Alternating Direction Method of Multipliers (ADMM) Techniques for Embedded Mixed-Integer Quadratic Programming and Applications

By

Jiaqi Liu
B.ASc., University of Toronto, 2018

A Project Report Submitted in Partial fulfillment of the
Requirements for the Degree of

MASTER OF ENGINEERING

in the Department of Electrical and Computer Engineering

SUPERVISORY COMMITTEE

Alternating Direction Method of Multipliers (ADMM) Techniques for Embedded Mixed-Integer Quadratic Programming and Applications

by

Jiaqi Liu

B.ASc., University of Toronto, 2018

Supervisory Committee

Dr. Tao Lu, Department of Electrical and Computer Engineering, University of Victoria (Supervisor)

Dr. Wu-Sheng Lu, Department of Electrical and Computer Engineering, University of Victoria (Departmental Member)

Abstract

In this project, we delve into an important class of constrained nonconvex problems known as mixed-integer quadratic programming (MIQP). The popularity of MIQP is primarily due to the fact that many real-world problems can be described via MIQP models. The development of efficient MIQP algorithms has been an active and rapidly evolving field of research. As a matter of fact, previously well-known techniques for MIQP problems are found unsuitable for large-scale or online MIQP problems where algorithm's computational efficiency is a crucial factor. In this regard, the alternating direction method of multipliers (ADMM) as a heuristic has shown to offer satisfactory suboptimal solutions with much improved computational complexity relative to global solvers based on for example branch-and-bound. This project provides the necessary details required to understand the ADMM-based algorithms as applied to MIQP problems. Three illustrative examples are also included in this project to demonstrate the effectiveness of the ADMM algorithm through numerical simulations and performance comparisons.

Keywords: Mixed integer quadratic programming (MIQP), Alternating direction method of multipliers (ADMM), MATLAB.

Table of Contents

List of Tables

List of Figures

Abbreviations

ADMM	Alternating Direction Method of Multipliers
BIP	Binary Integer Programming
CP	Convex Programming
IP	Integer Programming
KKT	Karush–Kuhn–Tucker
NP	Nondeterministic Polynomial
MILP	Mixed-Integer Linear Programming
MIQP	Mixed-Integer Quadratic Programming
MIP	Mixed-Integer Programing
QP	Quadratic Programming

Acknowledgements

First of all, I would like to thank Dr. Tao Lu and Dr. Wu-Sheng Lu, for their guidance through each stage of the process. It is no exaggeration to say that without their help, I could not have finished my graduation project.

Next, I would like to express my sincere thanks to the course instructors in University of Victoria. Thanks to their teaching, which gave me a deeper understanding of wireless communication, microwave and machine learning.

In addition, I am very glad that I have met some good friends and classmates in Victoria, thank them for their help in my study and life.

Finally, I really appreciate my family for their unselfish supports all the time.

Dedication

To schools

IVY Experimental High School
where I received my high school degree

and

University of Toronto
where I received my bachelor degree

Chapter 1

Introduction

Research on optimization has taken a giant leap with the advent of the digital computer in the early fifties. In recent years, optimization techniques advanced rapidly and considerable progresses have been achieved. At the same time, digital computers became faster, more versatile, and more efficient. As a consequence, it is now possible to solve complex optimization problems which were thought intractable only a few years ago [1].

Optimization problems occur in most disciplines including engineering, physics, mathematics, economics, commerce, and social sciences, etc. Typical areas of application are modeling, characterization, and design of devices, circuits, and systems; design of instruments and equipment; design of process control; approximation theory, curve fitting, solution of systems of equations; forecasting, production scheduling, and quality control; inventory control, accounting, budgeting, etc. Some recent innovations rely crucially on optimization techniques, for example, adaptive signal processing, machine learning, and neural networks [2].

In this project, we examine solution techniques for a class of nonconvex problems known as *mixed-integer quadratic programming* (MIQP) where a quadratic objective function is minimized subject to conventional linear constraints and a part of the decision variables belonging to a certain integer (such as Boolean) set. Developing efficient algorithms for MIQP has been a field of current research in optimization as it finds applications in admission control [3], economic dispatch [4], scheduling [5], and hybrid vehicle control [6], etc. An effective technical tool in the dealings with embedded MIQP problems is the algorithm of alternating direction method of multipliers (ADMM) [7]-[10].

In this introductory chapter, we provide some background information concerning integer programming in general and MIQP in particular.

1.1 Background

We begin by considering *integer programming* (IP) which refers to the class of constrained optimization problems where, in addition to be subject to conventional linear or nonlinear equality and inequality constraints, the decision variables are constrained to be integers. For illustration, Fig.1 depicts the feasible region of an IP problem

$$\begin{aligned} \text{minimize} \quad & f(x_1, x_2) \\ \text{subject to:} \quad & x_1 \geq 0.5 \\ & x_2 \geq 0.5 \\ & -0.5x_1 + x_2 \leq 4.25 \\ & 4x_1 - x_2 \leq 25.5 \\ & x_1, x_2 \in \mathbb{Z} \end{aligned}$$

where $\mathbb{Z}$ denotes the set of all integers. We see that decision variables x_1 and x_2

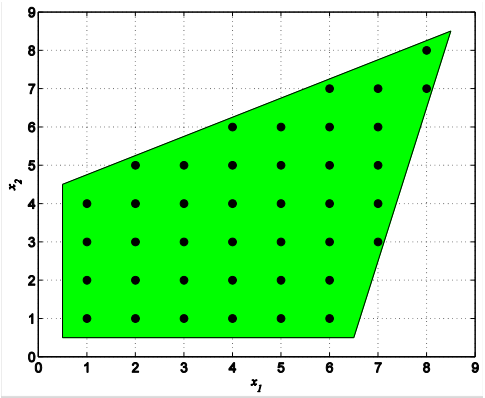

Figure 1. Feasible region of an IP problem

are constrained to be within a polygon (shown in green color) and, at the same time, both x_1 and x_2 must be integers. Therefore, the feasible region is the set of dots in the green area, which is obviously discrete. Because the feasible region is these discrete black dots instead of continuous feasible region, it is nonconvex. Solving IP problems as such are challenging because they are inherently nonconvex problems and the discontinuous nature of the decision variables implies that popular gradient-based algorithms will fail to work. A particular important special case of IP is *binary integer programming* (BIP) where each decision variable is constrained to be 0 or 1(or to be –1 or 1). For the same reason, solving BIP problems is not at all trivial.

Yet another related class of problems is the mixed-integer programing (MIP) in which only a portion of the decision variables is allowed to be continuous while the rest of the variables are constrained to be integers. Again, solving MIP problems is challenging because they are always nonconvex and gradient-based algorithms do not work properly. On the other hand, many MIP problems are encountered in real-life applications arising from the areas of logistics, finance, transportation, resource management, integrated circuit design, and power management [13]. As such, over the years, researchers are highly motivated to develop solution techniques for MIP problems. Our studies in this project will be focused on an important subclass of MIP, namely the mixed-integer quadratic programming (MIQP).

1.1.1 Mixed integer quadratic programming problem

A standard MIQP problem assumes the form

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} + r \\ \text{subject to:} \quad & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \in \chi \end{aligned} \tag{1.1}$$

where $\boldsymbol{P} \in R^{n\times n}$ is symmetric and positive semidefinite, $\boldsymbol{q} \in R^{n\times 1}, r \in R, \boldsymbol{A} \in R^{p\times n}$, and $\boldsymbol{b} \in R^{p\times 1}$ with $p < n$. In (1.1), $\chi = \chi_1 \times \chi_2 \times \cdots \times \chi_n$ is a Cartesian product of n real, closed, nonempty sets, and $\boldsymbol{x} \in \chi$ means that the ith decision variable x_i is constrained to belong to set χ_i for $i = 1, 2, \ldots, n$. As is known to all, if x is constrained to be the continuous decision variables, then the problem in (1.1) is a convex quadratic programming (QP) problem which can readily be solved [1]. In this project, we are interested in the cases where at least one (but possibly more) of the component sets of χ is *nonconvex*. Of practical importance are those cases where several nonconvex component sets of χ are Boolean or integer sets. We also remark that (1.1) covers the class of mixed-integer linear programming (MILP) problems as a special case where matrix $\boldsymbol{P}$ vanishes.

1.1.2 Application of MIQP to economic dispatch

In this section, we briefly introduce the work of [4] where economic dispatch of generators with prohibited operating zones is investigated via an MIQP model. The main goal of the work is to produce a certain amount of electricity at the lowest possible cost subject to constraints on the operating area of the generator due to physical limitations on individual power plant components, where the physical limitations are related to shaft bearing vibration amplification under certain working conditions. These limitations can lead to instability for some loads. To avoid the instability, the concept of forbidden work zones arises. Furthermore, the existence of forbidden zone of a single generator leads to disjunction of solution spaces, the integer variables are introduced to capture these disjoint operating sub-regions. Because the feasible region is consisted by these discrete integer variables, hence the forbidden zone becomes a *nonconvex* feasible region.

The work of [4] establishes an optimization model for the problem described above, where total cost of fuel as the objective function is minimized subject to constraints on power balance, spinning reserve, power output, and prohibited operating zones. The discontinuity in the forbidden zones leads to a mixed-integer quadratic programming problem.

1.2 Solution Methods for Embedded Applications of MIQP

Although MIQP problems are nonconvex, there are many techniques to compute global minimizers for MIQP problems, these include branch-and-bound (Lawler & Wood [15]) and branch-and-cut (Stubbs & Mehrotra, 1999[16]). Branch-and-cut is a combinatorial optimization method for integer programming, in which some or all of the unknowns are limited to integer values. Branch-and-cut involves running a branch and bound algorithm and using cutting planes to tighten the linear programming relaxations. Moreover, the branch and bound algorithm is used to find a value that maximizes or minimizes the value of the real valued function [12]. In general, a problem can be divided into primary and subproblems, which is called column generation. Nowadays, many commercial solvers such as CPLEX, SBB, and MOSEK are developed based on these algorithms. The advantage of these methods is able to

find the global value. Nevertheless, practical implementations of the techniques mentioned above when applying to MIQP problems have indicated that they are inefficient in terms of runtime such as taking up to 16 hours to solve the problem of randomly generated quadratic programming in [10]. It's not that surprising, because MIQP problems are shown to be NP (nondeterministic polynomial)-hard. A problem is NP-hard if an algorithm for solving it can be translated into one for solving any NP-problem. NP-hard therefore means "at least as hard as any NP-problem," although it might, in fact, be harder [14]. Obviously, under the circumstances of *embedded* applications, where an MIQP is solved subject to limited computing resources and constraint on the runtime allowed, the above-mentioned solvers for precise global solutions become less favorable. Instead, one is more interested in methods that can much quickly secure suboptimal solutions with satisfactory performance.

The past several years had witnessed a growing interest in developing heuristics for various nonconvex problems including those tailored to imbedded MIQP problems. In [9] and [10], a technique, known as ADMM heuristic, is applied to solve the MIQP problems, such as economic dispatch [3], hybrid vehicle control, etc., which will be further studied in the Ch.3. Below we present a brief review of ADMM, that is a key algorithmic component in solving embedded MIQP problems [10].

1.2.1 The overview of ADMM

ADMM is an algorithm that solves convex optimization problems by breaking them into smaller blocks, each of which is easier to handle. And it has a strong ability to deal with large-scale convex problems. The idea was first proposed by Gabay, Mercier, Glowinski, and Marrocco in the mid-1970s, although similar ideas have been around since the mid-1950s. The algorithm was studied throughout the 1980s [11], and by the mid-1990s almost all of the theoretical results mentioned here had been established. The fact that ADMM developed well before the availability of large-scale distributed computing systems and a number of optimization problems explains why it is not as well known today as we think [8].

1.2.2 ADMM heuristic for nonconvex constraints

Originally ADMM was developed for convex constrained problems and around 2010 was extended to nonconvex settings as an effective heuristic [8]. Although ADMM is not guaranteed to find the global value, it can find suboptimal solution in very short amount of time. For the MIQP problem in (1.1), the only possible nonconvex items are presented in $\boldsymbol{x} \in \chi$, when some sets in χ are nonconvex. The decision variable vector $\boldsymbol{x}$ associated with nonconvex constraint $\boldsymbol{x} \in \chi$ is *renamed* as variable $\boldsymbol{y}$. Each ADMM iteration in this scenario boils down to two sub-problems: the first sub-problem is essentially the same problem as the original one, but it is solved with respect to variable $\boldsymbol{x}$ with $\boldsymbol{y}$ fixed. In this way, the technical difficulties to deal with nonconvex constraints $\boldsymbol{y} \in \chi$ will not occur; the second sub-problem is simply an orthogonal projection problem where the *relaxed* solution obtained from the first sub-problem is projected to Cartesian product χ .Technical details of ADMM iterations are described in Ch. 2.

1.2.3 Improvement in the solution method

This report also proposes that an algorithmic step called polish be added to the ADMM-based algorithm so as to further improve the solution quality in terms of either reduced objective function or improved constraint satisfaction. Details of the technique will be provided in Ch.2 and its effectiveness will be demonstrated in the case studies in Ch.3.

1.3 Organization of the Report

The rest of the report is organized as follows. After the introduction of necessary background of embedded MIQP problems and basic idea of ADMM iterations in Chapter 1, Chapter 2 provides the technical details concerning ADMM algorithms, their nonconvex extension, and application to the MIQP problem in (1.1). Also included are discussions on issues related to convergence and initialization of the algorithm; performance enhancement via preconditioning; and a proposal of “polish” technique for further improvement of the solution. Chapter 3 presents three examples of applications of MIQP problems to demonstrate the validity and effectiveness of the

algorithms from Chapter 2. Several concluding remarks and suggestions for future work are made in Chapter 4.

1.4 Contributions

The main contributions of my project are listed as follows:

- The advantages of ADMM for embedded application are revealed based on the large number of experimental data.
- Strategy of finding α to achieve the smallest objective value is performed.
- The technique named polish is applied to improve the quality of solution. Formulations are developed to test the effect of polish on both equality constraint satisfaction and inequality constraint satisfaction. And through a large number of experimental data, the effect of polish on the quality of the answer is proved.
- Setting up the model for economic dispatch problems. Building up matrices ***A, b, P,*** and ***q*** for the case of 4 generators based on the several constraints. Inequality constraints are converted to equality constraints while setting up the model.

Chapter 2

ADMM-Based Heuristics for MIQP Problems

The main objective of this chapter is to present algorithms for MIQP problems that are based on alternating direction method of multipliers (ADMM). To this end, the chapter first provides basics of ADMM for convex problems, which is then followed by its extension to nonconvex problems, especially for MIQP. Finally, a simple yet effective follow-up technique, called *polish*, is applied for performance enhancement of the ADMM-based heuristic. We begin by introducing the notion of duality which is a key ingredient in the development of ADMM.

2.1 Duality and Ascent Dual Algorithm

2.1.1 Dual function and dual problem

The concept of duality as applied to optimization is essentially a problem transformation that leads to an indirect but sometimes more efficient solution method. In a duality-based method, the original problem, which is referred to as the *primal* problem, is transformed into a problem whose decision variables are the Lagrange multipliers of the primal. The transformed problem is called the *dual* problem.

To describe how a dual problem is constructed, we need to define a function known as *Lagrange dual* function. Consider the general convex programming (CP) problem

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) \\ &\text{subject to:} && \boldsymbol{a}_i^T \boldsymbol{x} = b_i \quad \text{for } 1 \le i \le p \\ &&& c_j(\boldsymbol{x}) \le 0 \quad \text{for } 1 \le j \le q \end{aligned} \tag{2.1}$$

where $f(\boldsymbol{x})$ and $c_j(\boldsymbol{x})$ for j = 1, 2, …, q are all convex. The Lagrangian of the problem in (2.1) is defined by

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i \left(\boldsymbol{a}_i^T \boldsymbol{x} - b_i \right) + \sum_{j=1}^{q} \mu_i \, c_j(\boldsymbol{x})$$

where $\{\lambda_i, i = 1, 2, ..., p\}$ and $\{\mu_j, j = 1, 2, ..., q\}$ are Lagrange multipliers.

Definition 2.1 The *Lagrange dual function* of problem (2.1) is defined as

$$q(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$$

for $\boldsymbol{\lambda} \in R^p$ and $\boldsymbol{\mu} \in R^q$ with $\boldsymbol{\mu} \geq \mathbf{0}$. Where $\inf_{\boldsymbol{x}}$ is infimum, which means maximum lower bound of $L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$. Note that the Lagrangian $L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ defined above is *convex* with respect to $\boldsymbol{x}$. On the other hand, it can be verified by definition that $L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu})$ is concave with respect to $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$, namely,

Property 2.1 $q(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is a concave function with respect to $\{\boldsymbol{\lambda}, \boldsymbol{\mu}\}$.

Therefore, it makes sense to consider the problem of *maximizing* $q(\boldsymbol{\lambda}, \boldsymbol{\mu})$.

Definition 2.2 The *Lagrange dual problem* with respect to problem (2.1) is defined as

$$\begin{aligned} &\underset{\boldsymbol{\lambda}, \boldsymbol{\mu}}{\text{maximize}} \quad q(\boldsymbol{\lambda}, \boldsymbol{\mu}) \\ &\text{subject to:} \quad \boldsymbol{\mu} \geq \mathbf{0} \end{aligned} \tag{2.2}$$

With the dual problem defined, it is natural to introduce the notion of duality gap.

Property 2.2 For any $\boldsymbol{x}$ feasible for problem (2.1) and $\{\boldsymbol{\lambda}, \boldsymbol{\mu}\}$ feasible for problem (2.2), we have

$$f(\boldsymbol{x}) \geq q(\boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{2.3}$$

This is because

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) + \sum_{i=1}^{p} \lambda_i \left(\boldsymbol{a}_i^T \boldsymbol{x} - b_i \right) + \sum_{j=1}^{q} \mu_i c_j(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{j=1}^{q} \mu_i c_j(\boldsymbol{x}) \leq f(\boldsymbol{x})$$

thus

$$q(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \leq f(\boldsymbol{x})$$

We call the convex minimization problem in (2.1) the *primal problem* and the concave maximization problem in (2.2) the *dual problem*. From (2.3), the *duality gap* between the primal and dual objectives is defined as

$$\delta(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\boldsymbol{x}) - q(\boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{2.4}$$

It follows that for feasible $\{\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}\}$ the duality gap is always *nonnegative*.

Property 2.3 Let $\boldsymbol{x}^*$ be a solution of the primal problem in (2.1). Then the dual function at any feasible $\{\boldsymbol{\lambda}, \boldsymbol{\mu}\}$ serves as a lower bound of the optimal value of the primal objective, $f(\boldsymbol{x}^*)$, namely,

$$f(\boldsymbol{x}^*) \geq q(\boldsymbol{\lambda}, \boldsymbol{\mu}) \tag{2.5}$$

This property follows immediately from (2.3) by taking the minimum of $f(\boldsymbol{x})$ on its left-hand side. Furthermore, by maximizing the dual function $q(\boldsymbol{\lambda}, \boldsymbol{\mu})$ on the right-hand side of (2.5) subject to $\boldsymbol{\mu} \geq \mathbf{0}$, we obtain

$$f(\boldsymbol{x}^*) \geq q(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*) \tag{2.6}$$

where $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ denotes the solution of problem (2.2). Based on (2.6), we introduce the concept of *strong* and *weak* duality as follows.

Definition 2.3 Let $\boldsymbol{x}^*$ and $(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$ be solutions of primal problem (2.1) and dual problem (2.2), respectively. We say strong duality holds if $f(\boldsymbol{x}^*) = q(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$, i.e., the optimal duality gap is zero; and a weak duality holds if $f(\boldsymbol{x}^*) > q(\boldsymbol{\lambda}^*, \boldsymbol{\mu}^*)$.

It can be shown that if the primal problem is strictly feasible, i.e., there exists $\boldsymbol{x}$ satisfying

$$\begin{aligned} \boldsymbol{a}_i^T \boldsymbol{x} &= b_i \quad \text{for } 1 \leq i \leq p \\ c_j(\boldsymbol{x}) &< 0 \quad \text{for } 1 \leq j \leq q \end{aligned}$$

which is to say that the interior of the feasible region of problem (2.1) is *nonempty*, then strong duality holds, i.e., the optimal duality gap is zero.

2.1.2 A dual ascent algorithm

Now consider a linearly constrained convex problem

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) \\ &\text{subject to:} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \end{aligned} \tag{2.7}$$

where $\boldsymbol{x} \in R^n$, $f(\boldsymbol{x})$ is convex, and $\boldsymbol{A} \in R^{p \times n}$ with $p < n$. The Lagrange dual function for problem (2.7) is given by

$$q(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda})$$

where

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$$

with $\boldsymbol{\lambda} \in R^p$. Since the primal problem (2.7) does not involve inequality constraints, the

Lagrange dual problem is an unconstrained one:

$$\underset{\boldsymbol{\lambda}}{\text{maximize}} \;\; q(\boldsymbol{\lambda}) \tag{2.8}$$

and strong duality always holds. Moreover, if $\boldsymbol{\lambda}^*$ is a maximizer of the dual problem (2.8), the solution of primal problem (2.7) can be obtained by minimizing $L(\boldsymbol{x}, \boldsymbol{\lambda}^*)$, namely,

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}}{\text{argmin}}\, L(\boldsymbol{x}, \boldsymbol{\lambda}^*) \tag{2.9}$$

where argmin stands for argument of the minimum. In mathematics, the arguments of the minimum are the points, or elements, of the domain of some function at which the function values are minimized.

The above analysis suggests an iterative scheme for solving the problems (2.7) and (2.8):

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}_k) \tag{2.10a}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha_k (\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{b}) \tag{2.10b}$$

where $\alpha_k > 0$ is a step size, and $\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{b}$ is residual of the equality constraints in the *k*th iteration. It can be shown that the gradient of the dual function $q(\boldsymbol{\lambda})$ in the *k*th is equal to $\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{b}$ [8], and hence the step in (2.10b) updates $\boldsymbol{\lambda}_k$ along the *ascent* direction $\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{b}$ for the dual (maximization) problem, thus the name of the algorithm.

The convergence of the dual ascent algorithm can be considerably improved by working with an *augmented* Lagrangian

$$L_\tau(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + \boldsymbol{\lambda}^T (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) + \tfrac{\tau}{2} \| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} \|_2^2 \tag{2.11}$$

For some $\tau > 0$. That leads to modified iteration steps

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} L_\tau(\boldsymbol{x}, \boldsymbol{\lambda}_k) \tag{2.12a}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \tau (\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{b}) \tag{2.12b}$$

where the step size in (2.10b) is now replaced by parameter τ which is an iteration-

independent constant [8].

2.2 Alternating Direction Method of Multipliers

2.2.1 Problem formulation and basic ADMM

As a significant extension of the dual ascent algorithm, the alternating direction method of multipliers (ADMM) [8] is aimed at solving the class of convex problems

$$\text{minimize} \quad f(\boldsymbol{x}) + h(\boldsymbol{y}) \tag{2.13a}$$

$$\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} = \boldsymbol{c} \tag{2.13b}$$

where $\boldsymbol{x} \in R^n$ and $\boldsymbol{y} \in R^m$ are variables, $\boldsymbol{A} \in R^{p\times n}$, $\boldsymbol{B} \in R^{p\times m}$, $\boldsymbol{c} \in R^{p\times 1}$, and $f(\boldsymbol{x})$ and $h(\boldsymbol{y})$ are convex functions. Note that in (2.13) the variable in both objective function and constraint is split into two parts, namely $\boldsymbol{x}$ and $\boldsymbol{y}$, each covers only a set of variables. By definition, the Lagrangian for the problem in (2.13) is given by

$$L(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) + h(\boldsymbol{y}) + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c})$$

Recall the Karush–Kuhn–Tucker (KKT) condition, if $\boldsymbol{x}^*$ is a local minimizer of the problem (2.1) and is regular for the constraints that are active at $\boldsymbol{x}^*$, then

$$a_i(x^*) = 0 \quad \text{for } i = 1, 2, ..., p$$

$$c_j(x^*) \le 0 \quad \text{for } j = 1, 2, ..., q$$

There exist Lagrange multipliers λ_i^* for $1 \le i \le p$ and μ_j^* for $1 \le j \le q$ such that

$$\nabla f(x^*) + \sum_{i=1}^{P} \lambda_i^* \nabla a_i(x^*) + \sum_{j=1}^{q} \mu_j^* \nabla c_j(x^*) = 0$$

Complementarity condition

$$\lambda_i^* a_i(x^*) = 0 \quad \text{for } 1 \le i \le p$$

$$\mu_j^* c_j(x^*) = 0 \quad \text{for } 1 \le j \le q$$

$$\mu_j^* \ge 0 \quad \text{for } 1 \le j \le q$$

If both $f(\boldsymbol{x})$ and $h(\boldsymbol{y})$ are differentiable functions for this case, the KKT conditions for problem (2.13) are given by

$$\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}=\boldsymbol{c} \tag{2.14a}$$

$$\nabla f(\boldsymbol{x})+\boldsymbol{A}^T\boldsymbol{\lambda}=\boldsymbol{0} \tag{2.14b}$$

$$\nabla h(\boldsymbol{y})+\boldsymbol{B}^T\boldsymbol{\lambda}=\boldsymbol{0} \tag{2.14c}$$

The Lagrange dual of (2.13) assumes the form

$$\text{maximize} \quad q(\boldsymbol{\lambda}) \tag{2.15}$$

where

$$q(\boldsymbol{\lambda})=\inf_{x,y}\left[\, f(\boldsymbol{x})+h(\boldsymbol{y})+\boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}-\boldsymbol{c})\right]$$

which can be expressed as

$$\begin{aligned} q(\boldsymbol{\lambda}) &= \inf_{x}\left[\, f(\boldsymbol{x})+\boldsymbol{\lambda}^T\boldsymbol{A}\boldsymbol{x}\right]+\inf_{y}\left[h(\boldsymbol{y})+\boldsymbol{\lambda}^T\boldsymbol{B}\boldsymbol{y}\right]-\boldsymbol{\lambda}^T\boldsymbol{c} \\ &= -\sup_{x}\left[(-\boldsymbol{A}^T\boldsymbol{\lambda})^T\boldsymbol{x}-f(\boldsymbol{x})\right]-\sup_{y}\left[(-\boldsymbol{B}^T\boldsymbol{\lambda})^T\boldsymbol{y}-h(\boldsymbol{y})\right]-\boldsymbol{\lambda}^T\boldsymbol{c} \end{aligned}$$

where "sup" stands for supremum which by definition is the smallest upper bound of the set of numbers generated in $[\cdot]$. It can be shown that

$$\nabla q(\boldsymbol{\lambda})=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}-\boldsymbol{c} \tag{2.16}$$

where $\{\boldsymbol{x},\boldsymbol{y}\}$ minimizes $L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda})$ for a given $\boldsymbol{\lambda}$ [8].

If in addition we assume that $f(\boldsymbol{x})$ and $h(\boldsymbol{y})$ are strictly convex, a solution of problem (2.13) can be found by minimizing the Lagranging $L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda}^*)$ with respect to primal variables $\boldsymbol{x}$ and $\boldsymbol{y}$, where $\boldsymbol{\lambda}^*$ maximizes the dual function $q(\boldsymbol{\lambda})$. This in conjunction with (2.16) suggests *dual ascent* iterations for problem (2.13) as follows:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{x} L(\boldsymbol{x},\boldsymbol{y}_k,\boldsymbol{\lambda}_k)=\arg\min_{x}\left[\, f(\boldsymbol{x})+\boldsymbol{\lambda}_k^T\boldsymbol{A}\boldsymbol{x}\right] \\ \boldsymbol{y}_{k+1} &= \arg\min_{y} L(\boldsymbol{x}_k,\boldsymbol{y},\boldsymbol{\lambda}_k)=\arg\min_{y}\left[h(\boldsymbol{y})+\boldsymbol{\lambda}_k^T\boldsymbol{B}\boldsymbol{y}\right] \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k+\alpha_k(\boldsymbol{A}\boldsymbol{x}_{k+1}+\boldsymbol{B}\boldsymbol{y}_{k+1}-\boldsymbol{c}) \end{aligned} \tag{2.17}$$

The scalar $\alpha_k>0$ in (2.17) is chosen to maximize $q(\boldsymbol{\lambda})$ (see (2.16)) along the direction $\boldsymbol{A}\boldsymbol{x}_{k+1}+\boldsymbol{B}\boldsymbol{y}_{k+1}-\boldsymbol{c}$.

Convex problems of form (2.13) with less restrictive $f(\boldsymbol{x})$ and $h(\boldsymbol{y})$ as well as data

matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ can be handled by examining *augmented dual* based on the *augmented Lagrangian* which is defined by [8]

$$L_\alpha(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda}) = f(\boldsymbol{x}) + h(\boldsymbol{y}) + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}-\boldsymbol{c}) + \tfrac{\alpha}{2}\| \boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}-\boldsymbol{c}\|_2^2 \qquad (2.18)$$

Note that $L_\alpha(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda})$ in (2.18) includes the conventional Lagrangian $L(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda})$ as a special case when parameter α is set to zero. The introduction of augmented Lagrangian may be understood by considering the following [8]: if we modify the objective function in (2.13) by adding a penalty term $\frac{\alpha}{2}\| \boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}-\boldsymbol{c}\|_2^2$ to take care of violation of the equality constraint, namely,

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) + h(\boldsymbol{y}) + \tfrac{\alpha}{2}\| \boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}-\boldsymbol{c}\|_2^2 \\ &\text{subject to:} \quad \boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}=\boldsymbol{c} \end{aligned} \qquad (2.19)$$

then the conventional Lagrangian of problem (2.19) is exactly equal to $L_\alpha(\boldsymbol{x},\boldsymbol{y},\boldsymbol{\lambda})$ in (2.18). By definition the dual problem of (2.19) is given by

$$\text{maximize} \quad q_\alpha(\boldsymbol{\lambda})$$

where

$$q_\alpha(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x},\boldsymbol{y}} \left[f(\boldsymbol{x}) + h(\boldsymbol{y}) + \boldsymbol{\lambda}^T(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}-\boldsymbol{c}) + \tfrac{\alpha}{2}\| \boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}-\boldsymbol{c}\|_2^2 \right]$$

Unlike the dual ascent iterations in (2.17) where the minimization of the Lagrangian with respect to variables $\{\boldsymbol{x}, \boldsymbol{y}\}$ is split into two separate steps with reduced problem size, the augmented Lagrangian are no longer separable in variables $\boldsymbol{x}$ and $\boldsymbol{y}$ because of the presence of the penalty term. In ADMM iterations, this issue is addressed by *alternating* updates of the primal variables $\boldsymbol{x}$ and $\boldsymbol{y}$, namely,

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \boldsymbol{\lambda}_k^T \boldsymbol{A}\boldsymbol{x} + \tfrac{\alpha}{2}\| \boldsymbol{A}\boldsymbol{x}+\boldsymbol{B}\boldsymbol{y}_k-\boldsymbol{c}\|_2^2 \right] \\ \boldsymbol{y}_{k+1} &= \arg\min_{\boldsymbol{y}} \left[h(\boldsymbol{y}) + \boldsymbol{\lambda}_k^T \boldsymbol{B}\boldsymbol{y} + \tfrac{\alpha}{2}\| \boldsymbol{A}\boldsymbol{x}_{k+1}+\boldsymbol{B}\boldsymbol{y}-\boldsymbol{c}\|_2^2 \right] \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \alpha(\boldsymbol{A}\boldsymbol{x}_{k+1}+\boldsymbol{B}\boldsymbol{y}_{k+1}-\boldsymbol{c}) \end{aligned} \qquad (2.20)$$

A point to note is that parameter α from the quadratic penalty term is now used in (2.20) to update Lagrange multiplier $\boldsymbol{\lambda}_k$, thereby eliminating a line search step to compute α_k as required in (2.17). To justify (2.20), note that $\boldsymbol{y}_{k+1}$ minimizes

$h(\boldsymbol{y}) + \boldsymbol{\lambda}_k^T \boldsymbol{B}\boldsymbol{y} + \frac{\alpha}{2} \| \boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} \|_2^2$, hence

$$\boldsymbol{0} = \nabla h(\boldsymbol{y}_{k+1}) + \boldsymbol{B}^T \boldsymbol{\lambda}_k + \alpha \boldsymbol{B}^T (\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{y}_{k+1} - \boldsymbol{c}) = \nabla h(\boldsymbol{y}_{k+1}) + \boldsymbol{B}^T \left[\boldsymbol{\lambda}_k + \alpha (\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{y}_{k+1} - \boldsymbol{c}) \right]$$

which in conjunction with the 3rd equation in (2.20) leads to

$$\nabla h(\boldsymbol{y}_{k+1}) + \boldsymbol{B}^T \boldsymbol{\lambda}_{k+1} = 0$$

Therefore, the KKT condition in (2.14c) is satisfied by ADMM iterations. In addition, since $\boldsymbol{x}_{k+1}$ minimizes $f(\boldsymbol{x}) + \boldsymbol{\lambda}_k^T \boldsymbol{A}\boldsymbol{x} + \frac{\alpha}{2} \| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y}_k - \boldsymbol{c} \|_2^2$, we have

$$\begin{aligned} \boldsymbol{0} &= \nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^T \boldsymbol{\lambda}_k + \alpha \boldsymbol{A}^T (\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{y}_k - \boldsymbol{c}) \\ &= \nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^T \left[\boldsymbol{\lambda}_k + \alpha (\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{y}_k - \boldsymbol{c}) \right] \\ &= \nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^T \boldsymbol{\lambda}_{k+1} - \alpha \boldsymbol{A}^T \boldsymbol{B}(\boldsymbol{y}_{k+1} - \boldsymbol{y}_k) \end{aligned}$$

i.e.,

$$\nabla f(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^T \boldsymbol{\lambda}_{k+1} = \alpha \boldsymbol{A}^T \boldsymbol{B}(\boldsymbol{y}_{k+1} - \boldsymbol{y}_k) \tag{2.21}$$

On comparing (2.21) with (2.14b), a *dual residual* in the *k*th iteration can be defined as

$$\boldsymbol{d}_k = \alpha \boldsymbol{A}^T \boldsymbol{B}(\boldsymbol{y}_{k+1} - \boldsymbol{y}_k) \tag{2.22}$$

From (2.14a), a *primal residual* in the *k*th iteration is defined as

$$\boldsymbol{r}_k = \boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{y}_{k+1} - \boldsymbol{c} \tag{2.23}$$

Together, $\{\boldsymbol{r}_k, \boldsymbol{d}_k\}$ measures closeness of the *k*th ADMM iteration $\{\boldsymbol{x}_k, \boldsymbol{y}_k, \boldsymbol{\lambda}_k\}$ to the solution of problem (2.13), thus a reasonable criteria for terminating ADMM iterations is when

$$\| \boldsymbol{r}_k \|_2 < \varepsilon_p \quad \text{and} \quad \| \boldsymbol{d}_k \|_2 < \varepsilon_d \tag{2.24}$$

where ε_p and ε_d are prescribed tolerances for primal and dual residuals, respectively.

Convergence of the ADMM iterations in (2.20) has been investigated under various assumptions, see [8] and [17] and the references cited therein. If both $f(\boldsymbol{x})$ and $h(\boldsymbol{y})$ are strongly convex with parameters m_f and m_h, respectively, and parameter α is chosen to satisfy

$$\alpha^3 \le \frac{m_f m_h^2}{\rho(\boldsymbol{A}^T \boldsymbol{A}) \rho(\boldsymbol{B}^T \boldsymbol{B})^2}$$

where $\rho(\boldsymbol{M})$ denotes the largest eigenvalue of symmetric matrix $\boldsymbol{M}$, then both primal and dual residuals vanish at rate $O(1/k)$ [GOSB14], namely,

$$\| \boldsymbol{r}_k \|_2 \le O(1/k) \quad \text{and} \quad \| \boldsymbol{d}_k \|_2 \le O(1/k)$$

We now summarize the method for solving the problem in (2.13) as an algorithm below.

ADMM for problem (2.13)

Step 1 Input parameter $\alpha > 0$, $\boldsymbol{y}_0$, $\boldsymbol{\lambda}_0$, and tolerance $\varepsilon_p > 0$, $\varepsilon_d > 0$.

Set $k = 0$.

Step 2 Compute $\{\{\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1}, \boldsymbol{\lambda}_{k+1}\}$ using (2.20).

Step 3 Compute $\boldsymbol{d}_k$ and $\boldsymbol{r}_k$ using (2.22) and (2.23), respectively.

Step 4 If the conditions in (2. 24) are satisfied, output $(\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1})$ as solution and stop; Otherwise, set $k = k + 1$ and repeat from Step 2.

2.2.2 Scaled ADMM

Several variants of ADMM are available, one of them is that of the *scaled form* ADMM. The scaled form ADMM and the unscaled form ADMM are obviously equivalent, but the formula for the scaled ADMM is often shorter than the formula for the unscaled ADMM, so we will use the scaled ADMM in the following. We use the unscaled form when we want to emphasize the role of the dual variable or give explanations that depend on (unscaled) dual variable [8]. Firstly, by letting

$$\boldsymbol{r} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} \quad \text{and} \quad \boldsymbol{\nu} = \boldsymbol{\lambda}/\alpha,$$

we write the augmented Lagrangian as

$$\begin{aligned} L_\alpha(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\lambda}) &= f(\boldsymbol{x}) + h(\boldsymbol{y}) + \boldsymbol{\lambda}^T \boldsymbol{r} + \tfrac{\alpha}{2} \| \boldsymbol{r} \|_2^2 \\ &= f(\boldsymbol{x}) + h(\boldsymbol{y}) + \tfrac{\alpha}{2} \| \boldsymbol{r} + \boldsymbol{\nu} \|_2^2 - \tfrac{\alpha}{2} \| \boldsymbol{\nu} \|_2^2 \\ &= f(\boldsymbol{x}) + h(\boldsymbol{y}) + \tfrac{\alpha}{2} \| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} + \boldsymbol{\nu} \|_2^2 - \tfrac{\alpha}{2} \| \boldsymbol{\nu} \|_2^2 \end{aligned}$$

Consequently, the scaled ADMM algorithm can be outlined as follows.

Scaled ADMM for problem (2.13)

Step 1 Input parameter $\alpha > 0$, $\boldsymbol{y}_0$, $\boldsymbol{\nu}_0$, and tolerance $\varepsilon_p > 0$, $\varepsilon_d > 0$.

Set $k = 0$.

Step 2 Compute

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \tfrac{\alpha}{2} \| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{y}_k - \boldsymbol{c} + \boldsymbol{\nu}_k \|_2^2 \right] \\
\boldsymbol{y}_{k+1} &= \arg\min_{\boldsymbol{y}} \left[h(\boldsymbol{y}) + \tfrac{\alpha}{2} \| \boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{y} - \boldsymbol{c} + \boldsymbol{\nu}_k \|_2^2 \right] \\
\boldsymbol{\nu}_{k+1} &= \boldsymbol{\nu}_k + \boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{y}_{k+1} - \boldsymbol{c}
\end{aligned} \tag{2.25}$$

Step 3 Compute $\boldsymbol{d}_k$ and $\boldsymbol{r}_k$ using (2.22) and (2.23), respectively.

Step 4 If the conditions in (2. 24) are satisfied, output $(\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1})$ as solution and stop; Otherwise, set $k = k + 1$ and repeat from Step 2.

2.2.3 ADMM for general convex problems

Consider the general constrained convex problem

$$\begin{aligned}
&\text{minimize} \quad f(\boldsymbol{x}) \\
&\text{subject to:} \quad \boldsymbol{x} \in C
\end{aligned} \tag{2.26}$$

where $f(\boldsymbol{x})$ is a convex function and C is a convex set representing the feasible region of the problem. Evidently, the problem in (2.26) can be formulated as

$$\text{minimize} \quad f(\boldsymbol{x}) + I_C(\boldsymbol{x}) \tag{2.27}$$

where $I_C(\boldsymbol{x})$ is the indicator function associated with set C that is defined by

$$I_C(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in C \\ +\infty & \text{otherwise} \end{cases}$$

The problem in (2.27) can in turn be written as

$$\begin{aligned}
&\text{minimize} \quad f(\boldsymbol{x}) + I_C(\boldsymbol{y}) \\
&\text{subject to:} \quad \boldsymbol{x} - \boldsymbol{y} = \boldsymbol{0}
\end{aligned} \tag{2.28}$$

that fits nicely into the ADMM formulation in (2.13) [8]. The scaled ADMM iterations for (2.28) are given by

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \left[f(\boldsymbol{x}) + \tfrac{\alpha}{2} \| \boldsymbol{x} - \boldsymbol{y}_k + \boldsymbol{\nu}_k \|_2^2 \right] \\
\boldsymbol{y}_{k+1} &= \arg\min_{\boldsymbol{y}} \left[I_C(\boldsymbol{y}) + \tfrac{\alpha}{2} \| \boldsymbol{y} - (\boldsymbol{x}_{k+1} + \boldsymbol{\nu}_k) \|_2^2 \right] \\
\boldsymbol{\nu}_{k+1} &= \boldsymbol{\nu}_k + \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}
\end{aligned}$$

where the $\boldsymbol{y}$-minimization is obtained by minimizing $\|\boldsymbol{y}-(\boldsymbol{x}_{k+1}+\boldsymbol{\nu}_k)\|_2$ subject to $\boldsymbol{y}\in C$. This means that $\boldsymbol{y}_{k+1}$ can be obtained by projecting $\boldsymbol{x}_{k+1}+\boldsymbol{\nu}_k$ onto set C, and hence the ADMM iterations become

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}}\left[f(\boldsymbol{x})+\tfrac{\alpha}{2}\|\boldsymbol{x}-\boldsymbol{y}_k+\boldsymbol{\nu}_k\|_2^2\right]\\ \boldsymbol{y}_{k+1} &= P_C(\boldsymbol{x}_{k+1}+\boldsymbol{\nu}_k)\\ \boldsymbol{\nu}_{k+1} &= \boldsymbol{\nu}_k+\boldsymbol{x}_{k+1}-\boldsymbol{y}_{k+1}\end{aligned} \tag{2.29}$$

where $P_C(\boldsymbol{z})$ denotes the projection of point $\boldsymbol{z}$ onto convex set C. We remark that the projection can be accomplished by solving the convex problem

$$\begin{aligned}&\text{minimize} \quad \|\boldsymbol{y}-\boldsymbol{z}\|_2\\ &\text{subject to:} \quad \boldsymbol{y}\in C\end{aligned}$$

2.3 ADMM for Nonconvex Problems

In this section, ADMM is extended to some nonconvex problems as a *heuristic*. We consider the class of constrained problems [8, Sec. 9.1] which assumes the form

$$\begin{aligned}&\text{minimize} \quad f(\boldsymbol{x})\\ &\text{subject to:} \quad \boldsymbol{x}\in C\end{aligned} \tag{2.30}$$

where function $f(\boldsymbol{x})$ is convex but the feasible region C is a nonconvex, hence (2.30) formulates a class of nonconvex problems. On comparing the formulation in (2.30) with that in (2.26), the two problem formulations look quite similar except the convexity of the feasible region involved: the set C in (2.26) is convex while the set C in (2.30) is not. It is therefore intuitively reasonable that an ADMM heuristic approach be developed by extending the techniques used for the problem in (2.26) to the problem in (2.30). First, the problem in (2.30) is re-formulated as

$$\text{minimize} \quad f(\boldsymbol{x})+I_C(\boldsymbol{x}) \tag{2.31}$$

After that, in order to make the objective function separable, a new variable $\boldsymbol{y}$ is introduced. Then the problem is shifted back to

$$\begin{aligned}&\text{minimize} \quad f(\boldsymbol{x})+I_C(\boldsymbol{y})\\ &\text{subject to} \quad \boldsymbol{x}-\boldsymbol{y}=0.\end{aligned} \tag{2.32}$$

The ADMM iterations for nonconvex problems takes a similar form to that for convex problems:

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}}\left[f(\boldsymbol{x}) + \tfrac{\alpha}{2} \| \boldsymbol{x} - \boldsymbol{y}_k + \boldsymbol{\nu}_k \|_2^2 \right] \\
\boldsymbol{y}_{k+1} &= \arg\min_{\boldsymbol{y}}\left[I_C(\boldsymbol{y}) + \tfrac{\alpha}{2} \| \boldsymbol{y} - (\boldsymbol{x}_{k+1} + \boldsymbol{\nu}_k) \|_2^2 \right] \\
\boldsymbol{\nu}_{k+1} &= \boldsymbol{\nu}_k + \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}
\end{aligned}$$

where the $\boldsymbol{x}$-minimization is obviously a convex problem because $f(\boldsymbol{x})$ is convex while the y-minimization can be obtained by minimizing $\| \boldsymbol{y} - (\boldsymbol{x}_{k+1} + \boldsymbol{\nu}_k) \|_2$ subject to $\boldsymbol{y} \in C$. This means that $\boldsymbol{y}_{k+1}$ can be computed by projecting $\boldsymbol{x}_{k+1} + \boldsymbol{\nu}_k$ onto set C, and hence the ADMM iterations can be expressed as

$$\begin{aligned}
\boldsymbol{x}_{k+1} &= \operatorname{argmin}_{x}[f(\boldsymbol{x}) + \tfrac{\alpha}{2} \| \boldsymbol{x} - \boldsymbol{y}_k + \boldsymbol{\nu}_k \|_2^2] \\
\boldsymbol{y}_{k+1} &= P_C(\boldsymbol{x}_{k+1} + \boldsymbol{\nu}_k) \\
\boldsymbol{\nu}_{k+1} &= \boldsymbol{\nu}_k + \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}
\end{aligned} \tag{2.33}$$

where $P_C(\boldsymbol{x}_{k+1} + \boldsymbol{\nu}_k)$ denote the projection of $\boldsymbol{x}_{k+1} + \boldsymbol{\nu}_k$ onto nonconvex set C. It is the projection in the second equation in (2.33) that differs from that of (2.29) and is difficult to calculate in general as it involves a nonconvex feasible region C. As demonstrated in [8, Sec. 9.1], however, there are several important cases where the projection involved in (2.33) can be carried out precisely. Based on the analysis, an ADMM-based algorithm for the nonconvex problem in (2.30) can be outlined as follows.

Scaled ADMM for problem (2.30)

Step 1 Input parameters $\alpha > 0$, $\boldsymbol{y}_o, \boldsymbol{\nu}_0$, and tolerances $\varepsilon_p > 0$, $\varepsilon_d > 0$. Set number of iterations $k = 0$.

Step 2 Compute $\{\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1}, \boldsymbol{\nu}_{k+1}\}$ using (2.33).

Step 3 Compute dual residual

$$\boldsymbol{d}_k = -\alpha(\boldsymbol{y}_{k+1} - \boldsymbol{y}_k)$$

and primal residual

$$\boldsymbol{r}_k = \boldsymbol{x}_{k+1} - \boldsymbol{y}_{k+1}$$

Step 4 If

$$\|\boldsymbol{r}_k\|_2 < \varepsilon_p \text{ and } \|\boldsymbol{d}_k\|_2 < \varepsilon_d$$

output $\{\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1}\}$ as solution and stop; Otherwise, set $k = k+1$ and repeat from Step 2.

Example 2.1 In order to better understand the above algorithm, ADMM was applied to the following nonconvex problem

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) = x_2^2 - 2x_1 + x_2 \\ &\text{subject to:} && x_1^2 + x_2^2 - 16 = 0 \end{aligned}$$

where the feasible region

$$C = \{\boldsymbol{x} : x_1^2 + x_2^2 = 16\}$$

is a circle of radius 4 with a center at the origin, which is obviously nonconvex. The problem at hand seeks to find a point on that circle, which minimizes the objective function. The problem fits into the formulation in (2.30), and hence the scaled ADMM heuristic in (2.33) applies. The objective function in the ***x***-minimization (i.e. the first step in (2.33)) assumes the form

$$x_2^2 - 2x_1 + x_2 + \tfrac{\alpha}{2}\left\|\boldsymbol{x} - \boldsymbol{y}^k + \boldsymbol{v}^k\right\|_2^2 = \frac{1}{2}\boldsymbol{x}^T\begin{bmatrix}\alpha & 0\\ 0 & 2+\alpha\end{bmatrix}\boldsymbol{x} - \boldsymbol{x}^T\left(\alpha(\boldsymbol{y}^k - \boldsymbol{v}^k) + \begin{bmatrix}2\\ -1\end{bmatrix}\right)$$

up to a constant term. To compute the minimum point $\boldsymbol{x}_{k+1}$ in the *k+1*th iteration, we compute the gradient of the object function and then set it to zero, namely,

$$\nabla\left(\frac{1}{2}\boldsymbol{x}^T\begin{bmatrix}\alpha & 0\\ 0 & 2+\alpha\end{bmatrix}\boldsymbol{x} - \boldsymbol{x}^T\left(\alpha(\boldsymbol{y}_k - \boldsymbol{v}_k) + \begin{bmatrix}2\\ -1\end{bmatrix}\right)\right) = \boldsymbol{0}$$

which leads to

$$\boldsymbol{x}_{k+1} = \begin{bmatrix}\frac{1}{\alpha} & 0\\ 0 & \frac{1}{2+\alpha}\end{bmatrix}\left(\alpha(\boldsymbol{y}_k - \boldsymbol{v}_k) + \begin{bmatrix}2\\ -1\end{bmatrix}\right) \tag{2.34}$$

Next, $\boldsymbol{x}^{k+1} + \boldsymbol{v}^k$ is projected onto circle *C*. To proceed, suppose the two coordinates of $\boldsymbol{x}_{k+1} + \boldsymbol{v}_k$ be p_1 and p_2, and the two coordinates of the projection $P_C(\boldsymbol{x}_{k+1} + \boldsymbol{v}_k)$ be q_1

and q_2, then it can readily be verified that (i) if $p_1 = 0$ and $p_2 > 0$, then $q_1 = 0$ and $q_2 = 4$; (ii) if $p_1 = 0$ and $p_2 < 0$, then $q_1 = 0$ and $q_2 = -4$; (iii) if $p_1 > 0$, then $q_1 = t$ and $q_2 = t \cdot p_2/p_1$; and (iv) if $p_1 < 0$, then $q_1 = -t$ and $q_2 = -t \cdot p_2/p_1$, where $t = 4/\sqrt{1+(p_2/p_1)^2}$. Profiles of the primal residual $\| \boldsymbol{r}_k \|_2$ and dual residual $\| \boldsymbol{d}_k \|_2$ during the ADMM iterations are shown in Fig.2. As can be seen from the figure, with $\alpha = 0.8$, $\varepsilon_p = 10^{-4}$, and $\varepsilon_d = 10^{-4}$, It took scaled ADMM 12 iterations to achieve primal and dual residual less than 10^{-4}. It can be seen from the figure that the residual value is still decreasing after the 12th iteration, which also leads to the continuous change of the 5th and 6th decimal places of the solution value. Therefore, the solution is kept three decimal places for accuracy which is as follows,

$$\boldsymbol{x}^* = \begin{bmatrix} 3.980 \\ -0.400 \end{bmatrix}$$

at which the objective function assumes the value $f(\boldsymbol{x}^*) = -8.20$. The equality-constraint satisfaction at the solution was found to be $|(x_1^*)^2 + (x_2^*)^2 - 16|$ $= 3.5527 \times 10^{-15}$.

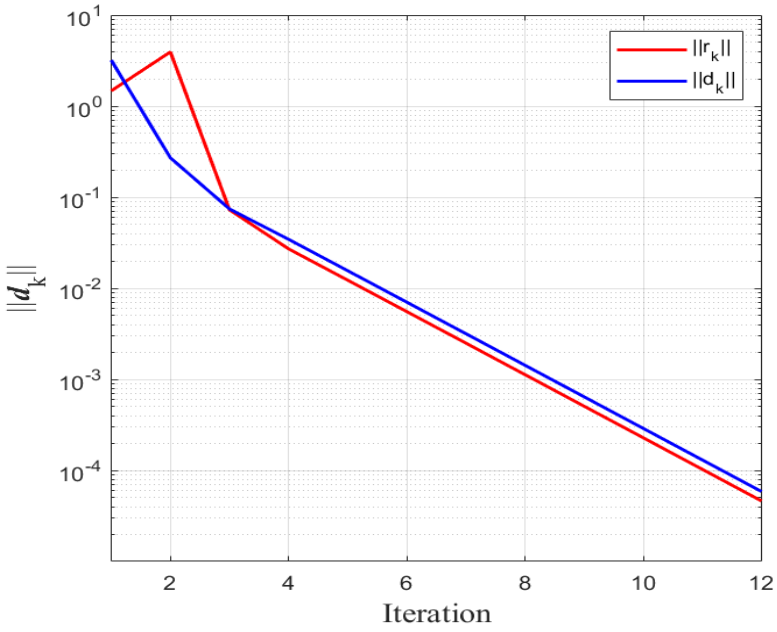

Figure 2. 2-norm of primal residual $\| \boldsymbol{r}_k \|_2$ and dual residual $\| \boldsymbol{d}_k \|_2$

2.4 An ADMM-Based Approach to Solving MIQP Problems

As reviewed in Chapter 1, mixed-integer quadratic programming (MIQP) represents an important class of optimization problems which find real-world applications. In this section, ADMM is applied to solve MIQP problems. We start by presenting a basic ADMM formulation of MIQP problems. This is followed by describing an easy-to-implement preconditioning technique for improving convergence rate of the ADMM-based algorithm. Finally, the novel part of this project called polish is applied for enhancing the performance in terms of either improving constraint satisfaction or reducing the objective or both.

2.4.1 ADMM formulation for MIQP problems

We consider a MIQP problem of the form

$$\text{minimize} \quad \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} + r \tag{2.35a}$$

$$\text{subject to:} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.35b}$$

$$\boldsymbol{x} \in \chi \tag{2.35c}$$

where $\boldsymbol{P} \in R^{n\times n}$ is symmetric and positive semidefinite, $\boldsymbol{q} \in R^{n\times 1}, r \in R, \boldsymbol{A} \in R^{p\times n}$, and $\boldsymbol{b} \in R^{p\times 1}$ with $p < n$. In (2.35c), $\chi = \chi_1 \times \chi_2 \times \cdots \times \chi_n$ is a Cartesian product of n real, closed, nonempty sets, and $\boldsymbol{x} \in \chi$ means that the ith decision variable x_i is constrained to belong to set χ_i for $i = 1, 2, \ldots, n$. As is known to all, if x is constrained to be continuous decision variables, then the problem in (2.35) is a convex quadratic programming (QP) problem which can readily be solved [1]. In this project, we examine the cases where at least one of the component sets of χ is *nonconvex*. Especially important cases are those where several nonconvex component sets of χ are Boolean or integer sets.

To apply ADMM, we reformulate (2.35) by applying the idea described in Sec. (2.3) as

$$\begin{aligned} &\text{minimize} \quad f(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} + r + I_{\chi}(\boldsymbol{y}) \\ &\text{subject to:} \quad \begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{x} - \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{y} = \begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{0}\end{bmatrix} \end{aligned} \tag{2.36}$$

where $I_{\chi}(\boldsymbol{y})$ is the indicator function of set of χ. Recall the indicator function I_c in the sec.2.2.3.

$$I_C(\boldsymbol{x}) = \begin{cases} 0 & \text{if } \boldsymbol{x} \in C \\ +\infty & \text{otherwise} \end{cases}$$

Following (2.33), the ADMM iterations of (2.36) are given by

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \left(\tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} + \tfrac{\alpha}{2}\left\| \begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{x} - \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{y}_k - \begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{0}\end{bmatrix} + \boldsymbol{v}_k \right\|_2^2 \right) \\ \boldsymbol{y}_{k+1} &= P_C(\boldsymbol{x}_{k+1} + \begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}\end{bmatrix}\boldsymbol{v}_k) \\ \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{x}_{k+1} - \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{y}_{k+1} - \begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{0}\end{bmatrix} \end{aligned} \tag{2.37}$$

where P_C is the projection onto set χ.

To solve the $\boldsymbol{x}$-minimization in the first step of (2.37), we compute the gradient of the objective function involved and set it to zero, namely,

$$\nabla\left(\tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} + \tfrac{\alpha}{2}\left\| \begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{x} - \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{y}_k - \begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{0}\end{bmatrix} + \boldsymbol{v}_k \right\|_2^2 \right) = \boldsymbol{0}$$

which leads to

$$\boldsymbol{x}_{k+1} = \left[\boldsymbol{P} + \boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{I}\right]^{-1}\left[-\boldsymbol{q} + \alpha[\boldsymbol{A}^T \ \boldsymbol{I}]\left(\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{y}_k + \begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{0}\end{bmatrix} - \boldsymbol{v}_k \right)\right]$$

and the ADMM iterations are more explicitly expressed as

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \left[\boldsymbol{P} + \boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{I}\right]^{-1}\left[-\boldsymbol{q} + \alpha[\boldsymbol{A}^T \ \boldsymbol{I}]\left(\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{y}_k + \begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{0}\end{bmatrix} - \boldsymbol{v}_k \right)\right] \\ \boldsymbol{y}_{k+1} &= P_C(\boldsymbol{x}^{k+1} + \begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}\end{bmatrix}\boldsymbol{v}_k) \\ \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k + \begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{x}_{k+1} - \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{y}_{k+1} - \begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{0}\end{bmatrix} \end{aligned} \tag{2.38}$$

An important point to note is that the inverse required in $\boldsymbol{x}$-minimization, namely $\left[\boldsymbol{P} + \boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{I}\right]^{-1}$, needs only compute *once* and it applies to all iterations because the matrices involved in the inverse are all constant ones. Needless to say, using the shared

inverse implies fast implementation of the algorithm.

2.4.2 Preconditioned ADMM

For embedded applications, the convergence rate of the algorithm is a primary concern. For applications involving Boolean constraints, the computational complexity of the ADMM iterations is dominated by that of the $\boldsymbol{x}$-minimization step which is essentially a problem of solving a system of linear equations. It is well known [18] that solving such a problem can be done efficiently if the linear system is *well conditioned* meaning that its system matrix has a reasonable condition number (which is defined as the ratio of the largest singular value to the smallest singular value). For ill-conditioned linear systems namely those with very large condition numbers, an effective technique to fix the problem is to pre-multiply the linear system in question by a nonsingular matrix, known as a conditioner, such that the converted linear system becomes less ill-conditioned, and the procedure is known as *preconditioning*.

For problem (2.36), diagonal scaling [19] as one of the many preconditioning techniques works quite well [10]. The specific preconditioned model assumes the form

$$\begin{aligned} &\text{minimize} \quad \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x}+\boldsymbol{q}^T\boldsymbol{x}+r+I_{\chi}(\boldsymbol{y}) \\ &\text{subject to:} \quad \begin{bmatrix}\boldsymbol{E}\boldsymbol{A}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{x}-\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{y}=\begin{bmatrix}\boldsymbol{E}\boldsymbol{b}\\ \boldsymbol{0}\end{bmatrix} \end{aligned} \tag{2.39}$$

where $\boldsymbol{E}$ is a diagonal matrix that normalizes the rows of $\boldsymbol{A}$ in 1-norm or 2-norm. Using the preconditioned formulation in (2.39), the ADMM iterations become

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \begin{bmatrix}\boldsymbol{I} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{P}+\alpha\boldsymbol{I} & \boldsymbol{A}^T\boldsymbol{E}\\ \boldsymbol{E}\boldsymbol{A} & -\frac{1}{\alpha}\boldsymbol{I}\end{bmatrix}^{-1}\begin{bmatrix}-\boldsymbol{q}+\alpha(\boldsymbol{y}_k+\boldsymbol{A}^T\boldsymbol{E}^2\boldsymbol{b}-[\boldsymbol{A}^T\boldsymbol{E} \quad \boldsymbol{I}])\boldsymbol{v}_k\\ \boldsymbol{0}\end{bmatrix} \\ \boldsymbol{y}_{k+1} &= P_C\left(\boldsymbol{x}_{k+1}+\begin{bmatrix}\boldsymbol{0} & \boldsymbol{I}\end{bmatrix}\boldsymbol{v}_k\right) \\ \boldsymbol{v}_{k+1} &= \boldsymbol{v}_k+\begin{bmatrix}\boldsymbol{E}\boldsymbol{A}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{x}_{k+1}-\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{I}\end{bmatrix}\boldsymbol{y}_{k+1}-\begin{bmatrix}\boldsymbol{E}\boldsymbol{b}\\ \boldsymbol{0}\end{bmatrix} \end{aligned} \tag{2.40}$$

where the inverse required in $\boldsymbol{x}$-minimization is evaluated once for *all* iterations.

2.4.3 The algorithm

The ADMM-based algorithm for problem (2.35) is summarized below.

ADMM-based algorithm for problem (2.35)

Step 1 Input parameter $\alpha > 0$, initial $\boldsymbol{y}_0$, $\boldsymbol{v}_0$, and tolerance $\varepsilon > 0$. Set $k = 0$.

Step 2 Compute $\{\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1}, \boldsymbol{v}_{k+1}\}$ using (2.40).

Step 3 Compute residual $\boldsymbol{r}_k = \boldsymbol{x}_{k+1} - \boldsymbol{y}^{k+1}$.

Step 4 If $\| \boldsymbol{r}_k \|_2 < \varepsilon$, output $\{\boldsymbol{x}_{k+1}, \boldsymbol{y}_{k+1}\}$ as solution and stop; Otherwise, set $k := k+1$ and repeat from Step 2.

2.5 Performance Enhancement

In this section, a technique called polish is applied to the ADMM-based algorithm described above as a follow-up step of the algorithm for performance enhancement.

2.5.1 The technique

For the sake of illustration, we consider an MIQP problem of the form

$$\text{minimize} \quad f(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T \boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T \boldsymbol{x} + r \tag{2.41a}$$

$$\text{subject to:} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.41b}$$

$$\boldsymbol{x} \in \chi \tag{2.41c}$$

where $\chi = \chi_1 \times \chi_2 \times \cdots \times \chi_n$ with the first n_1 sets $\{\chi_1, \chi_2, \ldots, \chi_{n_1}\}$ being convex and the remaining n_2 sets $\{\chi_{n_1+1}, \chi_{n_1+2}, \ldots, \chi_n\}$ being {0, 1}-type Boolean sets (here $n_2 = n - n_1$).

Suppose a solution $\boldsymbol{x}^*$ of problem (2.41) has been found using the ADMM-based algorithm (see Sec. 2.4.3). Denote

$$\boldsymbol{x}^* = \begin{bmatrix} \boldsymbol{x}_1^* \\ \boldsymbol{x}_2^* \end{bmatrix} \text{ with } \boldsymbol{x}_1^* \in R^{n_1 \times 1}, \quad \boldsymbol{x}_2^* \in R^{n_2 \times 1}$$

and project each component of $\boldsymbol{x}_2^*$ onto set {0, 1} and denote the resulting vector by $\hat{\boldsymbol{x}}_2^*$. It follows that $\hat{\boldsymbol{x}}_2^* \in \chi_{n_1+1} \times \chi_{n_1+2} \times \cdots \times \chi_n$. We are now in a position to apply a follow-up step called polish by performing the following procedure:

Consider a decision variable $\boldsymbol{x}$ with its last n_2 components fixed to $\hat{\boldsymbol{x}}_2^*$, namely,

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_1 \\ \hat{\boldsymbol{x}}_2^* \end{bmatrix} \tag{2.42}$$

With (2.42), the problem in (2.41) is reduced to a standard *convex* QP problem involving continuous decision vector $\boldsymbol{x}_1$ of dimension n_1, namely,

$$\text{minimize} \quad \tfrac{1}{2}\boldsymbol{x}_1^T \boldsymbol{P}_1 \boldsymbol{x}_1 + \hat{\boldsymbol{q}}^T \boldsymbol{x}_1 + \hat{r} \tag{2.43a}$$

$$\text{subject to:} \quad \boldsymbol{A}_1 \boldsymbol{x}_1 = \boldsymbol{b}_1 \tag{2.43b}$$

$$\boldsymbol{x}_1 \in \chi_1 \times \chi_2 \times \cdots \times \chi_{n_1} \tag{2.43c}$$

where $\hat{\boldsymbol{q}} = \boldsymbol{P}_2 \hat{\boldsymbol{x}}_2^* + \boldsymbol{q}_1$, $\boldsymbol{b}_1 = \boldsymbol{b} - \boldsymbol{A}_2 \hat{\boldsymbol{x}}_2^*$, and $\boldsymbol{P}_1, \boldsymbol{P}_2, \boldsymbol{q}_1, \boldsymbol{A}_1,$ and $\boldsymbol{A}_2$ are taken from

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{P}_1 & \boldsymbol{P}_2 \\ \boldsymbol{P}_2^T & \boldsymbol{P}_3 \end{bmatrix}, \quad \boldsymbol{q} = \begin{bmatrix} \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \end{bmatrix}, \text{ and } \quad \boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_2 \end{bmatrix}$$

Since $\boldsymbol{P}_1$ positive semidefinite and $\chi_1 \times \chi_2 \times \cdots \times \chi_{n_1}$ is convex, (2.43) is a convex QP problem which can be solved efficiently. If we denote the solution of problem (2.43) by $\hat{\boldsymbol{x}}_1^*$ and use it to construct

$$\hat{\boldsymbol{x}}^* = \begin{bmatrix} \hat{\boldsymbol{x}}_1^* \\ \hat{\boldsymbol{x}}_2^* \end{bmatrix} \tag{2.44}$$

then $\hat{\boldsymbol{x}}^*$ is expected to be a solution of problem (2.41) with improved accuracy relative to solution $\boldsymbol{x}^*$ produced from the algorithm in Sec. 2.4.3 in the following sense:

(1) Solution $\hat{\boldsymbol{x}}^*$ satisfies the n_2 Boolean constraints precisely because $\hat{\boldsymbol{x}}_2^*$ is obtained by projecting its components onto set $\{0, 1\}$.

(2) Solution $\hat{\boldsymbol{x}}^*$ satisfies the equality constraints $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ more accurately because its continuous portion, $\hat{\boldsymbol{x}}_1^*$, satisfies $\boldsymbol{A}_1 \boldsymbol{x}_1 = \boldsymbol{b}$ while the Boolean variables are fixed.

Consequently, the objective function value at point $\hat{\boldsymbol{x}}^*$, $f(\hat{\boldsymbol{x}}^*)$, provides a more reliable measure of the achievable optimal performance.

In the next section, the observations made above will be elaborated quantitatively in terms of numerical measures constraint satisfaction.

2.5.2 Numerical measures of constraint satisfaction

When a "solution" for a given constrained optimization problem is obtained by running a certain algorithm, verification of the solution in terms of constraint satisfaction must be performed to ensure that the solution represents a feasible hence acceptable design. For the MIQP problem in (2.41), the verification of constraint satisfaction boils down to that of the p linear equations in (2.41b) and n constraints $x_i \in \chi_i$ in (2.41c). Below we denote a solution of (2.41) by $\boldsymbol{x}^*$.

(1) Satisfaction of $\boldsymbol{Ax} = \boldsymbol{b}$

The satisfaction of the linear equations can be evaluated by several error measures. Based on the equivalence between $\boldsymbol{Ax}^* = \boldsymbol{b}$ and $\boldsymbol{Ax}^* - \boldsymbol{b} = \boldsymbol{0}$, the most straightforward measure is the averaged 2-norm error

$$E_2 = \frac{1}{\sqrt{p}} \| \boldsymbol{Ax}^* - \boldsymbol{b} \|_2 \tag{2.45}$$

Alternatively, satisfaction of the p equations in $\boldsymbol{Ax} = \boldsymbol{b}$ can be evaluated by the averaged 1-norm error

$$E_1 = \frac{1}{p} \| \boldsymbol{Ax}^* - \boldsymbol{b} \|_1 \tag{2.46}$$

Yet another way, one may instead use a *worst-case error* measure

$$E_\infty = \| \boldsymbol{Ax}^* - \boldsymbol{b} \|_\infty \tag{2.47}$$

For reference of the above terms, recall the definition of the p-norm of a vector $\boldsymbol{v} = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}^T$:

$$\| \boldsymbol{v} \|_p = \left[\sum_{i=1}^{n} | v_i |^p \right]^{1/p} \quad \text{for} \quad p \geq 1$$

and

$$\| \boldsymbol{v} \|_\infty = \max_{1 \leq i \leq n} \{ | v_i | \}$$

(2) Satisfaction of $\boldsymbol{x} \in \chi_1 \times \chi_2 \times \cdots \times \chi_n$

There are convex and Boolean sets, and we need to deal with them separately. Suppose the first n_1 sets $\{\chi_1, \chi_2, \ldots, \chi_{n_1}\}$ are convex, while the remaining n_2 sets $\{\chi_{n_1+1}, \chi_{n_1+2}, \ldots, \chi_n\}$ are {0, 1}-type Boolean sets. Denote

$$\boldsymbol{x}^* = \begin{bmatrix} \boldsymbol{x}_1^* \\ \boldsymbol{x}_2^* \end{bmatrix} \text{ with } \boldsymbol{x}_1^* \in R^{n_1} \text{ and } \boldsymbol{x}_2^* \in R^{n_2}$$

where $n_1 + n_2 = n$.

(i) Satisfaction of $\boldsymbol{x}_1^* \in \chi_1 \times \chi_2 \times \cdots \times \chi_{n_1}$

Let

$$\boldsymbol{x}_1^* = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \\ \vdots \\ x_{n_1}^{(1)} \end{bmatrix},$$

where each component is constrained to one-dimensional convex set as

$$x_i^{(1)} \in \chi_i \quad \text{for } i = 1, 2, \ldots, n_1$$

In this project we consider two important instances in this scenario are as follows: χ_i is the entire one-dimensional space, or $x_i^{(1)} \geq 0$. The former case simply means that component $x_i^{(1)}$ is actually *unconstrained* thus needs no error measures, while for the latter case a reasonable error measure appears to be

$$e_i = \max\{-x_i^{(1)}, 0\} \tag{2.48}$$

For illustration suppose, the first r_1 components of $\boldsymbol{x}_1^*$ are unconstrained while the rest of $r_2 = n_1 - r_1$ components of $\boldsymbol{x}_1^*$ are constrained to be nonnegative. Then, following (2.48), satisfaction of constraints $\boldsymbol{x}_1^* \in \chi_1 \times \chi_2 \times \cdots \times \chi_{n_1}$ can be measure by the average error

$$E_c = \frac{1}{r_2} \sum_{i=1}^{r_2} \max\{-x_{r_1+i}^{(1)}, 0\} \tag{2.49}$$

(ii) Satisfaction of $\boldsymbol{x}_2^* \in \chi_{n_1+1} \times \chi_{n_1+2} \times \cdots \times \chi_n$

Let

$$\boldsymbol{x}_2^* = \begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \\ \vdots \\ x_{n_2}^{(2)} \end{bmatrix}$$

Since each χ_{n_1+i} is a Boolean set {0, 1}, we define the projection of component $x_i^{(2)}$ onto {0, 1} as

$$x_{ip}^{(2)} = \begin{cases} 0 \text{ if } x_i^{(2)} < 0.5 \\ 1 \text{ if } x_i^{(2)} > 0.5 \end{cases}$$

and the satisfaction of constraint $x_i^{(2)} \in \chi_{n_1+i}$ can be measured by error $| x_i^{(2)} - x_{ip}^{(2)} |$. It follows that the satisfaction of constraints $\boldsymbol{x}_2^* \in \chi_{n_1+1} \times \chi_{n_1+2} \times \cdots \times \chi_n$ may be measured by the average error

$$E_b = \frac{1}{n_2} \sum_{i=1}^{n_2} | x_i^{(2)} - x_{ip}^{(2)} | \tag{2.50}$$

We now conclude this section with a remark on the evaluation of the value of the objective function $f(\boldsymbol{x})$ at two solution points $\boldsymbol{x}^*$ and $\boldsymbol{x}^{**}$. A point to note is that if one finds $f(\boldsymbol{x}^*) < f(\boldsymbol{x}^{**})$, then the claim that $\boldsymbol{x}^*$ is a better solution relative $\boldsymbol{x}^{**}$ is a valid statement only if both $\boldsymbol{x}^*$ and $\boldsymbol{x}^{**}$ are feasible points with practically the same or comparable constraint satisfaction as quantified in this section. In effect, if $f(\boldsymbol{x}^*)$ assume a smaller value but with poor constraint satisfaction as quantified in this section, then $\boldsymbol{x}^*$ should not be considered as a valuable design for two reasons: First, its feasibility remains a concern. Second, its poor constraint satisfaction allows increased number of candidate solution points in the minimization pool, yielding a "solution" from that pool with a reduced function value.

2.6 An Extension

The MIQP model studied so far (see (2.35)) does not include linear inequality constraints. In this section, we consider an extension of model (2.35) that deals with

both linear equations and linear inequality constraints. As such the MIQP model assumes the form

$$\text{minimize} \quad \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x}+\boldsymbol{q}^T\boldsymbol{x}+r \tag{2.51a}$$

$$\text{subject to:} \quad \boldsymbol{A}\boldsymbol{x}=\boldsymbol{b} \tag{2.51b}$$

$$\boldsymbol{C}\boldsymbol{x}\le\boldsymbol{d} \tag{2.51c}$$

$$\boldsymbol{x}\in\chi \tag{2.51d}$$

where $\boldsymbol{P}$, $\boldsymbol{q}$, $\boldsymbol{A}$, $\boldsymbol{b}$ and χ are defined in the same way as in (2.35), $\boldsymbol{C}\in R^{q\times n}$, and $\boldsymbol{d}\in R^{q\times 1}$. By introducing new decision variable $\boldsymbol{\xi}=\boldsymbol{d}-\boldsymbol{C}\boldsymbol{x}$ [1], the constraints in (2.51c) are equivalent to

$$\boldsymbol{\xi}=\boldsymbol{d}-\boldsymbol{C}\boldsymbol{x},\ \boldsymbol{\xi}\ge\boldsymbol{0}$$

Consequently, model (2.51) is equivalent to

$$\text{minimize} \quad \tfrac{1}{2}\tilde{\boldsymbol{x}}^T\tilde{\boldsymbol{P}}\tilde{\boldsymbol{x}}+\tilde{\boldsymbol{q}}^T\tilde{\boldsymbol{x}}+r \tag{2.52a}$$

$$\text{subject to:} \quad \tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}=\tilde{\boldsymbol{b}} \tag{2.52b}$$

$$\tilde{\boldsymbol{x}}\in\tilde{\chi} \tag{2.52c}$$

where

$$\tilde{\boldsymbol{x}}=\begin{bmatrix}\boldsymbol{x}\\ \boldsymbol{\xi}\end{bmatrix},\ \tilde{\boldsymbol{P}}=\begin{bmatrix}\boldsymbol{P} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix},\ \tilde{\boldsymbol{q}}=\begin{bmatrix}\boldsymbol{q}\\ \boldsymbol{0}\end{bmatrix},\ \tilde{\boldsymbol{A}}=\begin{bmatrix}\boldsymbol{A} & \boldsymbol{0}\\ \boldsymbol{C} & \boldsymbol{I}_q\end{bmatrix},\ \tilde{\boldsymbol{b}}=\begin{bmatrix}\boldsymbol{b}\\ \boldsymbol{d}\end{bmatrix},$$

and

$$\tilde{\chi}=\chi_1\times\chi_2\times\cdots\times\chi_n\times\chi_{n+1}\times\cdots\times\chi_{n+q}$$

with the last q sets $\{\chi_{n+1},\cdots,\chi_{n+q}\}$ being componentwise nonnegative, hence convex. In this way, we have a model in (2.52) for problem (2.51), which is exactly the same as that in (2.35) and hence the algorithm in Sec. 2.4.3 as well as the polish step apply.

Chapter 3

Results and discussions

In this chapter, we present three examples to demonstrate the usefulness of the ADMM-based technique studied in this project. The first two examples are originally from reference [10], and we use them to verify the technique and evaluate the performance before and after polish. The third example is originally from reference [4] which finds global solution of the MIQP problem by a commercial solver with branch-and-bound algorithm [24]. Here the problem in [4] is solved by the ADMM-based technique for the purpose of performance evaluation and comparison.

CVX, a package for specifying and solving convex programs [25], [26], was used for convenient MATLAB coding. All numerical computations were carried out on a PC with four 2.40 GHz cores and 8 GB RAM within an MATLAB environment, version 2018b.

3.1 Randomly Generated Quadratic Programming Problems

This example was originated from reference [10] where one deals with a set of mixed Boolean QP (MBQP) problems.

3.1.1 Data preparation

In the model

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) = \tfrac{1}{2}\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} + r \\ &\text{subject to:} && \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & && \boldsymbol{x} \in \chi = \chi_1 \times \chi_2 \times \cdots \times \chi_n \end{aligned}$$

the decision variable $\boldsymbol{x}$ is constrained to be either 0 or 1 for its first 100 components and to be nonnegative for 101st to 150th components. The Hessian matrix was set to $\boldsymbol{P} = \boldsymbol{Q}\boldsymbol{Q}^T$, and $\boldsymbol{Q}$, $\boldsymbol{q}$, and $\boldsymbol{A}$ were generated at random, satisfying normal distribution.

Parameter $\boldsymbol{b}$ was set to $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{x}_0$ where $\boldsymbol{x}_0$ is chosen at random from set χ.

3.1.2 Simulation results: Minimized objective value versus number of ADMM

iterations and parameter α

An important parameter in the ADMM iterations (see (2.40)) is α as it effects the algorithm's convergence in a critical manner. Often times the theoretical upper bound on α to ensure convergence (see Sec. 2.2.1) turns out to be too conservative. It was therefore decided to identify appropriate values of α.

The ADMM algorithm described in Section 2.4.3 was applied to solve the problem under several settings in terms of the value of parameter α in Eq. (2.40). Table 1 displays the minimized objective values, mean, and standard deviation as the given α is from 0.5 to 1, the algorithm also required at least 600 iterations to converge to a possible solution. All values are rounding to the integers. The primary and most important purpose of the standard deviation is to understand how the data set spreads out. A low standard deviation indicates that the values tend to be close to the average of the set (also known as the expected value), while a high standard deviation indicates that the values are distributed over a larger range. The three sigma rule tells us that 68% of the objective values fall within one standard deviation of the mean; 95% are within two standard deviations of the mean; 99.7% are within three standard deviations of the mean.

Table 1. Statistics of 70 initializations at different values of α

Value of α	Number of initializations	Minimized obj.	Mean	Standard deviation
0.5	70	2108	2272	139
0.6		2196	2524	179
0.7		2400	2767	188
0.8		2437	3063	249
0.9		2781	3385	284
1.0		2990	3617	297

Obviously, the method we use at present is linear searching algorithm, which is not efficient. Therefore, **fminbnd** searching algorithm is further applied to find the α value corresponding to the smallest minimized objective value.

As can be seen from the Fig.3, the **fminbnd** tests the value of the α set from 0 to 1 by running 600 iterations, α gets the value of 0.503074 and keeps changing at the last three decimal places. As a result, three decimal places are left with a value of 0.503. It is observed that in 600 iterations the smallest objective value the algorithm can get is 2108.

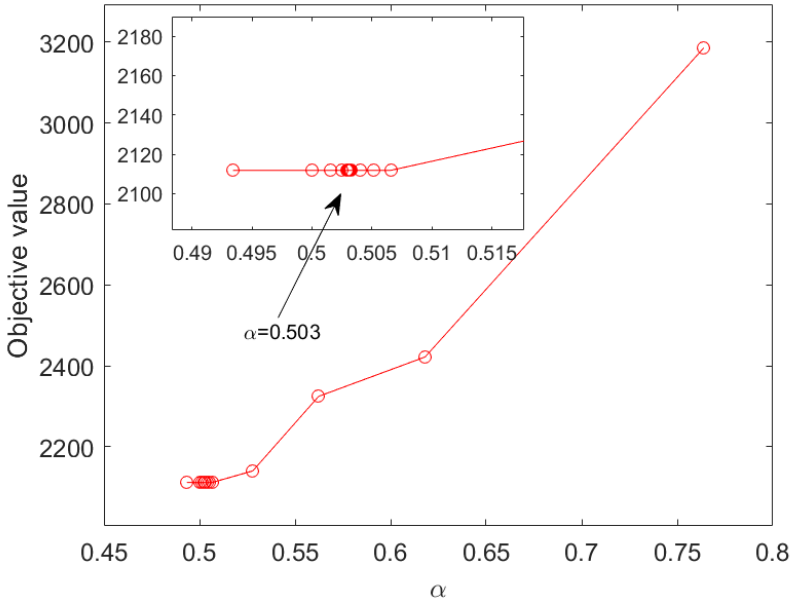

Figure 3. Objective value versus α

The algorithm's average run-time in the case of 600 iterations was found to be 3.2 seconds. As reported in [10], with the same parameters $\boldsymbol{P}, \boldsymbol{Q}, \boldsymbol{q}, r, \boldsymbol{b}$, and $\boldsymbol{A}$, the global solution $\boldsymbol{x}^{**}$ obtained by the commercial global solver MOSEK yielded $f(\boldsymbol{x}^{**}) = 2040$, representing a 3.3% reduction relative to that achieved by the ADMM-based algorithm. It is also noted that it took MOSEK more than 16 hours to secure the global solution $\boldsymbol{x}^{**}$ [10].

Table 2. Performance comparison of ADMM-based algorithm with MOSEK

Method	# of initializations	# of iterations	minimized obj.
ADMM	70	600	2108
MOSEK			2040

3.1.3 Constraint satisfaction

Based on the numerical simulations conducted, the time required by the polish step was about 1 second. After the ADMM iterations, a solution with improved constraint satisfaction may be obtained by executing a polish step under the circumstances of 70 initializations and 600 iterations.

Specifically, for the problem at hand, the constraint satisfaction was evaluated in terms of E_2 for linear equation $\boldsymbol{A}\boldsymbol{x}^* = \boldsymbol{b}$ and E_c for the last 50 components of $\boldsymbol{x}^*$, see Sec. 2.5.2 for the definitions of E_2 and E_c. The Boolean constraints for the first 100 components are always satisfied perfectly regardless of whether or not the polish step is implemented because each ADMM iteration includes a step that project the first 100 components of the current iterate onto set {0, 1}. Table 3 displays satisfaction of equality constraints in terms of E_2. The improvement by the polish technique appears to be significant. Table 3 also shows that good satisfaction of the inequality constraints was achieved with or without polish.

Displayed in the third column of Table 3.3c are the smallest values of the objective function obtained using 70 randomly selected initial points without the polish step; while the fourth column shows the smallest values of the objective function obtained using the same set of initial points, where the polish step was carried out. It is observed that the objective function was slightly increased 0.002784 after polish with 6 decimal places reserved. As pointed out in Sec. 2.5.2, the slight increase in the objective value is expected and the minimized values of the objective function after polish should be taken as the true achievable values of the objective function.

Table 3. Constraint satisfaction in terms of E_2, E_c, and minimized obj.

Test method	without polish	with polish
Equality constraints E_2	1.403×10^{-5}	7.616×10^{-10}
Inequality constraints E_c	0	0
Minimized objective value	2108	2108

As pointed out earlier, the ADMM-based method is merely a heuristic technique, and as such there is no guarantee to secure the global solution of the problem. This is not surprising because the problem at hand is *not* convex due to the presence of the Boolean constraints. On the other hand, it is intuitively clear that the probability of finding global minimizer or a good suboptimal solution shall increase with the number of independent random initial trials, and this was verified in the simulations as reported in Table 4 and Table 5 which list the results of by applying a total of 20 randomly generated data sets. With each random state (i.e. initial random seed), a total of 70 random initial points was generated to start the algorithm. With each initial point the algorithm was then performed by 1000 ADMM iterations, and the smallest objective value among 70 solution points is shown in the table. A point to note is that all numerical trials described here have utilized the *same* set of matrices the same matrices $\boldsymbol{P}$, $\mathbf{q}$, $\mathbf{A}$, and $\boldsymbol{b}$ that define the MIQP problem. The simulations produce two sets of results, the results obtained by the ADMM algorithm without polish are given in Table 4, while those obtained by ADMM with polish are given in Table 5. Minimized objective values are kept with 6 decimal places for accurately calculating mean and standard deviation.

Table 4. Performance without polish

random state	minimized obj.	equality constraints	inequality constraints
1	2379.917816	1.280×10^{-8}	0
2	2200.379829	1.392×10^{-5}	0
3	2113.110791	1.409×10^{-5}	0
4	2165.594249	1.402×10^{-5}	0
5	2217.018799	1.404×10^{-5}	0
6	2250.551708	1.386×10^{-5}	0
7	2424.519346	5.689×10^{-8}	0
8	2359.325493	3.981×10^{-6}	0

9	2186.141896	1.387×10^{-5}	0
10	2125.866011	1.411×10^{-5}	0
11	2183.055484	1.398×10^{-5}	0
12	2125.86602	1.400×10^{-5}	0
13	2400.9994	1.383×10^{-5}	0
14	2116.481569	1.391×10^{-5}	0
15	2134.276787	1.412×10^{-5}	0
16	2167.487995	8.836×10^{-10}	0
17	2355.053429	1.407×10^{-5}	0
18	2108.127412	1.403×10^{-5}	0
19	2197.559897	1.398×10^{-5}	0
20	2312.432457	1.382×10^{-5}	0

Table 5. Performance with polish

random state	minimized obj.	equality constraints	inequality constraints
1	2379.917814	1.391×10^{-10}	0
2	2200.38122	5.376×10^{-11}	0
3	2113.11305	2.217×10^{-10}	0
4	2165.594781	8.391×10^{-11}	0
5	2217.022597	4.810×10^{-11}	0
6	2250.553233	6.808×10^{-10}	0
7	2424.519335	1.410×10^{-9}	0
8	2359.325531	2.229×10^{-10}	0

9	2186.143189	2.786×10^{-10}	0
10	2125.868279	6.453×10^{-10}	0
11	2183.055725	1.034×10^{-9}	0
12	2125.868279	6.453×10^{-10}	0
13	2401.001274	2.847×10^{-10}	0
14	2116.481997	2.108×10^{-10}	0
15	2134.278672	2.018×10^{-10}	0
16	2167.487995	3.824×10^{-10}	0
17	2355.054647	2.673×10^{-10}	0
18	2108.130196	7.616×10^{-10}	0
19	2197.562158	3.387×10^{-11}	0
20	2312.433532	6.664×10^{-10}	0

The mean and standard deviation of the optimized objective values over the 20 random trials, for both scenarios of the ADMM algorithm with or without polish, are evaluated and the results are shown in Table 6. Also provided in the table are the mean and standard deviation of measure E_2 of the equality-constraints satisfaction over the 20 random trials for the two algorithmic scenarios.

From Table 6, we see that the mean of minimized objective values without polish was equal to 2226, which was practically the same as the mean value for the scenario with polish, and was 9% larger than the globally minimized value 2040. The standard deviation in this scenario was equal to 106 which, again, is the same as that of the scenario with polish. Both the mean and standard deviation of equality-constraints satisfaction E_2 for the case without polish are in the vicinity of 10^{-5}, while for the case with polish these values are much reduced to the vicinity of 10^{-10}.

Table 6. Mean and standard deviation of random trials

		without polish	with polish
minimized obj. value	mean	2226	2226
	standard deviation	106	106
equality constraints	mean	1.1×10^{-5}	3.6×10^{-10}
	standard deviation	0.5×10^{-5}	3.7×10^{-10}

3.2 Hybrid Vehicle Control

This example was also initiated from [10], where an MIQP problem arising from hybrid vehicle control system was addressed using ADMM-based heuristics. The hybrid vehicle consists of a battery, an electric motor/generator, and a heat engine in a parallel configuration. For a realistic model, there are several issues and assumptions that need to be taken into consideration [20], [21]. These include

(1) It is assumed that the demanded power P_t^{demand} at times $t = 0, \ldots, T-1$ are known in advance.

(2) The needed power may be obtained from both the battery and engine, hence the inequality constraint

$$P_t^{\text{batt}} + P_t^{\text{eng}} \geq P_t^{\text{demand}}$$

for $t = 0, \ldots, T-1$.

(3) The energy E_{t+1} currently stored in the battery can be described by

$$E_{t+1} = E_t - \tau P_t^{\text{batt}},$$

where τ is the length of time interval.

(4) The battery capacity is limited, hence the constraint

$$0 \leq E_t \leq E^{\max},$$

for all *t*, where $E^{\max}$ denotes maximum capacity of the battery.

(5) The terminal energy state of battery is penalized according to $g(E_T)$,where $g(E)$

obeys

$$g(E) = \eta(E^{\max} - E)^2$$

with $\eta \geq 0$.

(6) The engine's on and off is modelled with binary variable z_t. If the engine is on ($z_t = 1$), then $0 \leq P_t^{eng} \leq P^{\max}$, and $\alpha(P_t^{\text{eng}})^2 + \beta P_t^{\text{eng}} + \gamma$ units of fuel are consumed. If the engine is off ($z_t = 0$), no fuel is consumed and $P_t^{\text{eng}} = 0$. In summary, the power constraint can be expressed as

$$0 \leq P_t^{\text{eng}} \leq P^{\max} z_t$$

and the fuel cost is given by

$$f(P_t^{\text{eng}}, z_t) = \alpha(P_t^{\text{eng}})^2 + \beta P_t^{\text{eng}} + \gamma z_t \tag{3.1}$$

(7) Finally, turning of the engine on from the "off" state is done at a cost $\delta \geq 0$. Specifically, at time *t*, the cost is equal to $\delta(z_t - z_{t-1})_+$ where $(\cdot)_+$ denotes the positive part.

To summarize, the hybrid vehicle control problem can be formulated as

$$\text{minimize} \quad \eta(E_T - E^{\max})^2 + \sum_{t=0}^{T-1}\left[f(P_t^{\text{eng}}, z_t) + \delta(z_t - z_{t-1})_+ \right] \tag{3.2a}$$

$$\text{subject to:} \quad E_{t+1} = E_t - \tau P_t^{\text{batt}} \tag{3.2b}$$

$$P_t^{\text{batt}} + P_t^{\text{eng}} \geq P_t^{\text{demand}} \tag{3.2c}$$

$$z_t \in \{0,1\} \tag{3.2d}$$

where $t = 0, 1, \ldots, T-1$. The variables involved in problem (3.2) are continuous variables $\{E_t, t = 1,...,T-1\}$, $\{P_t^{\text{batt}}, t = 0,1,..,T-1\}$, and $\{P_t^{\text{eng}}, t = 0,1,..,T-1\}$, and Boolean variables $\{z_i, t = 0,1,..,T-1\}$. The objective function in (3.2a) has a composite structure with a convex quadratic component function of continuous variables, namely,

$$\eta(E_T - E^{\max})^2 + \sum_{t=0}^{T-1}\left[\alpha(P_t^{\text{eng}})^2 + \beta P_t^{\text{eng}} \right]$$

and a component function of Boolean variables, namely,

$$\sum_{t=0}^{T-1}\left[\gamma z_t + \delta(z_t - z_{t-1})_+\right]$$

Also note that the constraints involved in problem (3.2) includes two sets of linear inequalities of continuous variables and a set of Boolean constraints. As such, problem (3.1) fits nicely into the class of MIQP problems studied in this report.

3.2.1 Simulation results: Minimized objective value versus number of ADMM iterations and parameter α

In the simulations described below, we follow reference [10] to set the numerical values of the known parameters in problem (3.2) as follows:

$$\beta = 1,\ \gamma = 1,\ \delta = 1,\ \ \eta = 1,\ \tau = 4,\ \ E^{\max} = 40,\ \ E_0 = 40,\ \text{and}\ \ z_{-1} = 0$$

The ADMM algorithm described in Section 2.4.3 was applied to solve the problem under several settings in terms of the value of parameter α in Eq. (2.40) and the number of iterations. It turned out that for α in the range between 2 and 4.5, the algorithm required at least 4000 iterations to converge to a solution. Table 7 displays the algorithm's performance in terms of minimized objective value obtained using a given α after certain number of iterations for convergence. From the Table 7, it is also observed that the best performance is achieved when α is set to 2. We recorded the minimized objective values corresponding to 5 initializations, and then calculated the standard deviation and the mean of the recorded values. A low standard deviation of each α indicates that these values tend to be close to the average of the set (also known as the expected value).

Table 7. Statistics of 5 initializations at different values of α

Value of α	Number of initializations	Smallest minimized obj.	Mean	Standard deviation
2	5	137.75	138.03	0.15
2.5		138.33	138.74	0.60
3		138.41	141.50	1.85
3.5		140.96	143.25	2.87

4		141.14	145.48	2.90
4.5		141.28	146.06	3.02

The next set of simulations aims to a fine tuning of the value of parameter α in a vicinity of 2 for possible performance enhancement. Fig.4 provides the results obtained by running the **fminbnd** algorithm with 4000 iterations and value of α set from 1.50 to 2.50. α gets the value of 1.64382 and keeps changing at the last two decimal places. As a result, three decimal places are left with a value of 1.644. We see that α = 1.644 offers the best objective value 137.42.

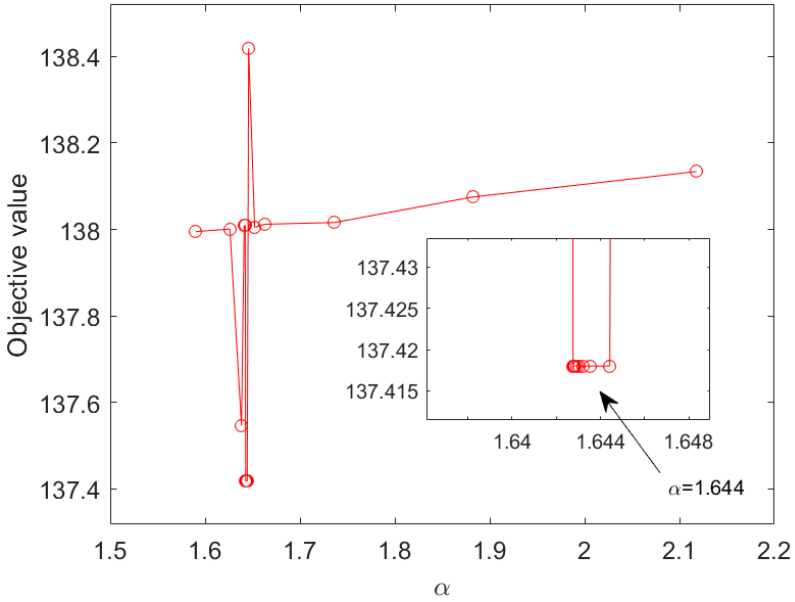

Figure 4. Objective value versus α

As the final stage of performance fine tuning, the value α was fixed to 1.644 while running the algorithm by increasing the number of iterations to identify the better solution. It can be found that the best minimized objective value 137.36 is obtained as 5200 iterations. To conclude, it has identified that as α = 1.644 and an adequate number of iterations, the algorithm yields the best objective value 137.36.

3.2.2 Simulation results: Constraint satisfaction with and without polish

As an integral part of our simulations and performance evaluations, the polish step was applied following the ADMM iterations, and constraint satisfactions with and without

polish were compared in terms of the numerical measures of constraint satisfaction defined in Section 2.5.2 under the circumstances of α = 1.643 and 5200 iterations. Specifically, we follow Eq. (2.45), namely

$$E_2 = \frac{1}{\sqrt{p}} \| \boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b} \|_2 ,$$

to evaluate the L_2 error of the equation constraints in (3.2b) . For the example conducted in the simulations, T was set to 72 hence there are p = 72 equality constraints. Table 8 also displays error E_2 with and without polish. It is observed that the E_2 error is much reduced when a polish step is applied.

To examine the inequality constraints in (3.2c), we define

$$d_t = P_t^{\text{batt}} + P_t^{\text{eng}} - P_t^{\text{demand}}$$

and write the constraints in (3.2c) as

$$d_t \geq 0 \qquad \text{for } t = 0, 1, \ldots, T-1$$

Under the circumstances, the error measure E_c defined in Eq. (2.49) becomes

$$E_c = \frac{1}{T} \sum_{t=1}^{T-1} \max\{-d_t, 0\}$$

where T = 72 in the simulation. Evidently, value E_c = 0 would indicate that all inequality constraints are satisfied while a E_c > 0 implies that some inequality constraints have been violated and the degree of violation is reflected by the actual value of E_c. Table 8 provides numerical evaluation of error E_c with and without polish. We see that the polish step leads to a solution at which the inequality constraints in (3.2) are all satisfied, while small degree of constraint violation occurs at the solution obtained without polish.

To better observe the differences between with and without polish. The minimized objective value is kept at 6 decimal places, among which the minimized objective values with and without polishing are 137.36 and 137.30, respectively. To our surprise, the solution obtained with the polish step also helps reduce the objective function a bit further.

Table 8. Constraint satisfaction in terms of E_2, E_c, and minimized obj.

Test method	without polish	with polish
Equality constraints E_2	1.3×10^{-4}	1.3×10^{-16}
Inequality constraints E_c	1.7×10^{-4}	0
Minimized objective value	137.36	137.30

3.2.3 Remarks

Fine tuning of the design parameter has yielded near optimal choices of $\alpha = 1.644$ which, in conjunction with the run of 5200 iterations, produces a better solution with the smallest objective value 137.30. The CPU time consumed by the ADMM-based algorithm was about 3.34 seconds. For reference, it was reported in [10] that it took MOSEK about 15 seconds to identify a solution with practically the same performance as the solution obtained by the ADMM algorithm.

3.3 Economic Dispatch

This application was initiated in reference [4]. As mentioned in Chapter 1 (see Sec. 1.1.2), the goal of the economic dispatch problem is to generate a given amount of electricity for several sets of generators at lowest cost possible. The parameters and design variables involved in the problem as well as the constraints imposed by the problem at hand are described as follows.

(1) The fuel cost of the *i*th generator is modelled as a quadratic function of its output power P_i (in MW), namely,

$$F_i(P_i) = a_i + b_i P_i + c_i P_i^2$$

where a_i, b_i, and c_i are cost coefficients for the *i*th generator. Thus the total fuel cost F, that needs to be minimized, is given by

$$F = \sum_{i\in\Omega} F_i(P_i)$$

where Ω is the set of all on-line generators.

(2) The total power for the set of all on-line generators is constrained to be equal to

total demand power P_D, that is,

$$\sum_{i\in\Omega} P_i = P_D.$$

(3) The spinning reserve is an additional generating capacity obtained by increasing the power of the generators that are already connected to the power system [22]. The total power of the spinning reserve contribution S_i of the ith generator is constrained to be greater than or equal to the spinning reserve requirement S_R, that is,

$$\sum_{i\in\Omega} S_i \geq S_R$$

Furthermore, for the generators without prohibited operating zones, the spinning reserve contribution S_i is constrained to be equal to the smaller value of $\{ P_i^{\max} - P_i$, $S_i^{\max} \}$. On the other hand, for the generators with prohibited operating zones, the spinning reserve contribution S_i is set to 0. In summary, the constraints for the spinning reserve contributions $\{S_i\}$ are given by

$$S_i \leq \min\left\{P_i^{\max} - P_i, S_i^{\max}\right\} \quad \forall i \in \Omega - \omega \tag{3.3}$$

$$S_i = 0, \ \forall i \in \omega$$

where $P_i^{\max}$ is the maximum generating power of the ith generator, $S_i^{\max}$ is the maximum spinning reserve contribution of generator i, and ω is the set of on-line generators with prohibited operating zones.

(4) The output power of each generator without prohibited operating zones is constrained to be in a certain range

$$P_i^{\min} \leq P_i \leq P_i^{\max} \quad \forall i \in \Omega - \omega$$

where $P_i^{\min}$ and $P_i^{\max}$ denotes the lower and upper generating limits for the ith generator for $i \in \Omega - \omega$.

(5) For the generators with prohibited operating zones, each generator has $k-1$ prohibited zones and k *disjoint* operating sub-regions $(\hat{P}_{ik}^L, \hat{P}_{ik}^U)$, and the output power is constrained as

$$\hat{P}_{ik}^L \leq P_i \leq \hat{P}_{ik}^U, \ \forall i \in \omega, \ k = 1, \ ..., \ K$$

with $\hat{P}_{i1}^{L} = P_i^{\min}$ and $\hat{P}_{iK}^{U} = P_i^{\max}$.

The disjoint nature of operating sub-regions implies that the feasible region of the problem at hand is not a connected region and hence a nonconvex feasible region. As will be shown below, a natural treatment of the disjoint forbidden zones leads to a MIQP formulation. To this end, auxiliary design variables are introduced to deal with the disjoint operating sub-regions:

Y_{ik}: It is set to 1 if the *i*th generator operates within its power output range; otherwise, it is set to 0.

Θ_{ik}: it is set to P_i if the *i*th generator operates within its power output range (i.e., if Y_{ik} = 1); otherwise, it is set to 0.

Since a generator with prohibited operating zones can operates only in one of the K possible ranges, the Boolean variables $\{Y_{ik}\}$ are constrained by

$$\sum_{k=1}^{K} Y_{ik} = 1, \ \forall i \in \omega$$

Similarly, $\{\Theta_{ik}\}$ are related to power output via the following constraint equation by using the following two constraints

$$P_i = \sum_{k=1}^{K} \Theta_{ik} \qquad \forall i \in \omega$$

$$\hat{P}_{ik}^{L} Y_{ik} \le \Theta_{ik} \le \hat{P}_{ik}^{U} Y_{ik}, \ \forall i \in \omega, \ k = 1, \ ..., K$$

In summary, the problem at hand can be formulated as the constrained problem

$$\text{minimize} \qquad \sum_{i \in \Omega} F_i(P_i) = \sum_{i \in \Omega} a_i + b_i P_i + c_i P_i^2 \tag{3.5a}$$

$$\text{subject to:} \quad \sum_{i \in \Omega} P_i = P_D \tag{3.5b}$$

$$\sum_{i \in \Omega} S_i \ge S_R \tag{3.5c}$$

$$S_i \le \min\left\{P_i^{\max} - P_i, S_i^{\max}\right\} \quad \forall i \in \Omega - \omega \tag{3.5d}$$

$$S_i = 0, \qquad \forall i \in \omega \tag{3.5e}$$

$$P_i^{\min} \le P_i \le P_i^{\max} \quad \forall i \in \Omega - \omega \tag{3.5f}$$

$$\sum_{k=1}^{K} Y_{ik} = 1, \quad \forall i \in \omega \tag{3.5g}$$

$$P_i = \sum_{k=1}^{K} \Theta_{ik}, \quad \forall i \in \omega \tag{3.5h}$$

$$\hat{P}_{ik}^{L} Y_{ik} \le \Theta_{ik} \le \hat{P}_{ik}^{U} Y_{ik}, \quad \forall i \in \omega,\ k = 1,\ ...,\ K \tag{3.5i}$$

$$Y_{ik} \in \{0,1\} \tag{3.5j}$$

$$S_i \ge 0, \quad \forall i \in \Omega \tag{3.5k}$$

A careful inspection of the above formulation leads to a simplified formulation: First, note that the constraints in (3.5c) and (3.5e) may be combined to

$$\sum_{i \in \Omega - \omega} S_i \ge S_R$$

and the constraints in (3.5k) and (3.5e) may be combined to write

$$S_i \ge 0 \quad \forall i \in \Omega - \omega$$

Second, note that the objective function is independent of variables $\{S_i,\ \forall i \in \Omega - \omega\}$, and the constraints in (3.5d) and modified (3.5c) and (3.5k) can be treated *after* variables $\{P_i\}$ are optimized. To be precise, variables $\{P_i, \forall i \in \Omega\}$ are optimized by solving

$$\text{minimize} \quad \sum_{i \in \Omega} F_i(P_i) = \sum_{i \in \Omega} a_i + b_i P_i + c_i P_i^2 \tag{3.6a}$$

$$\text{subject to:} \quad \sum_{i \in \Omega} P_i = P_D \tag{3.6b}$$

$$P_i^{\min} \le P_i \le P_i^{\max} \quad \forall i \in \Omega - \omega \tag{3.6c}$$

$$\sum_{k=1}^{K} Y_{ik} = 1, \quad \forall i \in \omega \tag{3.6d}$$

$$P_i = \sum_{k=1}^{K} \Theta_{ik}, \quad \forall i \in \omega \tag{3.6e}$$

$$\hat{P}_{ik}^{L} Y_{ik} \le \Theta_{ik} \le \hat{P}_{ik}^{U} Y_{ik}, \quad \forall i \in \omega,\ k = 1,\ ...,\ K \tag{3.6f}$$

$$Y_{ik} \in \{0,1\} \tag{3.6g}$$

which evidently is an MIQP problem. Once the solution of (3.6), denoted by $\{ P_i^*$,

$\forall i \in \Omega\}$, is obtained, variables $\{S_i, \forall i \in \Omega - \omega\}$ are obtained by solving the feasibility problem

$$\text{Find } \{S_i\} \tag{3.7a}$$

$$\text{subject to: } \sum_{i \in \Omega - \omega} S_i \geq S_R \tag{3.7b}$$

$$0 \leq S_i \leq \min\{P_i^{\max} - P_i \, , S_i^{\max}\} \quad \forall i \in \Omega - \omega \tag{3.7c}$$

A simple solution of problem (3.7) can be deduced as follows. Let $q_i = \min\{P_i^{\max} - P_i^*, S_i^{\max}\}$ so that the constraint in (3.7c) becomes

$$0 \leq S_i \leq q_i \quad \forall i \in \Omega - \omega$$

Now consider a solution S_i that assumes the form $S_i = \gamma q_i$. Obviously {Si} of such form satisfies the constraints in (3.7c) as long as γ lies in the range $0 \leq \gamma \leq 1$. Now we choose a value $\gamma \in [0, 1]$ to further satisfy the constraint in (3.7b), namely,

$$\sum_{i \in \Omega - \omega} S_i = \sum_{i \in \Omega - \omega} \gamma q_i = \gamma \sum_{i \in \Omega - \omega} q_i \geq S_R$$

i.e.,

$$\gamma \geq \frac{S_R}{\sum_{i \in \Omega - \omega} q_i}$$

It follows the smallest value of γ satisfying (3.6b) is given by

$$\gamma^* = \frac{S_R}{\sum_{i \in \Omega - \omega} q_i} \tag{3.8}$$

If $\gamma^* \leq 1$, a solution of problem (3.7) is given by

$$S_i = \gamma^* q_i \qquad \forall i \in \Omega - \omega \tag{3.9}$$

otherwise, problem (3.7) admits no solution.

3.3.1 Data set and model for simulations

Our simulation studies for the problem at hand considers a system having 4 generators with the characteristics

$$F_i(P_i) = 500 + 10P_i + 0.001P_i^2 \quad \text{for } i = 1, 2, 3, 4$$

where the objective values are in unit \$/hour, and

$$P_i^{\min} = 100 \text{ MW}, P_i^{\max} = 500 \text{ MW}, \text{ and } S_i^{\max} = 50 \text{ MW} \quad \text{for} \quad \forall i \in \Omega - \omega$$

The problem requires the demand of 1375 MW and a system spinning reserve requirement of 100 MW, thus P_D = 1375 MW and S_R = 100 MW [23]. In our simulations, each of generators # 1 and # 2 has two prohibited operating zones, hence $\Omega = \{1, 2, 3, 4\}$ and $\omega = \{1, 2\}$. Table 9 provides numerical values of the parameters that define these prohibited operating zones for generators # 1 and # 2 in terms of disjoint operating sub-regions $(\hat{P}_{ik}^L, \hat{P}_{ik}^U)$. For example, the 3 disjoint operating sub-regions for generator # 1 are given by

$$(\hat{P}_{11}^L, \hat{P}_{11}^U),\ (\hat{P}_{12}^L, \hat{P}_{12}^U),\ (\hat{P}_{13}^L, \hat{P}_{13}^U)$$

where $\hat{P}_{11}^L = P_i^{\min}$ =100 MW and $\hat{P}_{13}^U = P_i^{\max} = 500$ MW. Following the parameters from Table 9, the ranges of the three operating sub-regions for generator # 1 are given by

(100 MW, 200 MW), (250 MW, 300 MW), (350 MW, 500 MW)

Table 9. Prohibited zones for generators # 1 and # 2

Generator	Zone 1 (MW)	Zone 2 (MW)
1	[200 - 250]	[300 - 350]
2	[210 - 260]	[310 - 360]

The MIQP problem in (3.6) includes both equality and inequality constraints and hence fits into the model addressed in Eq. (2.51). Following Sec. 2.6, a nonnegative decision variable vector $\boldsymbol{\xi}$ is introduced to convert the inequality constraints to equality constraints. In doing so, constraints (3.6c) and (3.6f) become

$$P_i + \boldsymbol{\xi}_j = P_i^{\max},\ \forall i \in 3, 4,\ j \in 1,\ 2$$
$$-P_i + \boldsymbol{\xi}_j = -P_i^{\min},\ \forall i \in 3, 4,\ j \in 3, 4$$
$$\Theta_{ik} - \hat{P}_{ik}^U Y_{ik} + \boldsymbol{\xi}_j = 0,\ \forall i,k \in \{1,1\}, \{1,2\}, \{1,3\}, \{2,1\}, \{2,2\}, \{2,3\},\ j \in 5, ..., 10$$
$$-\Theta_{ik} + \hat{P}_{ik}^L Y_{ik} + \boldsymbol{\xi}_j = 0,\ \forall i,k \in \{1,1\}, \{1,2\}, \{1,3\}, \{2,1\}, \{2,2\}, \{2,3\},\ j \in 11, ..., 16$$

In addition, under the current circumstances the equality constraints in (3.6b), (3.6d),

and (3.6e) can be made more specific as

$$\sum_{i\in\Omega} P_i = P_D,\ \forall i \in 1,\ 2,\ 3,\ 4$$

$$\sum_{k=1}^{K} Y_{ik} = 1,\quad \forall i,k \in \{1,1\},\ \{1,2\},\ \{1,3\},\ \{2,1\},\ \{2,2\},\ \{2,3\}$$

$$P_i = \sum_{k=1}^{K} \Theta_{ik},\ \forall i,k \in \{1,1\},\ \{1,2\},\ \{1,3\},\ \{2,1\},\ \{2,2\},\ \{2,3\}$$

which can in turn be combined into a standard form of equality constraints as $\boldsymbol{Ax} = \boldsymbol{b}$ where $\boldsymbol{A} \in R^{21\times 32}$, $\boldsymbol{b} \in R^{21\times 1}$, and decision variable $\boldsymbol{x} \in R^{32\times 1}$ is defined by

$$\boldsymbol{x} = \left[Y_{1,1}\ Y_{1,2}\ Y_{1,3}\ Y_{2,1}\ Y_{2,2}\ Y_{2,3}\ \theta_{1,1}\ \theta_{1,2}\ \theta_{1,3}\ \theta_{2,1}\ \theta_{2,2}\ \theta_{2,3}\ P_1\ P_2\ P_3\ P_4\ \boldsymbol{\xi}_1\ \cdots\ \boldsymbol{\xi}_{16} \right]^T$$

A point to note is that by definition parameters $\theta_{ik}, P_i, \boldsymbol{\xi}_j$ are non-negative. It is straightforward to verify that the constraints $P_i \geq 0$ and $\Theta_{ik} \geq 0$ are automatically guaranteed by constraints (3.6c), (3.6e), and (3.6f), while the non-negativity of $\boldsymbol{\xi}_j$ need to be imposed additional constraints. The MIQP problem can now be formulated as

$$\text{minimize} \quad \sum_{i=1}^{4} F_i(P_i) = 2000 + \sum_{i=1}^{4} \left(0.001 P_i^2 + 10 P_i \right) \qquad (3.11a)$$

$$\text{subject to:} \quad \boldsymbol{Ax} = \boldsymbol{b} \qquad (3.11b)$$

$$Y_{ik} \in \{0,1\} \quad \text{for } i = 1,2;\ k = 1,2,3 \qquad (3.11c)$$

$$\boldsymbol{\xi}_j \geq 0 \qquad \text{for } j = 1,\ ...,16 \qquad (3.11d)$$

The objective function in (3.11a) can be written as a function of decision variable $\boldsymbol{x}$ in the standard form as $f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x} + \boldsymbol{q}^T \boldsymbol{x} + r$ where $\boldsymbol{P} \in R^{32\times 32}$ is a positive semidefinite diagonal matrix given by

$$\boldsymbol{P} = \begin{bmatrix} P_{(1,1)} & & & & & & & \\ & \ddots & & & & & & \\ & & P_{(13,13)} & & & & & \\ & & & P_{(14,14)} & & & & \\ & & & & P_{(15,15)} & & & \\ & & & & & P_{(16,16)} & & \\ & & & & & & \ddots & \\ & & & & & & & P_{(21,21)} \end{bmatrix} = \begin{bmatrix} 0 & & & & & & & \\ & \ddots & & & & & & \\ & & 0.002 & & & & & \\ & & & 0.002 & & & & \\ & & & & 0.002 & & & \\ & & & & & 0.002 & & \\ & & & & & & \ddots & \\ & & & & & & & 0 \end{bmatrix},$$

$\boldsymbol{q} \in R^{32 \times 1}$ is given by

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_{13} \\ q_{14} \\ q_{15} \\ q_{16} \\ \vdots \\ q_{21} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 10 \\ 10 \\ 10 \\ 10 \\ \vdots \\ 0 \end{bmatrix}$$

and r = 2000. Having obtained a solution $\boldsymbol{x}^*$ of problem (3.11), we denote its 15$^{\text{th}}$ and 16$^{\text{th}}$ components as P_3^* and P_4^*, respectively, and use them to specify the feasibility problem (3.7) for variable S_3 and S_4 as

$$\text{Find } S_3 \text{ and } S_4$$

$$\text{subject to: } S_3 + S_4 \geq 100$$

$$0 \leq S_i \leq \min\{500 - P_i\ , 50\} \quad \text{for } i = 3, 4$$

where $P_i^{\max}$ = 500 MW, S_R = 100 MW, and $S_i^{\max}$ = 50 MW have been used for problem specification. A solution of (3.11) gives $P_3^* = 344.16$ and $P_4^* = 344.16$ which lead to $\beta_3 = \beta_4 = 50$, see (3.8). Following (3.9) and (3.10), we obtain $\gamma^* = 1$, S_3 = 50 MW and S_4 = 50 MW.

3.3.2 Simulation results: Minimized objective value versus number of ADMM iterations and parameter α

The ADMM algorithm described in Section 2.4.3 was applied to solve problem (3.11) with several settings in terms of the value of parameter α in Eq. (2.40) and the number of iterations. It turned out that for α in the range between 6.5 and 6.9, the algorithm required at least 1200 iterations to converge to a solution. The Table 10 reports the minimized objective value, mean, and standard deviation when α was from 6.5 to 6.9 with 1200 iterations. For each α, we recorded the minimized objective values corresponding to 5 initializations, and then calculated the standard deviation and the mean of the recorded values.

Table 10. Statistics of 5 initializations at different values of α

Value of α	Number of initializations	Smallest minimized obj.	Mean	Standard deviation
6.5	5	16194	16254	58
6.6		16235	16283	48
6.7		16216	16230	16
6.8		16189	16202	19
6.9		16200	16224	20

The next set of simulations aims to a fine tuning of the value of parameter α from 6.5 to 7.0 for possible performance enhancement. Fig.5 provides the results obtained by running the **fminbnd** algorithm with 1200 iterations and value of α set from 6.5 to 7.0. Finally, since α is constantly changing between 6.773 and 7.774, the α value should be 6.7735. As a result, we observed that α = 6.7735 offers the least objective value 16189.

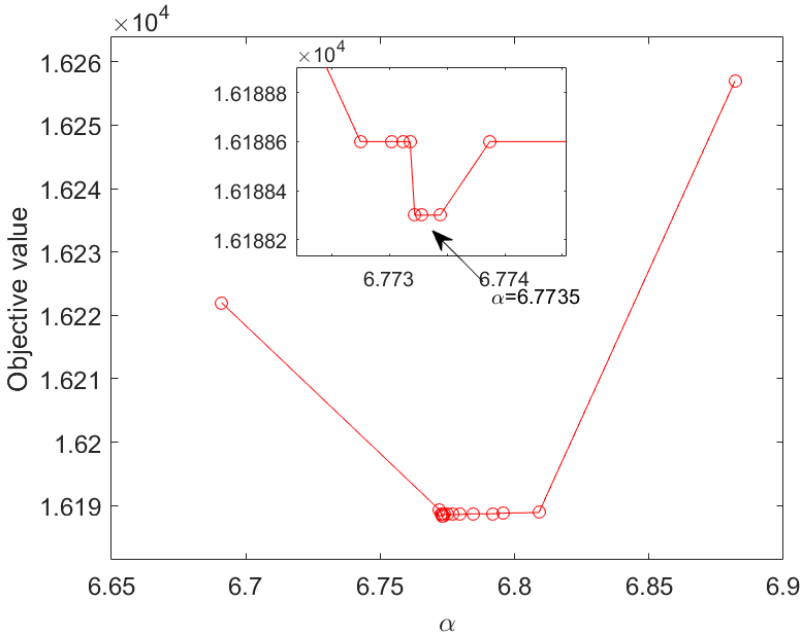

Figure 5. Objective value versus α

As the final stage of performance fine tuning, the value α was fixed to 6.7735 while running the algorithm by increasing the number of iterations to identify the better solution. The results where we observe that with α = 6.7735, 1550 iterations, the objective value is around 16027 and a better value is obtained. The ADMM algorithm yielded a solution at which the smallest objective value 16027 was achieved, which turns out to be less than that obtained in [23], which was 16223.2125. It is important to stress that the solution obtained by the ADMM algorithm at this stage of simulation is not yet ready to be taken as the final solution of the design problem at hand: what remains to be done is to apply the polish step as a follow-up and then evaluate the performance of the solutions in terms of constraint satisfaction.

3.3.3 Simulation results: Constraint satisfaction with and without polish

As an integral part of our simulations and performance evaluations, the polish step was also applied following the ADMM iterations, and constraint satisfactions with and without polish were compared in terms of the numerical measures of constraint satisfaction defined in Section 2.5.2. Specifically, we follow Eq. (2.45), namely

$$E_2 = \frac{1}{\sqrt{p}} \| \boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b} \|_2 ,$$

to evaluate the L_2 error of the equation constraints. As mentioned, $\boldsymbol{b} \in R^{p\times 1}$ with p = 21. Table 11 displays error E_2 with and without polish. It is observed that the E_2 error is much reduced when a polish step is applied.

To examine satisfaction of the inequality constraints in (3.11d), we evaluate error measure E_c which in the present case is defined by

$$E_c = \frac{1}{16}\sum_{j=1}^{16} \max\{-\boldsymbol{\xi}_j, 0\}$$

Evidently, value $E_c = 0$ would indicate that all inequality constraints are satisfied while an $E_c > 0$ implies that some inequality constraints in (3.11d) are violated and the degree of violation is reflected by the actual value of E_c. Table 11 provides numerical evaluation of error E_c with and without polish. As can be seen, the error of inequality constraint after polish was increased to 0.5×10^{-6} which is practically zero especially for the current problem because its primary decision variables $\{P_i\}$ are in a range of several hundreds.

To better observe the differences between with and without polish. The minimized objective value is kept at 6 decimal places, among which the minimized objective values with and without polishing are 16027.2102 and 16223.2125, respectively. It is observed that the minimized objective value was increased by 1.223%. Based on the analysis in Sec. 2.5.2, this is not surprising and should be regarded as a cost of improved *overall* constraint satisfaction and reliable design.

Table 11. Constraint satisfaction in terms of E_2, E_c, and minimized obj.

Test method	without polish	with polish
Equality constraints E_2	12.29	1.9×10^{-6}
Inequality constraints E_c	0	0.5×10^{-6}
Minimized objective value	16027.2102	16223.2125

3.3.4 Remarks

Fine tuning of the design parameter has yielded near optimal choices of α = 6.7735 which, in conjunction with the run of 1550 iterations, produces a global minimum with the value of 16223.2125. The CPU time consumed by the ADMM-based algorithm was about 0.0298s. For reference, [23] solved the same problem using commercial solver SBB and CONOPT in 0.1s and the global solution obtained yielded an objective value of 16223.213 which is practically the same as that of the solution the ADMM algorithm produced.

Chapter 4

Concluding Remarks

In this project, we delve into an important class of constrained nonconvex problems known as mixed-integer quadratic programming (MIQP). The popularity of MIQP is primarily due to the fact that many real-world problems can be described via MIQP models. The development of efficient MIQP algorithms has been an active and rapidly evolving field of research. As a matter of fact, previously well-known techniques for MIQP problems such as branch-and-bound and branch-and-cut are found unsuitable for large-scale or online MIQP problems where algorithm's computational efficiency is a crucial factor. In this regard, the alternating direction method of multipliers (ADMM) as a heuristic has shown to offer satisfactory suboptimal solutions with much improved computational complexity relative to global solvers based on for example branch-and-bound. This project provides the necessary details required to understand the ADMM-based algorithms as applied to MIQP problems. The report also includes three illustrative examples to demonstrate the effectiveness of the ADMM algorithm through numerical simulations and performance comparisons.

The implementation of the ADMM-based algorithm involved in this project uses CPU only. As a future project, using GPU to accelerate the implementation may be a topic for consideration. From an algorithmic perspective, the issue of preconditioning appears to be worthwhile to investigate for the sake of improved convergence for the ADMM algorithm, especially because the number of iterations required by the current version of the algorithm remain large.

References

[1] A. Antoniou and W.-S. Lu, *Practical Optimization – Algorithms and Engineering Applications,* Springer, 2007.

[2] Sayed, Ali H, "Adaptation, learning, and optimization over networks." *Foundations and Trends® in Machine Learning,* vol. 7, no. 4-5, pp. 311-801, 2014.

[3] D. Oulai, S. Chamberland, and S. Pierre, "A new routing-based admission control for MPLS network," *IEEE Communications Letters,* vol. 11, no. 2, pp. 216-218, 2007.

[4] L. G. Papageorgiou and E. S. Fraga, "A mixed integer quadratic programming formulation for the economic dispatch of generators with prohibited operating zones," *Electric Power Systems Research,* vol. 77, no. 10, pp. 1292-1296, 2007.

[5] J. P. S. Catalão, H. M. I. Pousinho, and V. M. F. Mendes, "Scheduling of head-dependent cascaded hydro systems: Mixed integer quadratic programming approach," *Energy Conversion and Management,* vol. 51, no. 3, pp. 524-530, 2010.

[6] N. Murgovski, L. Johannesson, J. Sjöberg, and B. Egardt, "Component sizing of a plug-in hybrid electric powertrain via convex optimization," *Mechatronics,* vol. 22, no. 1, pp. 106-120, 2012.

[7] J. Eckstein and W. Yao, "Understanding the convergence of the alternating direction method of multipliers: Theoretical and computational perspectives," *Pacific Journal of Optimization,* vol. 11, pp. 619-644, 2015.

[8] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstain, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning,* vol. 3, no. 1, pp. 1-122, 2010.

[9] S. Diamond, R. Takapoui, and S. Boyd, "A general system for heuristic minimization of convex functions over nonconvex sets," *Optimization Methods and Software,* 2017.

[10] R. Takapoui, N. Moehle, S. Boyd, and A. Bemporad, "A simple effective heuristic for embedded mixed-integer quadratic programming," *Int. Journal of Control,* 2017.

[11] Hochba, D. S. (Ed.), “Approximation algorithms for NP-hard problems”, *ACM Sigact News*, vol. 28, no. 2, pp. 40-52, 1997.

[12] Vance, P. H., Barnhart, C., Johnson, E. L., & Nemhauser, G. L., “Solving binary cutting stock problems by column generation and branch-and-bound”, *Computational optimization and applications*, vol. 3, no.2, pp. 111-130, 1994

[13] http://web.mit.edu/15.053/www/AMP-Chapter-09.pdf

[14] Hochba, D. S. (Ed.). Approximation algorithms for NP-hard problems. *ACM Sigact News*, vol.*28, no.*2, pp.40-52, 1997.

[15] Lawler, E. L., & Wood, D. E. “Branch-and-Bound Methods: A Survey,” *Operations Research*, vol. 14, no. 4, pp.699–719, 1966.

[16] Stubbs, R., Mehrotra, S. A branch-and-cut method for 0-1 mixed convex programming. Math. Program. 86, 515–532, 1999.

[17] T. Goldstein, B. O’Donoghue, S. Setzer, and R. Baraniuk, “Fast alternating direction optimization methods,” *SIAM J. Imaging Sciences*, vol. 7, no. 3, pp. 1588-1623, 2014.

[18] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, 1989.

[19] P. Giselsson and S. Boyd, “Diagonal scaling in Douglas-Rachford splitting and ADMM,” *Proc.* 53rd *IEEE Conf. on Decision and Control*, pp. 5033-5039, Los Angeles, CA, 2014.

[20] Muta, Koichiro, Makoto Yamazaki, and Junji Tokieda. *Development of new-generation hybrid system THS II-Drastic improvement of power performance and fuel economy*. No. 2004-01-0064. SAE Technical Paper, 2004.

[21] D. W. Gao, C. Mi and A. Emadi, “Modeling and Simulation of Electric and Hybrid Vehicles.” *Proceedings of the IEEE*, vol. 95, no. 4, pp. 729-745, April 2007.

[22] https://en.wikipedia.org/wiki/Operating_reserve

[23] F. N. Lee, and A. M. Breipohl, “Reserve constrained economic dispatch with prohibited operating zones.” *IEEE transactions on power systems*, vol. 8, no. 1 pp. 246-254, 1993.

[24] J. Clausen, "*Branch and bound algorithms — principles and examples*", Technical Report, *University of Copenhagen*, 1999.

[25] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming," version 2.0, http://cvxr.com/cvx, September 2013.

[26] M. Grant and S. Boyd, "Graph implementations for nonsmooth convex programs," *Recent Advances in Learning and Control*, V. Blondel, S. Boyd, and H. Kimura, editors, pp. 95-110, Lecture Notes in Control and Information Sciences, Springer, http://stanford.edu/~boyd/graph_dcp.html, 2008.