INTERNATIONAL JOURNAL OF CONTROL, 2017
https://doi.org/10.1080/00207179.2017.1316016

Taylor & Francis
Taylor & Francis Group

A simple effective heuristic for embedded mixed-integer quadratic programming

Reza Takapoui[a], Nicholas Moehle[b], Stephen Boyd[c] and Alberto Bemporad[d]

[a]Department of Electrical Engineering, Stanford University, Stanford, USA; [b]Mechanical Engineering Department, Stanford University, Stanford, USA; [c]Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, USA; [d]IMT Institute for Advanced Studies Lucca, Lucca, Italy

ABSTRACT

In this paper, we propose a fast optimisation algorithm for approximately minimising convex quadratic functions over the intersection of affine and separable constraints (i.e. the Cartesian product of possibly nonconvex real sets). This problem class contains many NP-hard problems such as mixed-integer quadratic programming. Our heuristic is based on a variation of the alternating direction method of multipliers (ADMM), an algorithm for solving convex optimisation problems. We discuss the favourable computational aspects of our algorithm, which allow it to run quickly even on very modest computational platforms such as embedded processors. We give several examples for which an approximate solution should be found very quickly, such as management of a hybrid-electric vehicle drivetrain and control of switched-mode power converters. Our numerical experiments suggest that our method is very effective in finding a feasible point with small objective value; indeed, we see that in many cases, it finds the global solution.

ARTICLE HISTORY
Received 8 April 2016
Accepted 2 April 2017

KEYWORDS
Mixed-integer quadratic programming; heuristic; ADMM; embedded applications

1. Introduction

1.1 The problem

We consider the problem

$$\begin{array}{ll} \text{minimise} & (1/2)x^T Px + q^T x + r \\ \text{subject to} & Ax = b \\ & x \in \mathcal{X} \end{array} \tag{1}$$

with decision variable $x \in \mathbf{R}^n$. The problem parameters are the symmetric positive semidefinite matrix $P \in \mathbf{R}^{n\times n}$, the matrix $A \in \mathbf{R}^{m\times n}$, the vectors $b \in \mathbf{R}^m$ and $q \in \mathbf{R}^n$, and the real number $r \in \mathbf{R}$. The constraint set $\mathcal{X}$ is the Cartesian product of (possibly nonconvex) real, closed, nonempty sets, i.e. $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, where $\mathcal{X}_i \subseteq \mathbf{R}$ are closed, nonempty subsets of $\mathbf{R}$ for $i = 1, \ldots, n$. If $\mathcal{X}_i$ is a convex set, we refer to variable x_i as a *convex variable*, and if $\mathcal{X}_i$ is a nonconvex set, we call variable x_i a *nonconvex variable*.

Many problems can be put into the form of problem (1). For example, if some of the sets $\mathcal{X}_i$ are subsets of integers, our formulation addresses mixed-integer quadratic and mixed-integer linear programs. This includes applications such as admission control (Oulai, Chamberland, & Pierre, 2007), economic dispatch (Papageorgiou & Fraga, 2007), scheduling (Catalão, Pousinho, & Mendes, 2010), hybrid vehicle control (Murgovski, Johannesson, Sjöberg, & Egardt, 2012), thermal unit commitment problems (Carrión & Arroyo, 2006), Boolean satisfiability problems (Jeroslow & Wang, 1990), and hybrid model predictive control (Bemporad & Morari, 1999). Another application is embedded signal decoding in communication systems, when the nonconvex sets are signal constellations (e.g. QAM constellations; see Glover & Grant, 2010, p. 416).

If $\mathcal{X}$ is a convex set, problem (1) is a convex optimisation problem and can be readily solved using standard convex optimisation techniques. Otherwise, problem (1) can be hard in general. It trivially generalises mixed-integer quadratic programming, an NP-complete problem, and can therefore be used to encode other NP-complete problems such as the travelling salesman problem (Papadimitriou & Steiglitz, 1998), Boolean satisfiability (Karp, 1972; Li, Zhou, & Du, 2004), set cover (Hochbaum, 1982), and set packing (Padberg, 1973). Hence, any algorithm that guarantees finding the global solution to (1) suffers from non-polynomial worst-case time complexity (unless P = NP).

1.2 Solve techniques

There are a variety of methods for solving (1) exactly. When all of the nonconvex sets $\mathcal{X}_i$ in (1) are finite, the simplest method is brute force; enumerating through all possible combinations of discrete variables, solving a convex optimisation problem for each possible combination,

CONTACT Reza Takapoui takapoui@stanford.edu

and finding the point with the smallest objective value. Other methods such as branch-and-bound (Lawler & Wood, 1966) and branch-and-cut (Stubbs & Mehrotra, 1999) are guaranteed to find the global solution. Cutting plane methods (Chvátal, Cook, & Hartmann, 1989; Gomory et al., 1958) rely on solving the relaxation and adding a linear constraint to drive the solution towards being integer. Special purpose methods have been introduced for some specific subclasses of (1). Unfortunately, these methods have non-polynomial worst-case runtime, and are often burdensome to use in practice, especially for embedded optimisation, where runtime, memory limits, and code simplicity are prioritised. Also, these methods suffer from a large variance in the algorithm runtime.

On the other hand, many heuristics have been introduced that can deliver a good, but suboptimal (and possibly infeasible) point in a very short amount of time. For example, the *relax-and-round* heuristic consists of replacing each $\mathcal{X}_i$ by its convex hull, solving the resulting relaxation (a convex quadratic program), and projecting the solution onto the nonconvex constraint sets. Another heuristic is to fix the nonconvex variables to several reasonable guess values and solve the convex optimisation problem for convex variables. (Each of these methods may not find a feasible point, even if one exists.) The *feasibility pump* is a heuristic to find a feasible solution to a generic mixed-integer program and is discussed in Achterberg and Berthold (2007), Fischetti, Glover, and Lodi (2005), Bertacco, Fischetti, and Lodi (2007). Such heuristics are often quite effective, and can be implemented on very modest computational hardware, making them very attractive for embedded applications (even without any theoretical guarantees).

1.3 Embedded applications

We focus on embedded applications where finding a feasible point with relatively small objective value will often result in performance that is practically indistinguishable from implementing the global solution. In embedded applications, the computational resources are limited and a solution must be found in a small time. Hence, methods to find the global solution are not favourable, because their large variance in runtime cannot be tolerated.

In an embedded application, it is often required to solve several instances of (1), with different values of the parameters. Here we distinguish two separate use cases, depending on whether one or both of P or A change. This distinction will play an important role in solution methods. In the first use case, we solve many instances of (1) in which any of the parameters may change between instances. In the second use case, we solve instances of (1) in which q, b, and $\mathcal{X}$ change between instances, but P and A are constant. Although this is more restrictive than the first use case, many applications can be well modelled using this approach, including linear, time-invariant model predictive control and moving horizon estimation. Indeed, all of the three examples we present in Section 3 are of this type.

1.4 Contributions

Our proposed algorithm is a simple and computationally efficient heuristic to find approximate solutions to problem (1) quickly. It is based on the alternating direction method of multipliers (ADMM), an algorithm for solving convex optimisation problems. Because the problem class we address includes nonconvex optimisation problems, our method is not guaranteed to find the global solution, or even converge.

Numerical experiments suggest that this heuristic is an effective tool to find the global solution in a variety of problem instances. Even if our method does not find the global solution, it usually finds a feasible point with reasonable objective value. This makes it effective for many embedded optimisation applications, where finding a feasible point with relatively small objective value often results in performance that is practically indistinguishable from implementing the global solution. An implementation of our algorithm along with numerical examples is available at www.github.com/cvxgrp/miqp_admm.

Comparison of the runtime with commercial solvers such as MOSEK (ApS, 2015) and CPLEX (CPLEX, 2009) shows that our method can be substantially faster than solving a global optimisation method, while having a competitive practical performance.

1.5 Related work

In recent years, much research has been devoted to solving moderately sized convex optimisation problems quickly (i.e. in milliseconds or microseconds), possibly on embedded platforms. Examples include the SOCP solvers ECOS (Domahidi, Chu, & Boyd, 2013), and FiordOs (Ullmann, 2011), and the QP solver CVXGEN (Mattingley & Boyd, 2012). Other algorithms have been developed exclusively for convex optimal control problems (see Diehl, Bock, & Schlöder, 2005; Ferreau, Kirches, Potschka, Bock, & Diehl, 2014; O'Donoghue, Stathopoulos, & Boyd, 2013; Wang & Boyd, 2010). In addition, recent advances in automatic code generation for convex optimisation (Chu, Parikh, Domahidi, & Boyd, 2013; Mattingley, Wang, & Boyd, 2011) can significantly reduce the cost and complexity of using an embedded solver. Some recent effort has been devoted to (globally) solving

mixed-integer convex programs very quickly(see Bemporad, 2015, Frick, Domahidi, & Morari, 2015, and references therein).

Even though ADMM was originally introduced as a tool for convex optimisation problems, it turns out to be a powerful heuristic method even for NP-hard nonconvex problems (Boyd, Parikh, Chu, Peleato, & Eckstein, 2011, Sections 5 and 9). ADMM has been studied extensively in the 80s (Bertsekas & Eckstein, 1988; Gabay & Mercier, 1976; Glowinski & Marroco, 1975). More recently, it has found applications in a variety of distributed settings in machine learning such as model fitting, resource allocation, and classification (see e.g. Aybat, Zarmehri, & Kumara, 2015; Mota, Xavier, Aguiar, & Püschel, 2011; Schizas, Ribeiro, & Giannakis, 2008; Sedghi, Anandkumar, & Jonckheere, 2014; Wahlberg, Boyd, Annergren, & Wang, 2012; Wang, Lu, & Yang, 2013; Zhang & Kwok, 2014). Recently, this tool has been used as a heuristic to find approximate solutions to nonconvex problems (Chartrand & Wohlberg, 2013; Chartrand, 2012; Fält & Jimbergsson, 2015; Makela, Warrington, Morari, & Andersson, 2014). Derbinsky, Bento, Elser, and Yedidia (2013) study the *Divide and Concur* algorithm as a special case of a message-passing version of the ADMM, and introduce a three-weight version of this algorithm which greatly improves the performance for some nonconvex problems such as circle packing and the Sudoku puzzle. Consensus ADMM has been used for general quadratically constrained quadratic programming in Huang and Sidiropoulos (2016). In Xu, Yin, Wen, and Zhang (2012), ADMM has been applied to nonnegative matrix factorisation with missing values. ADMM also has been used for real and complex polynomial optimisation models in Jiang, Ma, and Zhang (2014), for constrained tensor factorisation in Liavas and Sidiropoulos (2015), and for optimal power flow in Erseghe (2014). There is a long history of using the method of multipliers to (attempt to) solve nonconvex problems (Chartrand, 2012; Chartrand & Wohlberg, 2013; Hong, 2014; Hong, Luo, & Razaviyayn, 2015; Li & Pong, 2015; Peng, Chen, & Zhu, 2015; Wang, Xu, & Xu, 2014).

2. Our heuristic

2.1 Algorithm

Our proposed algorithm is an extension of the ADMM for constrained optimisation to the nonconvex setting (Boyd et al., 2011, Sections 5,9). ADMM was originally introduced for solving convex problems, but practical evidence suggests that it can be an effective method to approximately solve some nonconvex problems as well. In order to use ADMM, we rewrite problem (1) as

$$\begin{array}{ll}\text{minimise} & (1/2)x^TPx + q^Tx + r + I_{\mathcal{X}}(z) \\ \text{subject to} & \begin{bmatrix} A \\ I \end{bmatrix} x - \begin{bmatrix} 0 \\ I \end{bmatrix} z = \begin{bmatrix} b \\ 0 \end{bmatrix}. \end{array} \tag{2}$$

Here, $I_{\mathcal{X}}$ denotes the indicator function of $\mathcal{X}$, so that $I_{\mathcal{X}}(x) = 0$ for $x \in \mathcal{X}$ and $I_{\mathcal{X}}(x) = \infty$ for $x \notin \mathcal{X}$. Each iteration in the algorithm consists of the following three steps:

$$x^{k+1/2} := \operatorname*{argmin}_x \Bigg((1/2)x^TPx + q^Tx + r + \tag{3}$$

$$(\rho/2) \left\| \begin{bmatrix} A \\ I \end{bmatrix} x - \begin{bmatrix} 0 \\ I \end{bmatrix} x^k - \begin{bmatrix} b \\ 0 \end{bmatrix} + u^k \right\|_2^2 \Bigg) \tag{4}$$

$$x^{k+1} := \Pi\left(x^{k+1/2} + \begin{bmatrix} 0 & I \end{bmatrix} u^k\right) \tag{5}$$

$$u^{k+1} := u^k + \begin{bmatrix} A \\ I \end{bmatrix} x^{k+1/2} - \begin{bmatrix} 0 \\ I \end{bmatrix} x^k - \begin{bmatrix} b \\ 0 \end{bmatrix}. \tag{6}$$

Here, Π denotes the projection onto $\mathcal{X}$, vector $u \in \mathbf{R}^{m+n}$ is the dual variable, and $\rho \in \mathbf{R}$ is a scalar parameter. (We will discuss parameter selection later in this paper.) Note that if $\mathcal{X}$ is not convex, the projection onto $\mathcal{X}$ may not be unique; for our purposes, we only need that $\Pi(z) \in \operatorname{argmin}_{x\in\mathcal{X}} \|x - z\|_2$ for all $z \in \mathbf{R}^n$. Since $\mathcal{X}$ is the Cartesian product of subsets of the real line, i.e. $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, we can take $\Pi(z) = (\Pi_1(z_1), \ldots, \Pi_n(z_n))$, where Π_i is a projection function onto $\mathcal{X}_i$. Usually evaluating $\Pi_i(z)$ is inexpensive; for example, if $\mathcal{X}_i = [\alpha, \beta]$ is an interval, $\Pi_i(z) = \min\{\max\{z, \alpha\}, \beta\}$. If $\mathcal{X}_i$ is the set of integers, Π_i rounds its argument to the nearest integer. For any finite set $\mathcal{X}_i$ with k elements, $\Pi_i(z)$ is a closest point to z that belongs to $\mathcal{X}_i$, which can be found by $\lceil \log_2 k \rceil$ comparisons.

2.2 Convergence

If the set $\mathcal{X}$ is convex and problem (1) is feasible, the algorithm is guaranteed to converge to an optimal point (Boyd et al., 2011, §3). However, for $\mathcal{X}$ nonconvex, there is no such guarantee. Indeed, because problem (1) can be NP-hard, any algorithm that finds the global solution suffers from nonpolynomial worst-case runtime. Our approach is to give up the accuracy and use methods that find an approximate solution in a small time. (Note that although the points x^{k+1} are not necessarily feasible, they are always in the set $\mathcal{X}$, which may be sufficient for some applications.)

Our numerical results verify that even for simple examples, the algorithm may fail to converge, converge

to a suboptimal point, or fail to find a feasible point, even if one exists. Since the objective value need not decrease monotonically (or at all), it is critical to keep track of the best point found. That is, for a selected primal feasibility tolerance ϵ^{tol}, we shall reject all points x such that $\|Ax - b\| > \epsilon^{\text{tol}}$, and among those primal feasible points x that $\|Ax - b\| \leq \epsilon^{\text{tol}}$, we choose the point with the smallest objective value. Here, ϵ^{tol} is a tolerance for accepted feasibility. We should remind the reader again that this point need not be the global minimum.

2.3 Initialisation

To initialise x^0, one can randomly choose a point in $\mathbf{Co}\,\mathcal{X}$, where $\mathbf{Co}\,\mathcal{X}$ denotes the convex hull of $\mathcal{X}$. More specifically, this means that we need to have access to a subroutine that generates random points in $\mathbf{Co}\,\mathcal{X}$. Our numerical results show that running the algorithm multiple times with different random initialisations increases the chance of finding a feasible point with smaller objective value. Hence, we suggest running the algorithm multiple times initialised with random starting points and report the best point as the approximate solution. We always initialise $u^0 = 0$.

2.4 Computational cost

In this subsection, we make a few comments about the computational cost of each iteration. The first step involves minimising a strongly convex quadratic function and is actually a linear operator. The point $x^{k+1/2}$ can be found by solving the following system of equations:

$$\begin{bmatrix} P + \rho I & A^T \\ A & -(1/\rho)I \end{bmatrix} \begin{bmatrix} x^{k+1/2} \\ v \end{bmatrix} = \begin{bmatrix} q' \\ 0 \end{bmatrix},$$

where $q' = -q + \rho\left(x^k + A^T b - \begin{bmatrix} A^T & I \end{bmatrix} u^k\right)$. Since the matrix on the left-hand side remains constant for all iterations, we can precompute the LDL^T factorisation of this matrix once and cache the factorisation for use in subsequent iterations. When P and A are dense, the factorisation cost is $O(n^3)$, yet each subsequent iteration costs only $O(n^2)$. (Both factorisation and solve costs can be significantly smaller if P or A is sparse.) Amortising the factorisation step over all iterations means that the first step is quite efficient. Also notice that the matrix on the left-hand side is quasi-definite and hence suitable for LDL^T factorisation.

In many applications, P and A do not change across problem instances. In this case, for different problem instances, we solve (1) for the same P and A and varying b and q. This lets us use the same LDL^T factorisation, which results in a significant saving in computation.

The second step involves projection onto $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$ and can typically be done much more quickly than the first step. It can be done in parallel since the projection onto $\mathcal{X}$ can be found by projections onto $\mathcal{X}_i$ for $i = 1, \ldots, n$. The third step is simply a dual update and is computationally inexpensive.

2.5 Preconditioning

Both theoretical analysis and practical evidence suggest that the precision and convergence rate of first-order methods can be significantly improved by preconditioning the problem. Here, we use diagonal scaling as preconditioning as discussed in Beck (2014) and Wright and Nocedal (1999). Diagonal scaling can be viewed as applying an appropriate linear transformation before running the algorithm. When the set $\mathcal{X}$ is convex, the preconditioning can substantially affect the speed of convergence, but does not affect the quality of the point returned (which must be a solution to the convex problem). In other words, for convex problems, preconditioning is simply a tool to help the algorithm converge faster. Optimal choice of preconditioners, even in the convex case, is still an active research area (Boley, 2013; Deng & Yin, 2012; Giselsson, 2014; Giselsson & Boyd, 2014a, 2014b, 2014c; Ghadimi, Teixeira, Shames, & Johansson, 2015; Hong & Luo, 2012; Shi, Ling, Yuan, Wu, & Yin, 2014). In the nonconvex case, however, preconditioning can have a critical role in the *quality* of approximate solution, as well as the speed at which this solution is found.

Specifically, let $F \in \mathbf{R}^{n\times n}$, $E \in \mathbf{R}^{m\times m}$ be diagonal matrices with positive diagonal entries. The goal is to choose F and E to improve the convergence of ADMM on the preconditioned problem

$$\begin{array}{ll} \text{minimise} & (1/2)x^T P x + q^T x + I_{\mathcal{X}}(z) \\ \text{subject to} & \begin{bmatrix} EA \\ F \end{bmatrix} x - \begin{bmatrix} 0 \\ F \end{bmatrix} z = \begin{bmatrix} Eb \\ 0 \end{bmatrix}. \end{array} \tag{7}$$

We use the choice of E and F recommended in Giselsson and Boyd (2014a) to minimise the effective condition number (the ratio of the largest singular value to the smallest non-zero singular value) of the following matrix:

$$\begin{bmatrix} E & 0 \\ 0 & F \end{bmatrix} \begin{bmatrix} A \\ I \end{bmatrix} P^{\dagger} \begin{bmatrix} A^T & I \end{bmatrix} \begin{bmatrix} E & 0 \\ 0 & F \end{bmatrix},$$

where $P^{\dagger}$ denotes the pseudo-inverse of P. Given matrix $M \in \mathbf{R}^{n\times n}$, minimising the condition number of DMD for diagonal $D \in \mathbf{R}^{n\times n}$ can be cast as a semidefinite program. However, a heuristic called *matrix equilibration*

can be used to avoid the computational cost of solving a semidefinite program (see Bradley, 2010; Sluis, 1969, and references therein). Since for embedded applications computational resources are limited, we avoid finding $P^\dagger$ or equilibrating completely. We instead find E to normalise the rows of A (usually in ℓ_1 or ℓ_2 norm) and set F to be the identity.

After finding E and F, preconditioned ADMM has the following form:

$$\begin{aligned} x^{k+1/2} &:= \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} P+\rho F^2 & A^TE \\ EA & -(1/\rho)I \end{bmatrix}^{-1} \\ &\quad \begin{bmatrix} -q+\rho\left(F^2x^k + A^TE^2b - \begin{bmatrix} A^TE & F \end{bmatrix} u^k\right) \\ 0 \end{bmatrix} \\ x^{k+1} &:= \Pi\left(x^{k+1/2} + \begin{bmatrix} 0 & F^{-1} \end{bmatrix} u^k\right) \\ u^{k+1} &:= u^k + \begin{bmatrix} EA \\ F \end{bmatrix} x^{k+1/2} - \begin{bmatrix} 0 \\ F \end{bmatrix} x^k - \begin{bmatrix} Eb \\ 0 \end{bmatrix}. \end{aligned} \tag{8}$$

2.6 The overall algorithm

We use the update rules (8) for $k = 1, \ldots, N$, where N denotes the (fixed) maximum number of iterations. Also, as described above, the algorithm is repeated for M number of random initialisations. The computational cost of the algorithm consists of a factorisation and MN matrix products and projections. A description of the overall algorithm is given in Algorithm 1, with $f(x) = (1/2)x^TPx + q^Tx + r$.

Algorithm 1 Approximately solving nonconvex constrained QP (1)

if A or P changed, **then**
  find E and F by equilibrating $\begin{bmatrix} A \\ I \end{bmatrix} P^\dagger \begin{bmatrix} A^T & I \end{bmatrix}$
  find LDL^T factorizsation of $\begin{bmatrix} P+\rho F^2 & A^TE \\ EA & -(1/\rho)I \end{bmatrix}$
end if
$x_{\text{best}} := \emptyset$, $f(x_{\text{best}}) := \infty$
for random initializsation $1, 2, \ldots, N$ **do**
  for iteration $1, 2, \ldots, M$ **do**
    update x from (8)
    if $\|Ax - b\|_2 \le \epsilon^{\text{tol}}$ and $f(x) < f(x_{\text{best}})$, **then**
      $x_{\text{best}} = x$
    end if
  end for
end for
return x_{best}.

We mention a solution refinement technique here that can be used to find a solution with possibly better objective value after the algorithm stops. This technique, sometimes known as *polishing*, consists of fixing the nonconvex variable and solving the resulting convex optimisation problem. Using this technique, one may use larger ϵ^{tol} during the N iterations and only reduce ϵ^{tol} at the refinement step. Depending on the application, it might be computationally sensible to solve the resulting convex optimisation problem. Another effective technique is to introduce a notion of *no-good cut* during iterations for problems with binary variables. A no-good cut forces the vector of binary variables to change over iterations, by appending the linear equality constraint $\sum_{i\in T} x_i - \sum_{i\in F} x_i \le B - 1$, to the minimisation in the first step of (6), where we have $T = \{i \mid x^k_{b_i} = 1\}$ (i.e. T is the set of binary variables for which the last iterate was 1), $F = \{i \mid x^k_{b_i} = 0\}$ (i.e. F is the set of binary variables for which the last iterate was 0), and B is the number of elements of T. We do not use either of these techniques in the following examples.

3. Numerical examples

In this section, we explore the performance of our proposed algorithm on some example problems. For each example, ρ was chosen between 0.1 and 10 to yield good performance; all other algorithm parameters were kept constant. As a benchmark, we compare our results to the commercial solver MOSEK, which can globally solve MIQPs. All experiments were carried out on a system with two 3.06 GHz cores with 4 GB of RAM.

The results suggest that this heuristic is effective in finding approximate solutions for mixed integer quadratic programs.

3.1 Randomly generated QP

First, we demonstrate the performance of our algorithm qualitatively for a random mixed-Boolean quadratic program. The matrix P in (1) was chosen as $P = QQ^T$, where the entries of $Q \in \mathbf{R}^{n\times n}$, as well as those of q and A, were drawn from a standard normal distribution. The constant r was chosen such that the optimal value of the unconstrained quadratic minimisation is 0. The vector b was chosen as $b = Ax_0$, where $x_0 \in \mathcal{X}$ was chosen uniformly randomly, thus ensuring that the problem is feasible. We used $n = 200$ and $m = 50$ with $\mathcal{X}_i = \{0, 1\}$ for $i = 1, \ldots, 100$, $\mathcal{X}_i = \mathbf{R}_+$ for $i = 101, \ldots, 150$, and $\mathcal{X}_i = \mathbf{R}$ for the other indices i.

We used MOSEK to find the optimal value for the problem. After more than 16 hours, MOSEK certifies that the optimal value is equal to 2040. We ran algorithm 1

for 10 different initialisations and 200 iterations for each initialisation, with step size $\rho = 0.5$. For a naive implementation in MATLAB, it took 120 milliseconds to complete all precomputations (preconditioning and factorisation), and 800 milliseconds to do all 2000 iterations. The best objective value found for the problem was 2067 (1.3% suboptimal). Our implementation in C enables us to solve sparse problems significantly faster.

One interesting observation is that the parameter ρ tends to trade off feasibility and optimality: with small values of ρ, the algorithm often fails to find a feasible point, but feasible points found tend to have low objective value. On the other hand, with large values of ρ, feasible points are found more quickly, but tend to have higher objective value.

3.2 Hybrid vehicle control

We consider a simple hybrid electric vehicle drivetrain (similar to that of Boyd & Vandenberghe, 2004, Exercise 4.65), which consists of a battery, an electric motor/generator, and a heat engine, in a parallel configuration. Control of a hybrid vehicle appears as an embedded practice in application (Chakraborty et al., 2012; Muta, Yamazaki, & Tokieda, 2004; Wenzhong, Mi, & Emadi, 2007). We assume that the demanded power P_t^{des} at the times $t = 0, \dots, T-1$ is known in advance. Our task is to plan out the battery and engine power outputs P_t^{batt} and P_t^{eng}, for $t = 0, \dots, T-1$, so that

$$P_t^{\text{batt}} + P_t^{\text{eng}} \geq P_t^{\text{des}}.$$

(Strict inequality above corresponds to braking.)

The battery has stored energy E_t at time t, which evolves according to

$$E_{t+1} = E_t - \tau P_t^{\text{batt}}, \qquad t = 0, \dots, T-1,$$

where τ is the length of each discretised time interval. The battery capacity is limited, so that $0 \leq E_t \leq E^{\max}$ for all t, and the initial energy E_0 is known. We penalise the terminal energy state of the battery according to $g(E_T)$, where

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

for $\eta \geq 0$.

At time t, the engine may be on or off, which is modelled with binary variable z_t. If the engine is on ($z_t = 1$), then we have $0 \leq P_t^{\text{eng}} \leq P^{\max}$, and $\alpha(P_t^{\text{eng}})^2 + \beta P_t^{\text{eng}} + \gamma$ units of fuel are consumed, for nonnegative constants α, β, and γ. If the engine is off ($z_t = 0$), it consumes no fuel, and $P_t^{\text{eng}} = 0$. Because $z_t \in \{0, 1\}$, the power constraint can be written as $0 \leq P^{\text{eng}} \leq P^{\max} z_t$, and the fuel cost as $f(P_t^{\text{eng}}, z_t)$, where

$$f(P, z) = \alpha P^2 + \beta P + \gamma z.$$

Additionally, we assume that turning the engine on after it has been off incurs a cost $\delta \geq 0$, i.e. at each time t, we pay $\delta(z_t - z_{t-1})_+$, where $(\cdot)_+$ denotes the positive part.

The hybrid vehicle control problem can be formulated as

$$\begin{array}{ll} \text{minimise} & \eta(E_T - E^{\max})^2 + \sum_{t=0}^{T-1} f(P_t^{\text{eng}}, z_t) \\ & \quad + \delta(z_t - z_{t-1})_+ \\ \text{subject to} & E_{t+1} = E_t - \tau P_t^{\text{batt}} \\ & P_t^{\text{batt}} + P_t^{\text{eng}} \geq P_t^{\text{des}} \\ & z_t \in \{0, 1\}, \end{array} \tag{9}$$

where all constraints must hold for $t = 0, \dots, T-1$. The variables are P_t^{batt}, P_t^{eng}, and z_t for $t = 0, \dots, T-1$, and E_t, for $t = 1, \dots, T$. In addition to the parameters given above, we take z_{-1} to be a parameter denoting the initial engine state.

We used the parameter values $\alpha = 1$, $\beta = 10$, $\gamma = 1.5$, $\delta = 10$, $\eta = 0.1$, $\tau = 5$, $P^{\max} = 1$, $E^{\max} = 200$, $E_0 = 200$, and $z_{-1} = 0$. The demanded power trajectory P_t^{des} is not shown, but can be obtained by summing the engine power and battery power in Figure 1. We ran the algorithm with $\rho = 0.4$ for 900 iterations, with primal optimality threshold $\epsilon^{\text{tol}} = 10^{-4}$. The global solution found by MOSEK generates an objective value of 139.52 and the best objective value with our algorithm was 140.07. In Figure 1, we see that qualitatively, the optimal trajectory and the trajectory generated by ADMM are very similar. Our implementation in C carries out precomputations in 27 milliseconds. The total time for all 900 iterations is 63 milliseconds, which gives each iteration an average time of 70 microseconds. MOSEK finds the first feasible point after 1 second, and it takes about 15 seconds to find a point with the same quality as found with our heuristic.

3.3 Power converter control

We discuss the control of an embedded switched-mode power converter control (Buso & Mattavelli, 2006; Camara, Gualous, Gustin, Berthon, & Dakyo, 2010; Smedley & Cuk, 1995). We consider control of the switched-mode power converter shown in Figure 2. The circuit dynamics are

$$\xi_{t+1} = G\xi_t + Hu_t, \qquad t = 0, 1, \dots, T-1,$$

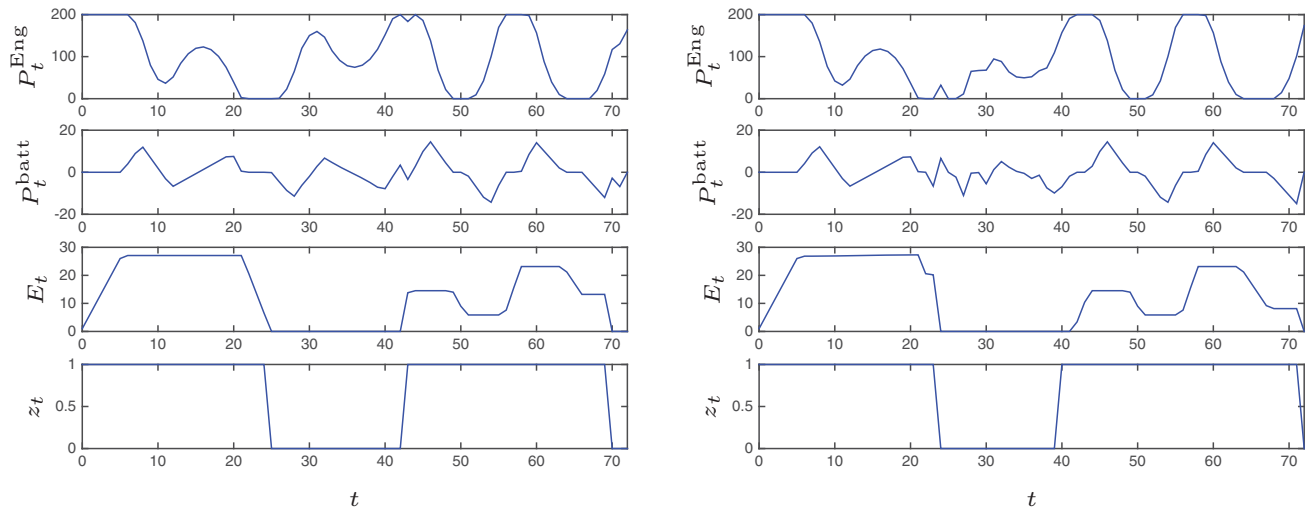

Figure 1. Engine power, battery power, battery energy, and engine on/off signals versus time. Left: the global solution. Right: the solution found using ADMM (Algorithm 1).

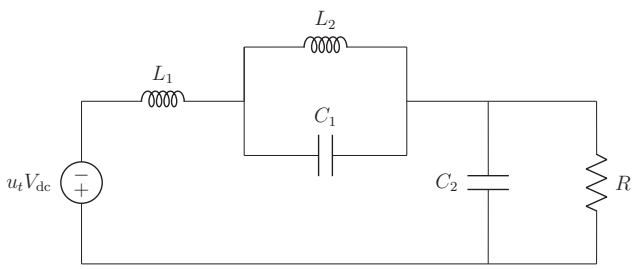

Figure 2. Converter circuit model.

where $\xi_t = (i_{1,t}, v_{1,t}, i_{2,t}, v_{2,t})$ is the system state at epoch t, consisting of all inductor currents and capacitor voltages, and $u_t \in \{-1, 0, 1\}$ is the control input. The dynamics matrices $G \in \mathbf{R}^{4\times 4}$ and $H \in \mathbf{R}^{4\times 1}$ are obtained by discretising the dynamics of the circuit in Figure 2.

We would like to control the switch configurations so that v_2 tracks a desired sinusoidal waveform. This can be done by solving

$$\begin{array}{ll} \text{minimise} & \sum_{t=0}^{T}(v_{2,t} - v_{\mathrm{des}})^2 + \lambda|u_t - u_{t-1}| \\ \text{subject to} & \xi_{t+1} = G\xi_t + Hu_t \\ & \xi_0 = \xi_T \\ & u_0 = u_T \\ & u_t \in \{-1, 0, 1\}, \end{array} \tag{10}$$

where $\lambda \geq 0$ is a tradeoff parameter between output voltage regulation and switching frequency. The variables are ξ_t for $t = 0, \ldots, T$ and u_t for $t = 0, \ldots, T-1$.

Note that if we take $\lambda = 0$, and take the input voltage u_t to be unconstrained (i.e. allow u_t to take any values in $\mathbf{R}$), (10) can be solved as a convex quadratic minimisation problem, with solution ξ_t^{ls}. Returning to our original problem, we can penalise deviation from this ideal waveform by including a regularisation term $\mu\|\xi - \xi_t^{\mathrm{ls}}\|^2$ to (10), where $\mu > 0$ is a positive weighting parameter. We solved this regularised version of (10), with $L_1 = 10\ \mu\mathrm{H}$, $C_1 = 1\ \mu\mathrm{F}$, $L_2 = 10\ \mu\mathrm{H}$, $C_2 = 10\ \mu\mathrm{F}$, $R = 1\Omega$, $V_{\mathrm{dc}} = 10\ \mathrm{V}$, $T = 100$ (with a discretisation interval of $0.5\ \mu\mathrm{s}$), $\lambda = 1.5\mathrm{V}^2$, and $\mu = 0.1$. We run algorithm 1 with $\rho = 2.7$ and 500 iterations for three different initialisations. It takes less than 20 milliseconds for our implementation to carry out all precomputations, and it takes about 150 milliseconds for all iterations (with an average time of 100 microseconds per iteration). An approximate solution is found via our heuristic in less than 170 milliseconds, whereas it takes for MOSEK more than 4 hours to find the global solution. Figure 3 compares the approximate solution derived by the heuristic with the global solution.

3.4 Signal decoding

We consider maximum-likelihood decoding of a message passed through a linear multiple-input and multiple-output (MIMO) channel (Damen, Chkeif, & Belfiore, 2000; Sinnokrot, Barry, & Madisetti, 2008; Viterbo & Boutros, 1999). In particular, we have

$$y = Hx + v,$$

where $y \in \mathbf{R}^p$ is the message received, $H \in \mathbf{R}^{p\times n}$ is the channel matrix, $x \in \mathbf{R}^n$ is the message sent, and the elements of the noise vector $v \in \mathbf{R}^p$ are independent,

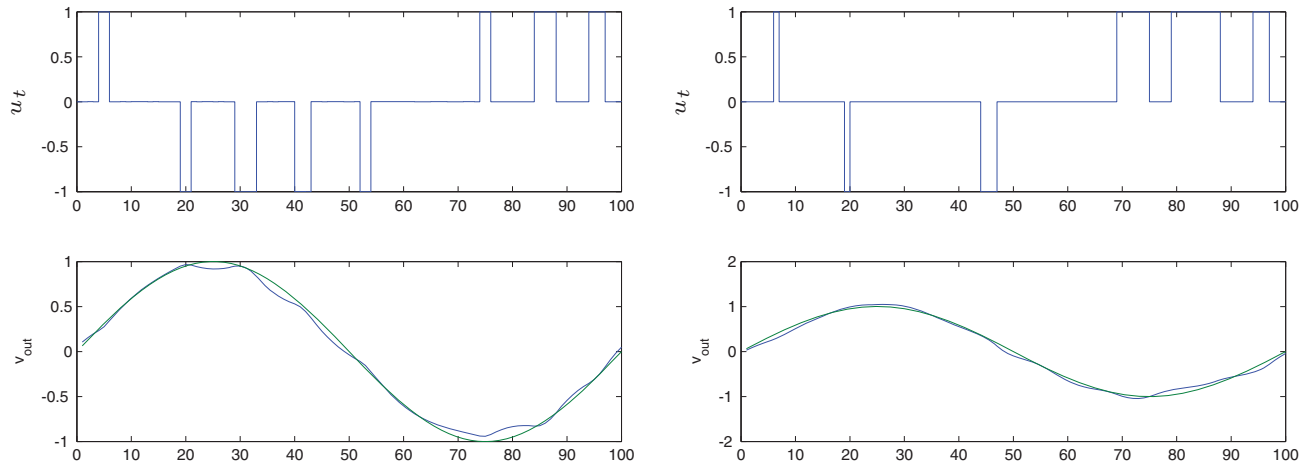

Figure 3. The switch configuration and the output voltage. Left: the global solution. Right: the solution using ADMM (Algorithm 1).

identically distributed Gaussian random variables. We further assume that the elements of x belong to the *signal constellation* $\{-3, -1, 1, 3\}$. The maximum likelihood estimate of x is given by the solution to the problem

$$\begin{aligned} &\text{minimise } \|H\hat{x} - y\|^2 \\ &\text{subject to } \hat{x}_i \in \{-3, -1, 1, 3\}, \quad i = 1, \ldots, n, \end{aligned} \tag{11}$$

where $\hat{x} \in \mathbf{R}^n$ is the variable.

We generate 1000 random problem instances with $H \in \mathbf{R}^{2000\times 400}$ chosen from a standard normal distribution. The uncorrupted signal x is chosen uniformly randomly and the additive noise is Gaussian such that the signal-to-noise ratio (SNR) is 8 dB. For such a problem in embedded application, branch-and-bound methods are not desirable due to their worst-case time complexity. We run the heuristic with only one initialisation, with 10 iterations to find x^{admm}. The average runtime for each problem (including preprocessing) is 80 milliseconds, which is substantially faster than branch-and-bound based methods. We compare the performance of the points x^{admm} with the points found by relax-and-round technique x^{rlx}. In Figure 4, we have plotted the histogram of the difference between the objective values evaluated at x^{admm} and x^{rlx}. Depicted in Figure 4, we see that in 95% of the cases, the bit error rate (BER) using our heuristic was at least as good as the BER using relax-and-round.

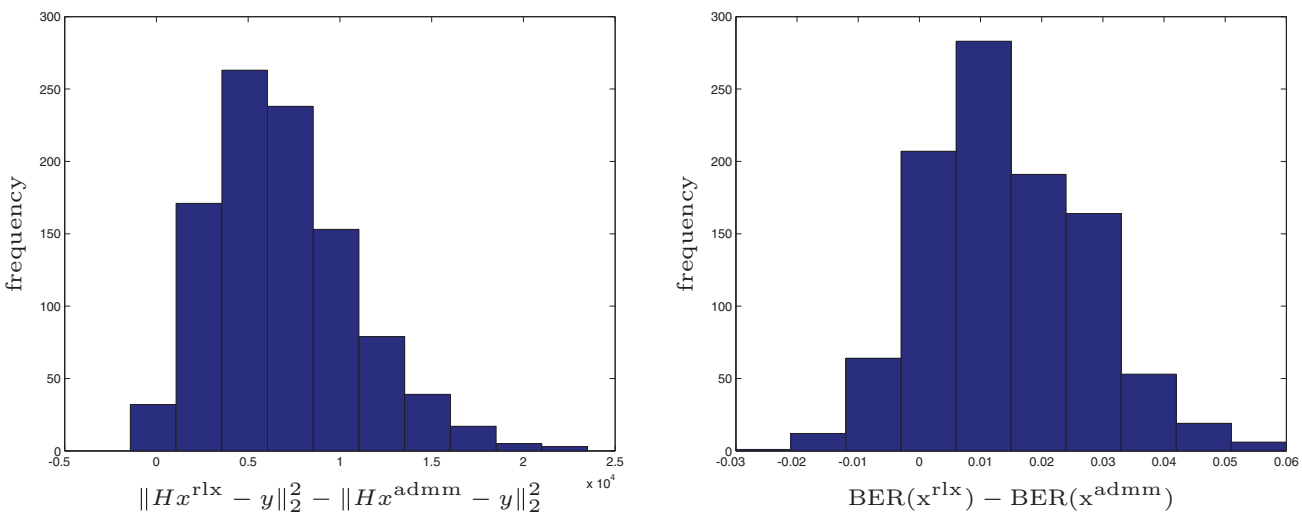

Figure 4. Comparison of ADMM heuristic and relax-and-round. Left: The difference in objective values. Right: The difference in bit error rates.

4. Conclusions

In this paper, we introduced an effective heuristic for finding approximate solutions to convex quadratic minimisation problems over the intersection of affine and nonconvex sets. Our heuristic is significantly faster than branch-and-bound algorithms and has shown effective in a variety of embedded problems including hybrid vehicle control, power converter control, and signal decoding.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Achterberg, T., & Berthold, T. (2007). Improving the feasibility pump. *Discrete Optimization, 4*(1), 77–86.

Aybat, N., Zarmehri, S., & Kumara, S. (2015). An ADMM algorithm for clustering partially observed networks. In *Proceedings of the 2015 SIAM international conference on data mining*, Vancouver, Canada: SIAM.

Beasley, J.E. (1998). *Heuristic algorithms for the unconstrained binary quadratic programming problem*. London: Management School, Imperial College.

Beck, A., (2014). *Introduction to nonlinear optimization: Theory, algorithms, and applications with MATLAB* (Vol. 19). Philadelphia, PA: SIAM.

Bemporad, A. (2015). Solving mixed-integer quadratic programs via nonnegative least squares. In *5th IFAC conference on nonlinear model predictive control*, (pp. 73–79). Seville, Spain: International Federation of Automatic Control.

Bemporad, A., & Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica, 35*(3), 407–427.

Bertacco, L., Fischetti, M., & Lodi, A. (2007). A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization, 4*(1), 63–76.

Bertsekas, D.P., & Eckstein, J. (1988). Dual coordinate step methods for linear network flow problems. *Mathematical Programming, 42*(1–3), 203–243.

Bienstock, D. (1996). Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming, 74*(2), 121–140.

Boley, D. (2013). Local linear convergence of the alternating direction method of multipliers on quadratic or linear programs. *SIAM Journal on Optimization, 23*(4), 2183–2207.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*, Cambridge: Cambridge University Press.

Boyd, S., Parikh, N., Chu, E., Peleato, B., & Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning, 3*(1), 1–122.

Bradley, A.M. (2010). *Algorithms for the equilibration of matrices and their application to limited-memory quasi-newton methods* (Doctoral thesis). Stanford University, CA: Stanford University.

Buso, S., & Mattavelli, P. (2006). Digital control in power electronics. *Lectures on Power Electronics, 1*(1), 1–158.

Camara, M., Gualous, H., Gustin, F., Berthon, A., & Dakyo, B. (2010). DC/DC converter design for supercapacitor and battery power management in hybrid vehicle applications-polynomial control strategy. *IEEE Transactions on Industrial Electronics, 57*(2), 587–597.

Carrión, M., & Arroyo, J. M. (2006). A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. *IEEE Transactions on Power Systems, 21*(3), 1371–1378.

Catalão, J.P.S., Pousinho, H.M.I., & Mendes, V.M.F. (2010). Scheduling of head-dependent cascaded hydro systems: Mixed-integer quadratic programming approach. *Energy Conversion and Management, 51*(3), 524–530.

Chakraborty, S., Lukasiewycz, M., Buckl, C., Fahmy, S., Chang, N., Park, S., ... Adlkofer, H. (2012). Embedded systems and software challenges in electric vehicles. In *Proceedings of the conference on design, automation and test in Europe*, (pp. 424– 429). San Jose, CA: EDA Consortium.

Chartrand, R. (2012). Nonconvex splitting for regularized low-rank + sparse decomposition. *IEEE Transactions on Signal Processing, 60*(11), 5810–5819.

Chartrand, R., & Wohlberg, B. (2013). A nonconvex ADMM algorithm for group sparsity with sparse groups. In *Proceedings of IEEE international conference on acoustics, speech and signal processing (ICASSP)*, (pp. 6009–6013). Vancouver, Canada: IEEE.

Chu, E., Parikh, N., Domahidi, A., & Boyd, S. (2013). Code generation for embedded second-order cone programming. In *Proceedings of the 2013 European control conference*, (pp. 1547–1552). Zurich, Switzerland: IEEE Control Systems Society.

Chvátal, V., Cook, W., & Hartmann, M. (1989). On cutting-plane proofs in combinatorial optimization. *Linear Algebra and Its Applications, 114*, 455–499.

Damen, O., Chkeif, A., & Belfiore, J. (2000). Lattice code decoder for space-time codes. *IEEE Communications Letters, 4*(5), 161–163.

Deng, W., & Yin, W. (2012). On the global and linear convergence of the generalized alternating direction method of multipliers, *66. Journal of Scientific Computing,* 1–28.

Derbinsky, N., Bento, J., Elser, V., & Yedidia, J.S. (2013). An improved three-weight message-passing algorithm. *arXiv:1305.1961*.

Diehl, M., Bock, H., & Schlöder, J. (2005). A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization, 43*(5), 1714–1736.

Domahidi, A., Chu, E., & Boyd, S. (2013). ECOS: An SOCP solver for embedded systems. In *Proceedings of the 12th European control conference*, (pp. 3071–3076). Zurich, Switzerland: IEEE.

Erseghe, T. (2014). Distributed optimal power flow using ADMM. *IEEE Transactions on Power Systems, 29*(5), 2370–2380.

lt, M., , & Jimbergsson, L. (2015). Using ADMM for hybrid system MPC, ISSN 0280-5316. Retrieved from: https://lup.lub.lu.se/student-papers/publication/7445552

Ferreau, H., Kirches, C., Potschka, A., Bock, H., & Diehl, M. (2014). qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation, 6*(4), 327–363.

Fischetti, M., Glover, F., & Lodi, A. (2005). The feasibility pump. *Mathematical Programming, 104*(1), 91–104.

Frick, D., Domahidi, A., & Morari, M. (2015). Embedded optimization for mixed logical dynamical systems. *Computers and Chemical Engineering, 72*, 21–33.

Gabay, D., & Mercier, B. (1976). A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications, 2*(1), 17–40.

Ghadimi, E., Teixeira, A., Shames, I., & Johansson, M. (2015). Optimal parameter selection for the alternating direction method of multipliers (ADMM): Quadratic problems. *IEEE Transactions on Automatic Control, 60*(3), 644–658.

Giselsson, P. (2014). Improved fast dual gradient methods for embedded model predictive control. In *International federation of automatic control (IFAC)*, (pp. 2303–2309). Cape Town, South Africa: Elsevier.

Giselsson, P., & Boyd, S. (2014a). Diagonal scaling in Douglas–Rachford splitting and ADMM. In *53rd Annual IEEE conference on decision and control (CDC)*, (pp. 5033–5039). Los Angeles, CA: IEEE.

Giselsson, P., & Boyd, S. (2014b). Monotonicity and restart in fast gradient methods. In *53rd annual IEEE conference on decision and control (CDC)*, (pp. 5058–5063). Los Angeles, CA: IEEE.

Giselsson, P., & Boyd, S. (2014c). Preconditioning in fast dual gradient methods. In *53rd annual IEEE conference on decision and control (CDC)*, (pp. 5040–5045). Los Angeles, CA: IEEE.

Glover, I., & Grant, P.M. (2010). *Digital communications*. London: Pearson Education.

Glowinski, R., & Marroco, A. (1975). On the solution of a class of nonlinear Dirichlet problems by a penalty-duality method and finite elements of order one. *Revue française d'automatique, informatique, recherche opérationnelle. Analyse numérique, 9*(2), 41–76.

Gomory, R.E., . (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society, 64*(5), 275–278.

Hochbaum, D.S. (1982). Approximation algorithms for the set covering and vertex cover problems. *SIAM Journal on Computing, 11*(3), 555–556.

Hong, M. (2014). A distributed, asynchronous and incremental algorithm for nonconvex optimization: An ADMM based approach. *arXiv:1412.6058*.

Hong, M., & Luo, Z. (2012). On the linear convergence of the alternating direction method of multipliers. *arXiv:1208.3922*.

Hong, M., Luo, Z., & Razaviyayn, M. (2015). Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. In *IEEE international conference on acoustics, speech and signal processing*, (pp. 3836–3840). South Brisbane, Oueensland: IEEE.

Huang, K., & Sidiropoulos, N.D. (2016). Consensus-ADMM for general quadratically constrained quadratic programming. *arXiv:1601.02335*.

IBM ILOG CPLEX. (2009). *User's manual for CPLEX*, (Vol. 46, p. 157). North Castle, NY: International Business Machines Corporation.

Jeroslow, R.G., & Wang, J. (1990). Solving propositional satisfiability problems. *Annals of Mathematics and Artificial Intelligence, 1*(1–4), 167–187.

Jiang, B., Ma, S., & Zhang, S. (2014). Alternating direction method of multipliers for real and complex polynomial optimization models. *Optional,* 63(6), 883–898.

Karp, R.M. (1972). *Reducibility among combinatorial problems*. New York, NY: Springer.

Lawler, E.L., & Wood, D.E. (1966). Branch-and-bound methods: A survey. *Operations Research, 14*(4), 699–719.

Li, G., & Pong, T.K. (2015). Global convergence of splitting methods for nonconvex composite optimization. *SIAM Journal on Optimization, 25*(4), 2434–2460.

Li, R., Zhou, D., & Du, D. (2004). Satisfiability and integer programming as complementary tools. In *Proceedings of the 2004 Asia and South Pacific design automation conference*, (pp. 879–882). Kanagawa, Japan: IEEE.

Liavas, A.P., & Sidiropoulos, N.D. (2015). Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers. *IEEE Transactions on Signal Processing, 63*(20), 5450–5463.

Makela, O., Warrington, J., Morari, M., & Andersson, G. (2014). Optimal transmission line switching for large-scale power systems using the alternating direction method of multipliers. In *Power systems computation conference (PSCC) 2014*, (pp. 1–6). Wroclaw, Poland: IEEE.

Mattingley, J., & Boyd, S. (2012). CVXGEN: A code generator for embedded convex optimization. *Optimization and Engineering, 13*(1), 1–27.

Mattingley, J., Wang, Y., & Boyd, S. (2011). Receding horizon control: Automatic generation of high-speed solvers. *IEEE Control Systems Magazine, 31*(3), 52–65.

MOSEK ApS. (2015). *The MOSEK optimization toolbox for MATLAB manual. Version 7.1 (Revision 28)*. Retrieved from http://docs.mosek.com/7.1/toolbox/index.html

Mota, J., Xavier, J., Aguiar, P., & Püschel, M. (2011). Basis pursuit in sensor networks. In *2011 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, (pp. 2916–2919). Prague, Czech Republic: IEEE.

Murgovski, N., Johannesson, L., Sjöberg, J., & Egardt, B. (2012). Component sizing of a plug-in hybrid electric powertrain via convex optimization. *Mechatronics, 22*(1), 106–120.

Muta, K., Yamazaki, M., & Tokieda, J. (2004). Development of new-generation hybrid system THS II - drastic improvement of power performance and fuel economy. *SAE Transactions, 113*(3), 182–192.

O'Donoghue, B., Stathopoulos, G., & Boyd, S. (2013). A splitting method for optimal control. *IEEE Transactions on Control Systems Technology, 21*(6), 2432–2442.

Oulai, D., Chamberland, S., & Pierre, S. (2007). A new routing-based admission control for MPLS networks. *IEEE Communications Letters, 11*(2), 216–218.

Padberg, M.W. (1973). On the facial structure of set packing polyhedra. *Mathematical Programming, 5*(1), 199–215.

Papadimitriou, C.H., & Steiglitz, K. (1998). *Combinatorial optimization: Algorithms and complexity*, North Chelmsford, MA: Courier Corporation.

Papageorgiou, L.G., & Fraga, E.S. (2007). A mixed-integer quadratic programming formulation for the economic dispatch of generators with prohibited operating zones. *Electric Power Systems Research, 77*(10), 1292–1296.

Peng, Z., Chen, J., & Zhu, W. (2015). A proximal alternating direction method of multipliers for a minimization problem with nonconvex constraints. *Journal of Global Optimization,*, *62* 1–18.

Schizas, L., Ribeiro, A., & Giannakis, G. (2008). Consensus in ad hoc WSNs with noisy links? Part I: Distributed estimation of deterministic signals. *IEEE Transactions on Signal Processing, 56*(1), 350–364.

Sedghi, H., Anandkumar, A., & Jonckheere, E. (2014). Multi-step stochastic ADMM in high dimensions: Applications to sparse optimization and matrix decomposition. In *Advances in neural information processing systems* (pp. 2771–2779). Montreal, Canada: Advances in Neural Information Proecssing Systems 27.

Shi, W., Ling, Q., Yuan, K., Wu, G., & Yin, W. (2014). On the linear convergence of the admm in decentralized consensus optimization. *IEEE Transactions on Signal Processing, 62*(7), 1750–1761.

Sinnokrot, M., Barry, J., & Madisetti, V. (2008). Embedded Alamouti space-time codes for high rate and low decoding complexity. In *2008 42nd Asilomar conference on signals, systems and computers*, (pp. 1749–1753). Pacific Grove, CA: IEEE.

Sluis, A.V.D. (1969). Condition numbers and equilibration of matrices. *Numerische Mathematik, 14*(1), 14–23.

Smedley, K., & Cuk, S. (1995). One-cycle control of switching converters. *IEEE Transactions on Power Electronics, 10*(6), 625–633.

Stubbs, R.A., & Mehrotra, S. (1999). A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming, 86*(3), 515–532.

Ullmann, F. (2011). *FiOrdOs: A Matlab toolbox for C-code generation for first order methods* (master's thesis). ETH Zurich.

Viterbo, E., & Boutros, J. (1999). A universal lattice code decoder for fading channels. *IEEE Transactions on Information Theory, 45*(5), 1639–1642.

Wahlberg, B., Boyd, S., Annergren, M., & Wang, Y. (2012). An ADMM algorithm for a class of total variation regularized estimation problems. *IFAC Proceedings Volumes, 45*(16), 83–88.

Wang, D., Lu, H., & Yang, M. (2013). Online object tracking with sparse prototypes. *IEEE Transactions on Image Processing, 22*(1), 314–325.

Wang, F., Xu, Z., & Xu, H. (2014). Convergence of alternating direction method with multipliers for non-convex composite problems. *arXiv:1410.8625*.

Wang, Y., & Boyd, S. (2010). Fast model predictive control using online optimization. *IEEE Transactions on Control Systems Technology, 18*(2), 267–278.

Wenzhong Gao, D. Mi, C., & Emadi, A. (2007). Modeling and simulation of electric and hybrid vehicles. *Proceedings of the IEEE, 95*(4), 729–745.

Wolsey, L.A. (1998). *Integer programming* (Vol. 42). New York, NY: Wiley.

Wright, S.J., & Nocedal, J. (1999). *Numerical optimization* (Vol. 2). New York, NY: Springer.

Xu, Y., Yin, W., Wen, Z. & Zhang, Y. (2012). An alternating direction algorithm for matrix completion with nonnegative factors. *Frontiers of Mathematics in China, 7*(2), 365–384.

Zhang, R., & Kwok, J.T. (2014). Asynchronous distributed ADMM for consensus optimization. In *ICML*. (pp. 1701–1709). Beijing, China: International Conference on Machine Learning.