2):532–544, 2017.

[47] Roland Glowinski. On alternating direction methods of multipliers: a historical perspective. In *Modeling, Simulation and Optimization for Science and Technology*, pages 59–82. Springer, 2014.

[48] Roland Glowinski and A. Marroco. Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problèmes de dirichlet non linéaires. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique*, 9(R2):41–76, 1975.

[49] Tom Goldstein, Christoph Studer, and Richard Baraniuk. A field guide to forward-backward splitting with a fasta implementation. *arXiv preprint arXiv:1411.3406*, 2014.

[50] Vehbi C. Gungor, Bin Lu, and Gerhard P. Hancke. Opportunities and challenges of wireless sensor networks in smart grid. *IEEE Transactions on Industrial Electronics*, 57(10):3557–3564, 2010.

[51] Deren Han and Xiaoming Yuan. A note on the alternating direction method of multipliers. *Journal of Optimization Theory and Applications*, 155(1):227–238, 2012.

[52] Robert Hannah and Wotao Yin. On unbounded delays in asynchronous parallel fixed-point algorithms. *Journal of Scientific Computing*, pages 1–28, 2016.

[53] Bingsheng He and Xiaoming Yuan. On the $o(1/n)$ convergence rate of the douglas–rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 50(2):700–709, 2012.

[54] B.S. He, Hai Yang, and S.L. Wang. Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities. *Journal of Optimization Theory and applications*, 106(2):337–356, 2000.

[55] Mingyi Hong. A distributed, asynchronous and incremental algorithm for nonconvex optimization: An admm based approach. *arXiv preprint arXiv:1412.6058*, 2014.

[56] Mingyi Hong and Zhi-Quan Luo. On the linear convergence of the alternating direction method of multipliers. *Mathematical Programming*, 162(1-2):165–199, 2017.

[57] Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem. Asynchronous distributed optimization using a randomized alternating direction method of multipliers. In *2013 IEEE 52nd Annual Conference on Decision and Control (CDC)*, pages 3671–3676. IEEE, 2013.

[58] Franck Iutzeler, Pascal Bianchi, Philippe Ciblat, and Walid Hachem. Explicit convergence rate of a distributed alternating direction method of multipliers. *IEEE Transactions on Automatic Control*, 61(4):892–904, 2016.

[59] Franck Iutzeler and Julien M. Hendrickx. A generic online acceleration scheme for optimization algorithms via relaxation and inertia. *Optimization Methods and Software*, pages 1–23, 2017.

[60] Dušan Jakovetić, Joao Xavier, and José MF Moura. Fast distributed gradient methods. *IEEE Transactions on Automatic Control*, 59(5):1131–1146, 2014.

[61] Solmaz S. Kia, Jorge Cortés, and Sonia Martínez. Distributed convex optimization via continuous-time coordination algorithms with discrete-time communication. *Automatica*, 55:254–264, 2015.

[62] Mark Aleksandrovich Krasnosel'skiĭ. Two remarks on the method of successive approximations. *Uspekhi Matematicheskikh Nauk*, 10(1):123–127, 1955.

[63] Soomin Lee and Angelia Nedić. Asynchronous gossip-based random projection algorithms over networks. *IEEE Transactions on Automatic Control*, 61(4):953–968, 2016.

[64] Pierre-Louis Lions and Bertrand Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

[65] W. Robert Mann. Mean value methods in iteration. *Proceedings of the American Mathematical Society*, 4(3):506–510, 1953.

[66] Gonzalo Mateos, Juan Andrés Bazerque, and Georgios B. Giannakis. Distributed sparse linear regression. *IEEE Transactions on Signal Processing*, 58(10):5262–5276, 2010.

[67] Mehran Mesbahi and Magnus Egerstedt. *Graph Theoretic Methods in Multiagent Networks*. Princeton University Press, 2010.

[68] Carl D. Meyer. *Matrix Analysis and Applied Linear Algebra*, volume 71. SIAM, 2000.

[69] Aryan Mokhtari, Wei Shi, Qing Ling, and Alejandro Ribeiro. A decentralized second-order method with exact linear convergence rate for consensus optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):507–522, 2016.

[70] João F.C. Mota, João M.F. Xavier, Pedro M.Q. Aguiar, and Markus Püschel. D-admm: A communication-efficient distributed algorithm for separable optimization. *IEEE Transactions on Signal Processing*, 61(10):2718–2723, 2013.

[71] João F.C. Mota, João M.F. Xavier, Pedro M.Q. Aguiar, and Markus Püschel. Distributed optimization with local domains: Applications in mpc and network flows. *IEEE Transactions on Automatic Control*, 60(7):2004–2009, 2015.

[72] Angelia Nedić and Alex Olshevsky. Distributed optimization over time-varying directed graphs. *IEEE Transactions on Automatic Control*, 60(3):601–615, 2015.

[73] Angelia Nedić and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[74] Angelia Nedić, Asuman Ozdaglar, and Pablo A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 55(4):922–938, 2010.

[75] Robert Nishihara, Laurent Lessard, Ben Recht, Andrew Packard, and Michael Jordan. A general analysis of the convergence of admm. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 343–352, 2015.

[76] Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends® in Optimization*, 1(3):127–239, 2014.

[77] Panagiotis Patrinos, Lorenzo Stella, and Alberto Bemporad. Douglas-rachford splitting: Complexity estimates and accelerated variants. In *2014 IEEE 53rd Annual Conference on Decision and Control (CDC)*, pages 4234–4239. IEEE, 2014.

[78] Donald W. Peaceman and Henry H. Rachford, Jr. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, 3(1):28–41, 1955.

[79] Zhimin Peng, Yangyang Xu, Ming Yan, and Wotao Yin. Arock: an algorithmic framework for asynchronous parallel coordinate updates. *SIAM Journal on Scientific Computing*, 38(5):A2851–A2879, 2016.

[80] Zhimin Peng, Ming Yan, and Wotao Yin. Parallel and distributed sparse optimization. In *Conference on Signals, Systems and Computers, 2013 Asilomar*, pages 659–646. IEEE, 2013.

[81] Hung M. Phan. Linear convergence of the douglas–rachford method for two closed sets. *Optimization*, 65(2):369–385, 2016.

[82] Thomas Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *2011 IEEE International Conference on Computer Vision (ICCV)*, pages 1762–1769. IEEE, 2011.

[83] Nicholas G. Polson, James G. Scott, and Brandon T. Willard. Proximal algorithms in statistics and machine learning. *Statistical Science*, 30(4):559–581, 2015.

[84] Arvind U. Raghunathan and Stefano Di Cairano. Admm for convex quadratic programs: Linear convergence and infeasibility detection. *arXiv preprint arXiv:1411.7288*, 2014.

[85] Hugo Raguet, Jalal Fadili, and Gabriel Peyré. A generalized forward-backward splitting. *SIAM Journal on Imaging Sciences*, 6(3):1199–1226, 2013.

[86] Herbert Robbins and David Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In *Optimizing Methods in Statistics*, pages 233–257. Elsevier, 1971.

[87] R. Tyrrell Rockafellar. Augmented lagrangians and applications of the proximal point algorithm in convex programming. *Mathematics of Operations Research*, 1(2):97–116, 1976.

[88] Ralph Tyrell Rockafellar. *Convex Analysis*. Princeton University Press, 2nd edition, 1972.

[89] Ralph Tyrrell Rockafellar. Monotone operators and the proximal point algorithm. *SIAM Journal on Control and Optimization*, 14(5):877–898, 1976.

[90] Luca Schenato, Bruno Sinopoli, Massimo Franceschetti, Kameshwar Poolla, and S. Shankar Sastry. Foundations of control and estimation over lossy networks. *Proceedings of the IEEE*, 95(1):163–187, 2007.

[91] Thomas Sherson, Richard Heusdens, and W. Bastiaan Kleijn. Derivation and analysis of the primal-dual method of multipliers based on monotone operator theory. *arXiv preprint arXiv:1706.02654*, 2017.

[92] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

[93] Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Trans. Signal Processing*, 62(7):1750–1761, 2014.

[94] Bruno Sinopoli, Luca Schenato, Massimo Franceschetti, Kameshwar Poolla, Michael I. Jordan, and Shankar S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49(9):1453–1464, 2004.

[95] Konstantinos Slavakis, Georgios B. Giannakis, and Gonzalo Mateos. Modeling and optimization for big data analytics:(statistical) learning tools for our era of data deluge. *IEEE Signal Processing Magazine*, 31(5):18–31, 2014.

[96] Changkyu Song, Sejong Yoon, and Vladimir Pavlovic. Fast admm algorithm for distributed optimization with adaptive penalty. In *AAAI*, pages 753–759, 2016.

[97] Ruoyu Sun, Zhi-Quan Luo, and Yinyu Ye. On the expected convergence of randomly permuted admm. *arXiv preprint arXiv:1503.06387*, 2015.

[98] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[99] Marco Todescato, Nicoletta Bof, Guido Cavraro, Ruggero Carli, and Luca Schenato. Generalized gradient optimization over lossy networks for partition-based estimation. *arXiv preprint arXiv:1710.10829*, 2017.

[100] Marco Todescato, Guido Cavraro, Ruggero Carli, and Luca Schenato. A robust block-jacobi algorithm for quadratic programming under lossy communications. *IFAC-PapersOnLine*, 48(22):126–131, 2015.

[101] Bằng Công Vũ. A splitting algorithm for dual monotone inclusions involving cocoercive operators. *Advances in Computational Mathematics*, 38(3):667–681, 2013.

[102] Huahua Wang and Arindam Banerjee. Bregman alternating direction method of multipliers. In *Advances in Neural Information Processing Systems*, pages 2816–2824, 2014.

[103] Ermin Wei and Asuman Ozdaglar. On the $o(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers. In *2013 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 551–554. IEEE, 2013.

[104] Ermin Wei, Asuman Ozdaglar, and Ali Jadbabaie. A distributed newton method for network utility maximization–i: Algorithm. *IEEE Transactions on Automatic Control*, 58(9):2162–2175, 2013.

[105] Ermin Wei, Asuman Ozdaglar, and Ali Jadbabaie. A distributed newton method for network utility maximization–part ii: Convergence. *IEEE Transactions on Automatic Control*, 58(9):2176–2188, 2013.

[106] Meng Wen, Shigang Yue, Yuchao Tang, and Jigen Peng. A stochastic coordinate descent primal-dual algorithm with dynamic stepsize for large-scale composite optimization. *arXiv preprint arXiv:1604.04172*, 2016.

[107] Jinming Xu, Shanying Zhu, Yeng Chai Soh, and Lihua Xie. Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes. In *2015 IEEE 54th Annual Conference on Decision and Control (CDC)*, pages 2055–2060. IEEE, 2015.

[108] Jinming Xu, Shanying Zhu, Yeng Chai Soh, and Lihua Xie. A forward-backward bregman splitting scheme for regularized distributed optimization problems. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 1093–1098. IEEE, 2016.

[109] Zheng Xu, Mário A.T. Figueiredo, Xiaoming Yuan, Christoph Studer, and Tom Goldstein. Adaptive relaxed admm: Convergence theory and practical implementation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7234–7243, July 2017.

[110] Zheng Xu, Gavin Taylor, Hao Li, Mário AT Figueiredo, Xiaoming Yuan, and Tom Goldstein. Adaptive consensus admm for distributed optimization. In *International Conference on Machine Learning*, pages 3841–3850, 2017.

[111] Filippo Zanella, Damiano Varagnolo, Angelo Cenedese, Gianluigi Pillonetto, and Luca Schenato. Newton-raphson consensus for distributed convex optimization. In *2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 5917–5922. IEEE, 2011.

[112] Filippo Zanella, Damiano Varagnolo, Angelo Cenedese, Gianluigi Pillonetto, and Luca Schenato. Asynchronous newton-raphson consensus for distributed convex optimization. *IFAC Proceedings Volumes*, 45(26):133–138, 2012.

[113] Guoqiang Zhang and Richard Heusdens. Bi-alternating direction method of multipliers. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3317–3321. IEEE, 2013.

[114] Guoqiang Zhang, Richard Heusdens, and W. Bastiaan Kleijn. On the convergence rate of the bi-alternating direction method of multipliers. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3869–3873. IEEE, 2014.

[115] Lixian Zhang, Huijun Gao, and Okyay Kaynak. Network-induced constraints in networked control systems—a survey. *IEEE Transactions on Industrial Informatics*, 9(1):403–416, 2013.

[116] Ruiliang Zhang and James Kwok. Asynchronous distributed admm for consensus optimization. In *International Conference on Machine Learning*, pages 1701–1709, 2014.