Computational Optimization and Applications
https://doi.org/10.1007/s10589-021-00342-y

On a primal-dual Newton proximal method for convex quadratic programs

Alberto De Marchi[1]

Received: 12 December 2020 / Accepted: 1 December 2021

Abstract
This paper introduces QPDO, a primal-dual method for convex quadratic programs which builds upon and weaves together the proximal point algorithm and a damped semismooth Newton method. The outer proximal regularization yields a numerically stable method, and we interpret the proximal operator as the unconstrained minimization of the primal-dual proximal augmented Lagrangian function. This allows the inner Newton scheme to exploit sparse symmetric linear solvers and multi-rank factorization updates. Moreover, the linear systems are always solvable independently from the problem data and exact linesearch can be performed. The proposed method can handle degenerate problems, provides a mechanism for infeasibility detection, and can exploit warm starting, while requiring only convexity. We present details of our open-source C implementation and report on numerical results against state-of-the-art solvers. QPDO proves to be a simple, robust, and efficient numerical method for convex quadratic programming.

Keywords Semismooth Newton method · Proximal point method · Regularized primal-dual method · Convex quadratic programming

1 Introduction

Quadratic programs (QPs) are one of the fundamental problems in optimization. In this paper, we consider linearly constrained convex QPs, in the form:

$$\min_{\mathbf{x}} \quad \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{q}^\top \mathbf{x}, \quad s.t. \quad \mathbf{l} \leq \mathbf{A}\mathbf{x} \leq \mathbf{u}, \tag{1}$$

with $\mathbf{x} \in \mathbb{R}^n$. $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and $\mathbf{q} \in \mathbb{R}^n$ define the objective function, whereas the constraints are encoded by $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{l}, \mathbf{u} \in \overline{\mathbb{R}}^m$. We assume (i) $\mathbf{Q}$ is symmetric

✉ Alberto De Marchi
alberto.demarchi@unibw.de

[1] Universität der Bundeswehr München, 85577 Neubiberg/Munich, Germany

Published online: 06 January 2022

positive semidefinite, *i.e.*, $\mathbf{Q} \succeq 0$, and (ii) $\mathbf{l}$ and $\mathbf{u}$ satisfy $\mathbf{l} \leq \mathbf{u}$, $\mathbf{l} < +\infty$, and $\mathbf{u} > -\infty$ component-wise; cf. [33, 59]. We will refer to the nonempty, closed and convex set

$$\mathcal{C} := \{\mathbf{z} \in \mathbb{R}^m \ : \ \mathbf{l} \leq \mathbf{z} \leq \mathbf{u}\}$$

as the constraint set. Note that (1) represents a general convex QP, in that it accomodates also equality constraints and bounds. We denote N the sum of the number of nonzero entries in $\mathbf{Q}$ and $\mathbf{A}$, *i.e.*, $N := nnz(\mathbf{Q}) + nnz(\mathbf{A})$.

1.1 Background

Optimization problems in the form (1) appear in a variety of applications and are of interest in engineering, statistics, finance and many other fields. QPs often arise as sub-problems in methods for general nonlinear programming [10, 31, 44], and greatly vary in terms of problem size and structure.

Convex QPs have been studied since the 1950s [22] and several numerical methods have been developed since then. These differ in how they balance the number of iterations and the cost (*e.g.*, run time) per iteration.

Active-set methods for QPs originated from extending the simplex method for linear programs (LPs) [64]. These methods select a set of binding constraints and iteratively adapt it, seeking the set of active constraints at the solution. Active-set algorithms can be easily warm started and can lead to finite convergence. Moreover, by adding and dropping constraints from the set of binding ones, factorization updates can be adopted for solving successive linear systems. However, these methods may require many iterations to identify the correct set of active constraints. Modern solvers based on active-set methods are qpOASES [20] and NASOQ [13].

First-order methods iteratively compute an optimal solution using only first-order information about the cost function [46, 49]. As these methods consist of computationally cheap and simple steps, they are well suited to applications with limited computing resources [59]. However, first-order algorithms usually require many iterations to achieve accurate solutions and may suffer from ill-conditioning of the problem data. Several acceleration schemes have been proposed to improve their behaviour [1, 62]. The open-source solver OSQP [59] offers an implementation based on ADMM [9].

Interior-point methods move the problem constraints to the objective function via barrier functions and solve a sequence of parametric sub-problems [10, Chap. 11], [44, Sect. 16.6]. Although not easily warm started, the polynomial complexity makes interior-point methods appealing for large scale problems [29]. They usually require few but rather demanding iterations [31, 44]. Interior-point methods are currently the default algorithms in the commercial solvers GUROBI [32] and MOSEK [43]. Recent developments are found in the regularized method IP-PMM [51].

Semismooth Newton methods apply a nonsmooth version of Newton's method to the KKT conditions of the original problem [52, 53]. In the strictly convex case, *i.e.*, with $\mathbf{Q} \succ 0$, this approach performs very well as long as the underlying linear systems are nonsingular. Regularized, or stabilized, semismooth Newton-type methods, such as QPALM [33, 34] and FBstab [37], overcome these drawbacks.

The augmented Lagrangian framework [7, 14, 44], semismooth Newton methods [25, 53], and proximal techniques [48, 56] are undergoing a revival, as their seamless combination exhibits valuable properties and provides useful features, such as regularization and numerical stability [6, 33, 37]. These ideas form the basis for our approach.

1.2 Approach

In this work we present a numerical method for solving general QPs. The proposed algorithm is based on the proximal point algorithm and a semismooth Newton method for solving the sub-problems, which are always solvable for any choice of problem data. We therefore impose no restrictions such as strict convexity of the cost function or linear independence of the constraints. As such, our algorithm gathers together the benefits of fully regularized primal-dual methods and semismooth Newton methods with active-set structure. Our algorithm can exploit warm starting to reduce the number of iterations, as well as factorization caching and multi-rank update techniques for efficiency, and it can obtain accurate solutions.

Our approach, dubbed QPDO from "Quadratic Primal-Dual Optimizer", is inspired by and shares many characteristics with algorithms that have already been proposed, in particular with QPALM [33] and FBstab [37]. On the other hand, they differ on some key aspects. QPALM relates to the proximal method of multipliers [33, Rem. 2], which in turn is associated to the classical (primal) augmented Lagrangian function [55]. Instead, FBstab and QPDO apply the proximal point method, yielding exact primal-dual regularization. A more detailed comparison is deferred to Sect. 5. However, FBstab reformulates the sub-problem via the (penalized) Fischer-Burmeister NCP function [11, 21], and adopts the squared residual norm as a merit function for the inner iterative loop; this prevents the use of symmetric sparse linear solvers. Instead, QPDO adopts the minimum NCP function, which leads to symmetric linear systems with active-set structure. Then, we show the primal-dual proximal augmented Lagrangian function, introduced in [27, 54] and [17], is a suitable merit function for the proximal sub-problem, which allows us to perform an exact line-search in a fully primal-dual regularized context. Indeed, we believe, the main contribution of this work consists in recognizing this link, exploiting it to bridge the gap between previously proposed methods, and developing a robust and efficient algorithm that possesses their advantages but does not suffer from their disadvantages.

Notation $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{R}$, $\mathbb{R}_+$, and $\mathbb{R}_{++}$ denote the sets of natural, integer, real, non-negative real, and positive real numbers, respectively. We denote $\overline{\mathbb{R}} := \mathbb{R} \cup \{-\infty, \infty\}$ the extended real line. The identity matrix and the vector of ones of size n are denoted by $\mathbf{I}_n$ and $\mathbf{1}_n$, respectively. We may omit subscripts whenever clear from the context. $[a, b]$, (a, b), $[a, b)$, and $(a, b]$ stand for closed, open, and half-open intervals, respectively, with end points a and b. $[a; b]$, $(a; b)$, $[a; b)$, and $(a; b]$ stand for discrete intervals, *e.g.*, $[a;b] = [a, b] \cap \mathbb{Z}$. Given a vector $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^\top$ and $\mathbf{x}^i$ denote its transpose and its i-th component, respectively. We adopt the norms $\|\mathbf{x}\| = \|\mathbf{x}\|_2 := \sqrt{\mathbf{x}^\top \mathbf{x}}$ and $\|\mathbf{x}\|_\infty := \max_{i \in [1;n]} |\mathbf{x}^i|$. Given a set S, $|S|$ denotes its cardinality. In $\mathbb{R}^n$, the relations $<$, $\leq$, $=$, $\geq$, and $>$ are understood component-wise. Given a nonempty

closed convex set $\mathcal{C} \subseteq \mathbb{R}^n$, we denote $\chi_{\mathcal{C}} : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ its characteristic function, namely $\chi_{\mathcal{C}}(\mathbf{x}) = 0$ if $\mathbf{x} \in \mathcal{C}$ and $\chi_{\mathcal{C}}(\mathbf{x}) = +\infty$ otherwise, $dist_{\mathcal{C}} : \mathbb{R}^n \to \mathbb{R}$ its distance, namely $\mathbf{x} \mapsto \min_{\mathbf{z}\in\mathcal{C}} \|\mathbf{z} - \mathbf{x}\|$, and its projection $\Pi_{\mathcal{C}} : \mathbb{R}^n \to \mathbb{R}^n$, namely $\mathbf{x} \mapsto \arg\min_{\mathbf{z}\in\mathcal{C}} \|\mathbf{z} - \mathbf{x}\|$. Thus, it holds $dist_{\mathcal{C}}(\mathbf{x}) = \|\Pi_{\mathcal{C}}(\mathbf{x}) - \mathbf{x}\|$.

The algorithm is described with a nested structure, whose outer iterations are indexed by $k \in \mathbb{N}$. Given an arbitrary vector $\mathbf{x}$, $\mathbf{x}_k$ denotes that $\mathbf{x}$ depends on k, and analogously for matrices. We denote $\mathbf{y}$ the dual variable associated with the constraints in problem (1). A primal-dual pair $(\mathbf{x}, \mathbf{y})$ will be denoted $\mathbf{v}$, and we will refer interchangeably to it as a vector or to its components $\mathbf{x}$ and $\mathbf{y}$. An optimal solution to (1) will be denoted $(\mathbf{x}^\star, \mathbf{y}^\star)$, or $\mathbf{v}^\star$. Optimal solutions of proximal sub-problems will be denoted using an appropriate subscript, according to the iteration. For example, $(\mathbf{x}_k^\star, \mathbf{x}_k^\star)$, and $\mathbf{x}_k^\star$, denote the solution to the proximal sub-problem corresponding to the k-th outer iteration.

Outline The rest of the paper is organized as follows. Sections 2 and 3 develop and present our method in detail. In particular, in Sect. 3.1 we establish our key result, which relates the proximal operator and the primal-dual proximal augmented Lagrangian function. Our algorithmic framework is outlined in Section 4 and the convergence properties are analyzed in Sect. 4.1, while Sect. 5 contrasts QPDO with similar methods. We present details of our implementation in Sect. 6 and report on numerical experience in Sect. 7.

2 Outer loop: inexact proximal point method

Our method solves (1) using the proximal point algorithm with inexact evaluation of the proximal operator. The latter is evaluated by means of a semismooth Newton-type method, which constitutes an inner iterative procedure further investigated in Sect. 3. Here we focus on the outer loop corresponding to the proximal point algorithm, which has been extensively studied in the literature [56]. We recall some important results and refer to [38, 40, 45, 55] for more details.

2.1 Optimality conditions

Problem (1) can be equivalently expressed as

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) + g(\mathbf{A}\mathbf{x}), \tag{2}$$

where

$$f(\mathbf{x}) := \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{q}^\top \mathbf{x} \qquad \text{and} \qquad g(\mathbf{z}) := \chi_{\mathcal{C}}(\mathbf{z})$$

are the objective function and the characteristic function of the constraint set $\mathcal{C}$, respectively. The necessary and sufficient first-order optimality conditions of (2), and hence (1), read

$$\mathbf{0} \in \mathcal{T}(\mathbf{v}) := \begin{pmatrix} \mathbf{Q}\mathbf{x} + \mathbf{q} + \mathbf{A}^\top \mathbf{y} \\ -\mathbf{A}\mathbf{x} + \partial g^*(\mathbf{y}) \end{pmatrix}, \tag{3}$$

where ∂g^* denotes the (set-valued) conjugate subdifferential of g [45]. For all $i \in [1;m]$, $(\partial g^*(\mathbf{y}))^i = \mathbf{l}^i$ if $\mathbf{y}^i < 0$, $(\partial g^*(\mathbf{x}))^i = [\mathbf{l}^i, \mathbf{u}^i]$ if $\mathbf{x}^i = 0$, and $(\partial g^*(\mathbf{y}))^i = \mathbf{u}^i$ if $\mathbf{y}^i > 0$. We will refer to $\mathcal{T} : \mathbb{R}^\ell \rightrightarrows \mathbb{R}^\ell, \ell := n + m$, as the KKT operator for (1). However, noticing that, for any $\alpha > 0$, the conditions $\mathbf{v} = \Pi_{\mathcal{C}}(\mathbf{v} + \alpha\mathbf{u})$ and $\mathbf{v} \in \partial g^*(\mathbf{u})$ are equivalent [57, Sect. 23], conditions in (3) can be reformulated. Choosing $\alpha = 1$, we define the (outer) residual $\mathbf{r} : \mathbb{R}^\ell \to \mathbb{R}^\ell$ and equivalently express (3) as

$$\mathbf{0} = \mathbf{r}(\mathbf{v}) := \begin{pmatrix} \mathbf{Q}\mathbf{x} + \mathbf{q} + \mathbf{A}^\top \mathbf{y} \\ \mathbf{A}\mathbf{x} - \Pi_{\mathcal{C}}(\mathbf{A}\mathbf{x} + \mathbf{y}) \end{pmatrix}. \tag{4}$$

This reformulation can be obtained also by employing the minimum NCP function [60] and rearranging to obtain the projection operator $\Pi_{\mathcal{C}}$. The residual $\mathbf{r}$ is analogous to the natural residual function $\boldsymbol{\pi}$ investigated in [47]. Since it is an error bound for problem (1), *i.e.*, $dist_{\mathcal{T}^{-1}(\mathbf{0})}(\mathbf{v}) = \mathcal{O}(\|\mathbf{r}(\mathbf{v})\|)$ [47, Thm 18], $\mathbf{r}$ is a suitable optimality measure and its norm can be adopted as a stopping criterion. Although equivalent, (3) is considered here only as a theoretical tool for developing the proposed method, whereas the outer residual $\mathbf{r}$ in (4) serves as a computationally practical optimality criterion.

2.2 Proximal point algorithm

The proximal point algorithm [56] finds zeros of maximal monotone operators by recursively applying their proximal operator. Since $\mathcal{T}$ is a maximal monotone operator [45, 55], the proximal point algorithm converges to an element $\mathbf{v}^\star$ of the set of primal-dual solutions $\mathcal{T}^{-1}(\mathbf{0})$, if any exists [40, 56]. Starting from an initial guess $\mathbf{v}_0$, it generates a sequence $\{\mathbf{v}_k\}$ of primal-dual pairs by recursively applying the proximal operator $\mathcal{P}_k$:

$$\mathbf{v}_{k+1} = \mathcal{P}_k(\mathbf{v}_k), \qquad \mathcal{P}_k := (\mathbf{I} + \boldsymbol{\Sigma}_k^{-1}\mathcal{T})^{-1}, \tag{5}$$

where $\{\boldsymbol{\Sigma}_k\}$ is a sequence of non-increasing positive definite matrices, namely, $\boldsymbol{\Sigma}_k \succ 0$ and $\boldsymbol{\Sigma}_k - \boldsymbol{\Sigma}_{k+1} \succeq 0$ for all $k \in \mathbb{N}$. The matrices $\boldsymbol{\Sigma}_k$ control the primal-dual proximal regularization and, similarly to exact penalty methods, these are not required to vanish [55, 56]. Since $\mathcal{T}$ is maximal monotone, $\mathcal{P}_k$ is well defined and single valued for all $\mathbf{v} \in dom\mathcal{T} = \mathbb{R}^\ell$ [40]. Thus, from (5), evaluating $\mathcal{P}_k$ at $\mathbf{v}_k$ is equivalent to finding the unique $\mathbf{v} \in \mathbb{R}^\ell$ that satisfies

$$\mathbf{0} \in \mathcal{T}_k(\mathbf{v}) := \mathcal{T}(\mathbf{v}) + \boldsymbol{\Sigma}_k(\mathbf{v} - \mathbf{v}_k). \tag{6}$$

This is guaranteed to have a unique solution and to satisfy certain useful regularity properties; see Sect. 3 below. As a result, we can construct a fast inner solver for these sub-problems based on the semismooth Newton method.

2.3 Early termination

The proximal point algorithm tolerates errors, namely the inexact evaluation of $\mathcal{P}_k$ [56]. Criterion (A_r) in [38] provides conditions for the design of convergent inexact proximal point algorithms [38, Thm 2.1]. Let $\mathbf{v}_k^\star := \mathcal{P}_k(\mathbf{v}_k)$ denote the unique proximal sub-problem solution and $\mathbf{v}_{k+1} \approx \mathbf{v}_k^\star$ the actual recurrence update. The aforementioned criterion requires

$$\|\mathbf{v}_{k+1} - \mathbf{v}_k^\star\| \leq e_k \min\left(1, \|\mathbf{v}_{k+1} - \mathbf{v}_k\|^r\right), \tag{7}$$

where $r \geq 0$ and $\{e_k\}$ is a summable sequence of nonnegative inner tolerances, *i.e.*, $e_k \geq 0$ for all k and $\sum_{k=0}^{\infty} e_k < +\infty$. However, since $\mathbf{v}_k^\star$ is effectively unknown, this criterion is impractical. Instead, in Algorithm 1 it is required that $\mathbf{v}_{k+1}$ satisfy $\|\mathbf{r}_k(\mathbf{v}_{k+1})\|_\infty \leq \varepsilon_k$. Here, $\mathbf{r}_k$ denotes the residual for the k-th sub-problem, and is defined in (14). In Sect. 4.1 we will show that this criterion is a simple and viable substitute, which retains the significance of (A_r).

2.4 Warm starting

If a solution $\mathbf{v}^\star$ exists, the (outer) sequence $\{\mathbf{v}_k\}$ generated by (5) converges, by the global convergence of the proximal point algorithm [56]. Then, $\mathcal{P}_k(\mathbf{v}_k)$ and $\mathbf{v}_k$ are arbitrarily close to each other for sufficiently large k [37, Sect. 4]. This supports the idea of warm starting the inner solver with the current outer estimate $\mathbf{v}_k$, that is, setting $\mathbf{v} \leftarrow \mathbf{v}_k$ in Algorithm 2. In practice, for large k, only one or few Newton-type inner iterations are needed to find an approximate sub-problem solution $\mathbf{v}_{k+1}$.

2.5 Primal and dual infeasibility

Infeasibility detection in convex programming has been studied in [4, 5]. Certifying primal infeasibility of (1) amounts to finding a vector $\mathbf{y} \in \mathbb{R}^m$ such that

$$\mathbf{A}^\top \mathbf{y} = \mathbf{0}, \quad \sum_{\substack{i \in [1;m] \\ \mathbf{u}^i \in \mathbb{R}}} \mathbf{u}^i \max(\mathbf{y}^i, 0) + \sum_{\substack{i \in [1;m] \\ \mathbf{l}^i \in \mathbb{R}}} \mathbf{l}^i \min(\mathbf{y}^i, 0) < 0. \tag{8}$$

Similarly, it can be shown that a vector $\mathbf{x} \in \mathbb{R}^n$ satisfying

$$\mathbf{Q}\mathbf{x} = \mathbf{0}, \quad \mathbf{q}^\top \mathbf{x} < 0, \quad (\mathbf{A}\mathbf{x})^i \begin{cases} = 0 \ \mathbf{l}^i, \mathbf{u}^i \in \mathbb{R}, \\ \geq 0 \ \mathbf{l}^i \in \mathbb{R}, \mathbf{u}^i = +\infty, \\ \leq 0 \ \mathbf{l}^i = -\infty, \mathbf{u}^i \in \mathbb{R}, \end{cases} \quad i \in [1;m], \tag{9}$$

is a certificate of dual infeasibility for (1) [4, Prop. 3.1].

3 Inner loop: semismooth Newton method

In this section we focus on solving (6) via a semismooth Newton method. For the sake of clarity, and without loss of generality, we consider

$$\mathbf{\Sigma}_k := blockdiag(\sigma_k \mathbf{I}_n, \mu_k \mathbf{I}_m).$$

for some parameters $\sigma_k, \mu_k \in \mathbb{R}_{++}$.

3.1 Merit function

We now derive the simple yet fundamental result that is the key to developing our method. This provides the NCP reformulation of the proximal sub-problem with a suitable merit function. The former yields symmetric active-set linear systems, while the latter leads to exact linesearch.

Let us express (6) in the form

$$\mathbf{0} \in \begin{pmatrix} \mathbf{Q}\mathbf{x} + \mathbf{q} + \mathbf{A}^\top \mathbf{y} + \sigma_k(\mathbf{x} - \mathbf{x}_k) \\ -\mathbf{A}\mathbf{x} + \mu_k(\mathbf{y} - \mathbf{y}_k) + \partial g^*(\mathbf{y}) \end{pmatrix}. \tag{10}$$

Similarly to (4), for any given $\alpha > 0$, this can be rewritten as

$$\mathbf{0} = \begin{pmatrix} \mathbf{Q}\mathbf{x} + \mathbf{q} + \mathbf{A}^\top \mathbf{y} + \sigma_k(\mathbf{x} - \mathbf{x}_k) \\ \mathbf{A}\mathbf{x} + \mu_k(\mathbf{y}_k - \mathbf{y}) - \Pi_{\mathcal{C}}(\mathbf{w}_k) \end{pmatrix}, \tag{11}$$

where we denote

$$\mathbf{w}_k := \mathbf{A}\mathbf{x} + \mu_k(\mathbf{y}_k - \mathbf{y}) + \alpha \mathbf{y}.$$

The second condition in (11) can be expressed as $\mathbf{0} = \mathbf{w}_k - \Pi_{\mathcal{C}}(\mathbf{w}_k) - \alpha \mathbf{x}$. Then, we substitute y with $[\mathbf{w}_k - \Pi_{\mathcal{C}}(\mathbf{w}_k)]/\alpha$ in the first condition in (11), and multiply the second one by $(\alpha - \mu_k)/\alpha$. Hence, for any positive $\alpha \neq \mu_k$, (11) is equivalent to

$$\mathbf{0} = \begin{pmatrix} \mathbf{Q}\mathbf{x} + \mathbf{q} + \frac{1}{\alpha}\mathbf{A}^\top [\mathbf{w}_k - \Pi_{\mathcal{C}}(\mathbf{w}_k)] + \sigma_k(\mathbf{x} - \mathbf{x}_k) \\ \frac{\alpha - \mu_k}{\alpha}[\mathbf{w}_k - \Pi_{\mathcal{C}}(\mathbf{w}_k)] + (\mu_k - \alpha)\mathbf{x} \end{pmatrix}, \tag{12}$$

namely their unique solutions coincide. Now, we observe that the right-hand side of (12) is the gradient of the function

$$f(\mathbf{x}) + \frac{1}{2\alpha} dist^2_{\mathcal{C}}(\mathbf{w}_k) + \frac{\sigma_k}{2}\|\mathbf{x} - \mathbf{x}_k\|^2 + \frac{\mu_k - \alpha}{2}\|\mathbf{y}\|^2. \tag{13}$$

By construction, this is a continuously differentiable function whose gradient vanishes at the unique solution of the proximal sub-problem. Furthermore, for any $\alpha \in (0, \mu_k)$, it is strictly convex and hence admits a unique minimizer that must coincide with the unique proximal point. Therefore, (13) is a suitable merit function for

the sub-problem. The particular choice $\alpha := \mu_k/2$ inherits all these properties and leads to the inner optimality conditions

$$\mathbf{0} = \mathbf{r}_k(\mathbf{v}) := \begin{pmatrix} \mathbf{Q}\mathbf{x} + \mathbf{q} + \mathbf{A}^\top \mathbf{y} + \sigma_k(\mathbf{x} - \mathbf{x}_k) \\ \mathbf{A}\mathbf{x} + \mu_k(\mathbf{y}_k - \mathbf{y}) - \Pi_{\mathcal{C}}(\mathbf{A}\mathbf{x} + \mu_k(\mathbf{y}_k - \mathbf{y}/2)) \end{pmatrix}, \tag{14}$$

with $\mathbf{r}_k : \mathbb{R}^\ell \to \mathbb{R}^\ell$ the inner residual, and the associated merit function

$$\mathcal{M}_k(\mathbf{v}) := f(\mathbf{x}) + \frac{1}{\mu_k} dist^2_{\mathcal{C}}(\mathbf{A}\mathbf{x} + \mu_k(\mathbf{y}_k - \mathbf{y}/2)) + \frac{\sigma_k}{2}\|\mathbf{x} - \mathbf{x}_k\|^2 + \frac{\mu_k}{4}\|\mathbf{y}\|^2. \tag{15}$$

In fact, $\mathcal{M}_k : \mathbb{R}^\ell \to \mathbb{R}$ is the primal-dual proximal augmented Lagrangian function [17, 27, 54]; see Appendix A for a detailed derivation. This underlines once again the strong relationship between the proximal point algorithm and the augmented Lagrangian framework, pioneered in [55]. On the one hand, by (15), the dual regularization parameter μ_k controls the constraint penalization [23, Sect. 3.2]. On the other hand, it provides an "interpretation of the augmented Lagrangian method as an adaptive constraint regularization process" [3, Sect. 2].

The inner residual $\mathbf{r}_k$ in (14) is piecewise affine, hence strongly semismooth on $\mathbb{R}^\ell$ [36, 52]. In fact, given μ_k bounded away from zero and the unique, bounded, and nonsingular matrix $\mathbf{T}_k$ defined by

$$\mathbf{T}_k := \begin{bmatrix} \mathbf{I} & \frac{2}{\mu_k}\mathbf{A}^\top \\ \mathbf{0} & -\mathbf{I} \end{bmatrix}, \tag{16}$$

we have the identity $\nabla\mathcal{M}_k(\cdot) = \mathbf{T}_k\mathbf{r}_k(\cdot)$. Effectively, $\|\mathbf{r}_k(\cdot)\|$ can be employed as stopping criterion in place of $\|\nabla\mathcal{M}_k(\cdot)\|$. We prefer the former, since $\mathbf{r}_k$ corresponds to a perturbation of the outer residual $\mathbf{r}$; cf. (4).

The availability of a suitable merit function allows us to adopt a damped Newton-type method and design a linesearch globalization strategy, in contrast with [25, 37, 50]. Since $\mathcal{M}_k$ is continuously differentiable and piecewise quadratic, an exact linesearch procedure can be carried out, which yields finite convergence [61].

Finally, we highlight that the method asymptotically reduces to a sequence of regularized semismooth Newton steps applied to the original, unperturbed optimality system, in the vein of [2]. This closely relates to the concept of exact regularization [24]. In fact, the proximal primal-dual regularization is exact; see Theorem 1 and compare with [3, Thm 1].

Proposition 1 *Let $k \in \mathbb{N}$ be arbitrary.*

(i) *Suppose $\mathbf{v}_k^\star$ solves (14) for $\mathbf{v}_k := \mathbf{v}_k^\star$ and for some $\sigma_k \geq 0$ and $\mu_k > 0$. Then, $\mathbf{v}_k^\star$ solves (4).*

(ii) *Alternatively, suppose $\mathbf{v}_k^\star$ solves (14) for $\mathbf{y}_k := \mathbf{y}_k^\star$, $\sigma_k := 0$, and for some $\mu_k > 0$. Then, $\mathbf{v}_k^\star$ solves (4).*

(iii) *Conversely, suppose* $\mathbf{v}^\star$ *solves* (4). *Then,* $\mathbf{v}^\star$ *solves* (14) *for* $\mathbf{v}_k := \mathbf{v}^\star$ *and for any* $\sigma_k \geq 0$ *and* $\mu_k > 0$.

Proof The proof is immediate by direct comparison of (4) and (14). □

Subproblem (14) is equivalent to the unconstrained minimization of the primal-dual augmented Lagrangian function $\mathcal{M}_k$, given in (15). However, by introducing the auxiliary variable $\mathbf{s} \in \mathbb{R}^m$, we can rewrite subproblem (14) as the equivalent yet smoother problem

$$\begin{aligned} \min_{\mathbf{x},\mathbf{s}} \quad & \frac{1}{2}\mathbf{x}^\top \mathbf{Q}\mathbf{x} + \mathbf{q}^\top \mathbf{x} + \frac{\sigma_k}{2}\|\mathbf{x} - \mathbf{x}_k\|^2 + \frac{1}{2\mu_k}\|\mathbf{s} - \mu_k \mathbf{y}_k\|^2 \\ s.t. \quad & \mathbf{l} \leq \mathbf{A}\mathbf{x} + \mathbf{s} \leq \mathbf{u}, \end{aligned} \tag{17}$$

that is a primal-dual proximal regularization of (1). Indeed, it is always feasible and strictly convex and the constraints satisfy the linear independence constraint qualification (LICQ). This shows that each outer iteration is associated to a regularized QP, which can be effectively solved by Newton-type methods.

3.2 Search direction

A semismooth Newton direction $\delta\mathbf{v} = (\delta\mathbf{x}, \delta\mathbf{y})$ at $\mathbf{v} = (\mathbf{x}, \mathbf{y})$ solves

$$\mathbf{V}_k(\mathbf{v})\delta\mathbf{v} = -\mathbf{r}_k(\mathbf{v}), \tag{18}$$

where

$$\mathbf{V}_k(\mathbf{v}) = \begin{bmatrix} \mathbf{Q} + \sigma_k \mathbf{I} & \mathbf{A}^\top \\ (\mathbf{I} - \mathbf{P}_k(\mathbf{v}))\mathbf{A} & -\mu_k(\mathbf{I} - \mathbf{P}_k(\mathbf{v})/2) \end{bmatrix} \tag{19}$$

is an element of the generalized Jacobian [57, Sect. 23] of $\mathbf{r}_k$ at $\mathbf{v}$. In turn, the diagonal matrix $\mathbf{P}_k(\mathbf{v})$ with entries

$$\mathbf{P}_k^{ii}(\mathbf{v}) := \begin{cases} 1 \text{ if } \mathbf{l}^i < \mathbf{w}_k^i < \mathbf{u}^i \\ 0 \text{ otherwise} \end{cases}, \; i = 1, \ldots, m, \tag{20}$$

is an element of the generalized Jacobian of Π_C at $\mathbf{w}_k$. Owing to (20), (18) can be rewritten in symmetric form, similar to those arising in active-set methods [35]. To this end, we notice that, if $\mathbf{P}_k^{ii}(\mathbf{v}) = 1$, the corresponding inner residual in (14) simplifies into $\mathbf{r}_k^{n+i}(\mathbf{v}) = -\mu_k \mathbf{y}^i/2$, and the linear equation in (18) gives $\delta\mathbf{y}^i = -\mathbf{y}^i$. This yields the crucial observation that, by (20), $\mathbf{P}_k(\mathbf{v})\delta\mathbf{y} = -\mathbf{P}_k(\mathbf{v})\mathbf{y}$ for all $\mathbf{v} \in \mathbb{R}^\ell$. Then, an equivalent yet symmetric linear system is obtained, whose solution is the search direction $\delta\mathbf{v}$ at $\mathbf{v}$:

$$\begin{bmatrix} \mathbf{Q} + \sigma_k \mathbf{I} & \mathbf{A}^\top(\mathbf{I} - \mathbf{P}_k(\mathbf{v})) \\ (\mathbf{I} - \mathbf{P}_k(\mathbf{v}))\mathbf{A} & -\mu_k(\mathbf{I} - \mathbf{P}_k(\mathbf{v})/2) \end{bmatrix} \begin{pmatrix} \delta\mathbf{x} \\ \delta\mathbf{y} \end{pmatrix} = \begin{pmatrix} \mathbf{A}^\top \mathbf{P}_k(\mathbf{v})\mathbf{y} \\ \mathbf{0} \end{pmatrix} - \mathbf{r}_k(\mathbf{v}). \tag{21}$$

The active-set structure introduced by $\mathbf{P}_k$ allows us to obtain a symmetric linear system and adopt multi-rank factorization updates [15, 26] while maintaining structure and sparsity of the coefficient matrix [13, 59]. The linear system in (21) always admits a unique solution, since the coefficient matrix is symmetric quasi-definite [63], independent of the problem data.

3.3 Exact linesearch

Given a primal-dual pair $\mathbf{v}$ and a search direction $\delta\mathbf{v}$, we seek a stepsize $\tau > 0$ to effectively update $\mathbf{v}$ to $\mathbf{v} + \tau\,\delta\mathbf{v}$ in Algorithm 2. Similarly to $\mathcal{M}_k$, the function $\psi_k : \tau \mapsto \mathcal{M}_k(\mathbf{v} + \tau\delta\mathbf{v})$ is continuously differentiable, piecewise quadratic, and strictly convex. Thus, the optimal stepsize $\tau := arg\,min_{t\in\mathbb{R}}\psi_k(t)$ is found as the unique zero of ψ_k', *i.e.*, $\psi_k'(\tau) = 0$. Since ψ_k' is a piecewise linear, strictly monotone increasing function, the exact linesearch procedure amounts to solving a piecewise linear equation of the form

$$0 = \alpha_k\tau + \beta_k + \frac{2}{\mu_k}\delta\mathbf{w}_k\big[\mathbf{w}_k + \tau\delta\mathbf{w}_k - \Pi_{\mathcal{C}}\big(\mathbf{w}_k + \tau\delta\mathbf{w}_k\big)\big] \tag{22}$$

with respect to $\tau \in \mathbb{R}$. Here, the coefficients are given by

$$\alpha_k := \delta\mathbf{x}^\top(\mathbf{Q} + \sigma_k\mathbf{I})\delta\mathbf{x} + \mu_k\delta\mathbf{y}^\top\delta\mathbf{y}/2, \tag{23a}$$

$$\beta_k := \delta\mathbf{x}^\top[\mathbf{Q}\mathbf{x} + \mathbf{q} + \sigma_k(\mathbf{x} - \mathbf{x}_k)] + \mu_k\delta\mathbf{y}^\top\mathbf{y}/2, \tag{23b}$$

$$\mathbf{w}_k := \mathbf{A}\mathbf{x} + \mu_k\big(\mathbf{y}_k - \mathbf{y}/2\big), \tag{23c}$$

$$\delta\mathbf{w}_k := \mathbf{A}\delta\mathbf{x} - \mu_k\delta\mathbf{y}/2, \tag{23d}$$

whose derivation is reported in Appendix B. Thanks to its peculiar structure, (22) can be solved efficiently and exactly (up to numerical precision), *e.g.*, by sorting and linear interpolation, cf. [33, Alg. 2].

We underline that the stepsize τ is unique and strictly positive, since $\mathcal{M}_k$ is strictly convex and $\delta\mathbf{v}$ is a descent direction for $\mathcal{M}_k$ at $\mathbf{v}$. This follows from the observation that

$$\psi_k'(0) = \delta\mathbf{v}^\top\nabla\mathcal{M}_k(\mathbf{v}) = \delta\mathbf{v}^\top\mathbf{T}_k\mathbf{r}_k(\mathbf{v}) = -\delta\mathbf{v}^\top\mathbf{T}_k\mathbf{V}_k(\mathbf{v})\delta\mathbf{v} < 0,$$

since $\partial^2\mathcal{M}_k(\mathbf{v}) \ni \mathbf{T}_k\mathbf{V}_k(\mathbf{v}) \succ 0$.

4 Algorithm and convergence

Our Quadratic Primal-Dual Optimizer (QPDO), which weaves together the proximal point algorithm and a semismooth Newton method, is outlined in Algorithms 1 and 2. We highlight the nested structure for clarity of presentation. Effectively,

Algorithm 1 corresponds to the proximal point algorithm, as discussed in Sect. 2. The proximal operator, $\mathcal{P}_k$, is evaluated in Algorithm 2 by solving a sub-problem via the semismooth Newton method, as detailed in Sect. 3. We denote $\mathbf{r}$ and $\mathbf{r}_k$ the outer and inner residuals defined in (4,14), respectively, and $\mathbf{v}$ a primal-dual pair $(\mathbf{x}, \mathbf{y})$. Infeasibility detection, parameters update, and linear solvers are detailed in Sect. 6.

Algorithm 1 QPDO: Quadratic Primal-Dual Optimizer

input: $\mathbf{Q}$, $\mathbf{q}$, $\mathbf{A}$, $\mathbf{l}$, $\mathbf{u}$
parameters: $\epsilon > 0, \epsilon_0 \geq 0, \kappa_\epsilon \in [0, 1), 0 < \sigma_{\min} \leq \sigma_0, 0 < \mu_{\min} \leq \mu_0$
guess: $\mathbf{x}_0 \in \mathbb{R}^n$, $\mathbf{y}_0 \in \mathbb{R}^m$
for $k = 0, 1, 2, \ldots$ **do**
  if $\|\mathbf{r}(\mathbf{v}_k)\|_\infty \leq \epsilon$ **then**
    return $\mathbf{v}_k$
  end if
  find $\mathbf{v}_{k+1}$ such that $\|\mathbf{r}_k(\mathbf{v}_{k+1})\|_\infty \leq \epsilon_k$ by invoking Algorithm 2
  check for primal-dual infeasibility with $\Delta\mathbf{v}_k := \mathbf{v}_{k+1} - \mathbf{v}_k$
  choose parameters $\sigma_{k+1} \in [\sigma_{\min}, \sigma_k]$ and $\mu_{k+1} \in [\mu_{\min}, \mu_k]$
  set $\epsilon_{k+1} \leftarrow \kappa_\epsilon \epsilon_k$
end for

Algorithm 2 Inner loop: semismooth Newton method

$\mathbf{v} \leftarrow \mathbf{v}_k$
repeat
  get the search direction $\delta\mathbf{v} \in \mathbb{R}^\ell$ by solving the linear system (21)
  get the stepsize $\tau \in \mathbb{R}_{++}$ by solving the piecewise linear equation (22)
  set $\mathbf{v} \leftarrow \mathbf{v} + \tau\, \delta\mathbf{v}$
until $\|\mathbf{r}_k(\mathbf{v})\|_\infty \leq \epsilon_k$
$\mathbf{v}_{k+1} \leftarrow \mathbf{v}$

4.1 Convergence analysis

This section discusses the convergence of QPDO as outlined in Algorithm 1 and 2. We show that the proposed algorithm either generates a sequence of iterates $\{\mathbf{v}_k\}$ that in the limit satisfy the optimality conditions (4), when problem (1) is solvable, or provides a certificate of primal and/or dual infeasibility otherwise. Our analysis relies on well-established results for Newton and proximal point methods; in particular, we refer to [38, 56, 61].

First, we focus on the inner loop, described in Algorithm 2 and detailed in Sect. 3.

Lemma 1 *Consider an arbitrary but fixed outer iteration, indexed by $k \in \mathbb{N}$, and suppose $\varepsilon_k \geq 0$. Then, the procedure in Algorithm 2 is well defined and terminates after finitely many steps.*

Proof The search direction $\delta\mathbf{v}$ exists and is unique, since linear system (21) is always solvable. Similarly, there exists a unique, positive optimal stepsize τ which solves (22). Thus, all steps of Algorithm 2 are well-defined. Since $\mathcal{M}_k$ is continuously differentiable, strictly convex, and piecewise quadratic, the semismooth Newton method with exact linesearch exhibits finite convergence [61, Thm 3]. Thus, $\nabla\mathcal{M}_k(\mathbf{v}) = \mathbf{0}$ after finitely many iterations. Then, by $\nabla\mathcal{M}_k(\cdot) = \mathbf{T}_k\mathbf{r}_k(\cdot)$ with $\mathbf{T}_k$ nonsingular, it reaches $\mathbf{r}_k(\mathbf{v}) = \mathbf{0}$. Hence, for any $\varepsilon_k \geq 0$, the inner stopping criterion $\|\mathbf{r}_k(\mathbf{v})\|_\infty \leq \varepsilon_k$ is eventually satisfied, and the inner loop terminates. □

Notice that, with $\varepsilon_k = 0$, Algorithm 2 returns the unique (proximal) point $\mathbf{v}_k^\star := \mathcal{P}_k(\mathbf{v}_k)$.

Let us consider now the outer loop, sketched in Algorithm 1. Recall that, by construction, the regularization parameters are positive and non-increasing. The outer loop consists of inexact proximal point iterations [56], hence global and local convergence properties can be derived based on [38, Prop. 1.2]. The following result shows that criterion (A_r) [38] holds.

Lemma 2 *Let $\mathcal{T}^{-1}(\mathbf{0})$ be nonempty, any $\mathbf{v}_0 \in \mathbb{R}^\ell$ be given, and the sequence $\{\mathbf{v}_k\}$ be generated by Algorithm* 1. *Then, there exists a summable sequence $\{e_k\} \subseteq \mathbb{R}_+$ such that*

$$\|\mathbf{v}_{k+1} - \mathbf{v}_k^\star\| \leq e_k \quad \forall k \in \mathbb{N}.$$

Proof By $\varepsilon_0 \in \mathbb{R}_+$ and $\kappa_\varepsilon \in [0, 1)$, the sequence $\{\varepsilon_k\} \subseteq \mathbb{R}_+$ is summable, since $\sum_{k\in\mathbb{N}} \varepsilon_k = \sum_{k\in\mathbb{N}} \kappa_\varepsilon^k \varepsilon_0 = \varepsilon_0/(1-\kappa_\varepsilon) < +\infty$. By the inner stopping condition, for all $k \in \mathbb{N}$ it holds $\|\mathbf{r}_k(\mathbf{v}_{k+1})\| \leq \varepsilon_k$. Morever, since $\mathcal{M}_k$ is $\boldsymbol{\Sigma}_k$-strongly convex, we have that, for some $\tilde{\eta}_k > 0$, it is

$$\tilde{\eta}_k\|\mathbf{v} - \mathbf{v}_k^\star\| \leq \|\nabla\mathcal{M}(\mathbf{v}) - \nabla\mathcal{M}(\mathbf{v}_k^\star)\| = \|\nabla\mathcal{M}(\mathbf{v})\| = \|\mathbf{T}_k\,\mathbf{r}_k(\mathbf{v})\|$$

for all $\mathbf{v} \in \mathbb{R}^\ell$. By the boundedness of $\boldsymbol{\Sigma}_k$ away from zero, matrix $\mathbf{T}_k$ is bounded and there exists a constant $\eta > 0$ such that the bound $\|\mathbf{v} - \mathbf{v}_k^\star\| \leq \eta\|\mathbf{r}_k(\mathbf{v})\|$ holds for all $k \in \mathbb{N}$ and $\mathbf{v} \in \mathbb{R}^\ell$. Thus, in particular, for all $k \in \mathbb{N}$ it is

$$\|\mathbf{v}_{k+1} - \mathbf{v}_k^\star\| \leq \eta\|\mathbf{r}_k(\mathbf{v}_{k+1})\| \leq \eta\varepsilon_k.$$

Let $e_k := \eta\varepsilon_k$, and the proof is complete. □

Notice that we choose $r = 0$ in (A_r), particularly in (7), for the sake of simplicity, although this may prevent faster convergence; see [38, Thm 2.1]. Relying on the inexact proximal point algorithm, the following result states that Algorithm 1 converges to a solution, if one exists.

Theorem 1 *Let $\mathcal{T}^{-1}(\mathbf{0})$ be nonempty, any $\mathbf{v}_0 \in \mathbb{R}^\ell$ be given, and the sequence $\{\mathbf{v}_k\}$ be generated by Algorithm* 1. *Then, the sequence $\{\mathbf{v}_k\}$ is well defined and converges to a solution $\mathbf{v}^\star \in \mathcal{T}^{-1}(\mathbf{0})$.*

Proof The error bound condition, namely criterion (A_r), is enforced by costruction; cf. Lemma 2. It remains to show that there exists $a, \varepsilon > 0$ such that for all $\mathbf{u} \in \mathbb{R}^\ell$, $\|\mathbf{u}\| \leq \varepsilon$, it holds $dist_{\mathcal{T}^{-1}(\mathbf{0})}(\mathbf{v}) \leq a\|\mathbf{u}\|$ for all $\mathbf{v} \in \mathcal{T}^{-1}(\mathbf{u})$. Since problem (3) is a polyhedral variational inequality, this property holds globally [19, Sect. 3D]. Hence, we can invoke [38, Prop. 1.2] to conclude that $\|\mathbf{v}_k - \mathbf{v}^\star\| \to 0$. □

Finally, Theorem 2 guarantees that Algorithm 1 terminates if the original problem (1) does not admit any solution. This allows our method to detect infeasibility and to return a certificate.

Theorem 2 *Suppose problem* (1) *is primal and/or dual infeasible, i.e.,* $\mathcal{T}^{-1}(\mathbf{0})$ *is empty. Let any* $\mathbf{v}_0 \in \mathbb{R}^\ell$ *be given, the sequence* $\{\mathbf{v}_k\}$ *be generated by Algorithm* 1, *and define* $\Delta\mathbf{v}_k := \mathbf{v}_{k+1} - \mathbf{v}_k$. *Then, the sequence* $\{\Delta\mathbf{v}_k\}$ *admits a limit* $\Delta\mathbf{v}$, *i.e.,* $\Delta\mathbf{v}_k \to \Delta\mathbf{v}$. *Moreover,*

(i) *if* $\Delta\mathbf{y} \neq \mathbf{0}$, *then problem* (1) *is primal infeasible and* $\Delta\mathbf{y}$ *satisfies the primal infeasibility condition* (8);
(ii) *if* $\Delta\mathbf{x} \neq \mathbf{0}$, *then problem* (1) *is dual infeasible and* $\Delta\mathbf{x}$ *satisfies the dual infeasibility condition* (9).

Proof Lemma 5.1 in [4] ensures that $\Delta\mathbf{v}_k \to \Delta\mathbf{v}$, since Algorithm 1 is an instance of the proximal point algorithm. If $\mathcal{T}^{-1}(\mathbf{0}) = \emptyset$, then $\Delta\mathbf{v} \neq \mathbf{0}$, and this gives certificates of primal and/or dual infeasibility according to [4, Thm 5.1]. □

5 Relationship with similar methods

Our approach is inspired by and shares many features with other recently developed methods. This section elaborates upon their relationship with QPDO.

FBstab [37] "synergistically combines the proximal point algorithm with a primal-dual semismooth Newton-type method" to solve convex QPs. By adopting the Fischer-Burmeister [11, 21] NCP function, FBstab does not depend on an estimate of the active set, which may result in a more regular behavior than QPDO. In contrast, adopting the minimum NCP function, QPDO can exploit factorization updates, perform exact linesearch by solving a piecewise linear equation, and handle simultaneously bilateral constraints.

QPALM is a "proximal augmented Lagrangian based solver for convex QPs" [33]; recent advancements allow to handle nonconvex QPs as well [34]. Given a primal-dual estimate $\overline{\mathbf{v}}$, the exact, unique resolvent update $\mathbf{v}^\triangle$ of QPALM [33, Eq. 6], with $\boldsymbol{\Sigma} = blockdiag(\sigma^{-1}\mathbf{I}, \mu^{-1}\mathbf{I})$, is given by

$$\mathbf{x}^\triangle = arg\,min_{\mathbf{x}\in\mathbb{R}^n}\ \varphi(\mathbf{x}), \tag{24a}$$

$$\mathbf{y}^{\triangle} = \overline{\mathbf{y}} + \mu^{-1}\left[\mathbf{A}\mathbf{x}^{\triangle} - \Pi_{\mathcal{C}}\left(\mathbf{A}\mathbf{x}^{\triangle} + \mu\overline{\mathbf{y}}\right)\right]. \tag{24b}$$

In (24a), φ is given by [33, Eq. 8]

$$\varphi(\mathbf{x}) := f(\mathbf{x}) + \frac{1}{2\mu} dist_{\mathcal{C}}^2\left(\mathbf{A}\mathbf{x} + \mu\overline{\mathbf{y}}\right) + \frac{\sigma}{2}\|\mathbf{x} - \overline{\mathbf{x}}\|^2$$

and closely resembles $\mathcal{M}_k$ in (15). Since (24a) yields $\nabla\varphi(\mathbf{x}^{\triangle}) = \mathbf{0}$, combining with (24b) and rearranging give

$$\mathbf{0} = \mathbf{Q}\mathbf{x}^{\triangle} + \mathbf{q} + \mathbf{A}^{\top}\mathbf{y}^{\triangle} + \sigma\left(\mathbf{x}^{\triangle} - \overline{\mathbf{x}}\right), \tag{25a}$$

$$\mathbf{0} = \mathbf{A}\mathbf{x}^{\triangle} + \mu\left(\overline{\mathbf{y}} - \mathbf{y}^{\triangle}\right) - \Pi_{\mathcal{C}}\left(\mathbf{A}\mathbf{x}^{\triangle} + \mu\overline{\mathbf{y}}\right). \tag{25b}$$

Conditions Eqs. (25) and (14) differ only in the argument of $\Pi_{\mathcal{C}}$, where the term $-\mu\mathbf{y}/2$ is missing in (25b). This underlines the primal-dual nature of QPDO. A comparative investigation into how QPDO copes with changes in the active set [35] and controls the quality of both primal and dual variables during iterations [2, 28] is a topic for future work.

OSQP is a "solver for convex quadratic programs based on the alternating direction method of multipliers" [59]. Rearranging from [59, Alg. 1], with parameters $\alpha = 1$, $\rho = \mu^{-1}$, and given primal-dual estimate $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ and constraint estimate $\overline{\mathbf{z}}$, the (unique) primal-auxiliary update $(\mathbf{x}^{\diamondsuit}, \mathbf{s}^{\diamondsuit})$ satisfies

$$\mathbf{0} = \mathbf{Q}\mathbf{x}^{\diamondsuit} + \mathbf{q} + \mathbf{A}^{\top}\mathbf{s}^{\diamondsuit} + \sigma(\mathbf{x}^{\diamondsuit} - \overline{\mathbf{x}}), \tag{26a}$$

$$\mathbf{0} = \mathbf{A}\mathbf{x}^{\diamondsuit} + \mu(\overline{\mathbf{y}} - \mathbf{s}^{\diamondsuit}) - \overline{\mathbf{z}}. \tag{26b}$$

Then, the constraint and dual updates are given by $\mathbf{z}^{\diamondsuit} = \Pi_{\mathcal{C}}\left(\overline{\mathbf{z}} + \mu\mathbf{s}^{\diamondsuit}\right)$ and $\mathbf{y}^{\diamondsuit} = \mathbf{s}^{\diamondsuit} + \mu^{-1}\left(\overline{\mathbf{z}} - \mathbf{z}^{\diamondsuit}\right)$, respectively. Although conditions (26) resemble (14), an auxiliary variable $\mathbf{s}$ substitutes the dual variable y and the projection in (14) is replaced by the constraint estimate $\overline{\mathbf{z}}$. This makes sub-problem (26) a linear system and results in a first-order method.

6 Implementation details

QPDO has been implemented in C and provides a MATLAB interface. It can solve QPs of the form (1) and makes no assumptions about the problem data other than convexity; it is available online at

https://github.com/aldma/qpdo.

This section discusses some relevant aspects of the program, such as the linear solver, parameters update rules, infeasibility detection, and problem scaling.

6.1 Linear solver

The linear system (21) is solved with CHOLMOD [12], using a sparse Cholesky factorization. This linear solver is analogous to the one adopted in QPALM [33], for the sake of comparison. Let $(\mathbf{r}_k^{\text{dual}}, \mathbf{r}_k^{\text{prim}})$ partition the inner residual $\mathbf{r}_k$ in (14). Then, formally solving for $\delta\mathbf{y}$ in (21), we obtain the expression (omitting subscripts and arguments)

$$\begin{aligned}\delta\mathbf{y} &= \mu^{-1}(\mathbf{I}-\mathbf{P}/2)^{-1}\left[(\mathbf{I}-\mathbf{P})\mathbf{A}\delta\mathbf{x} + \mathbf{r}^{\text{prim}}\right] \\ &= \mu^{-1}(\mathbf{I}+\mathbf{P})\left[(\mathbf{I}-\mathbf{P})\mathbf{A}\delta\mathbf{y} + \mathbf{r}^{\text{prim}}\right] \\ &= \mu^{-1}(\mathbf{I}-\mathbf{P})\mathbf{A}\delta\mathbf{x} + \mu^{-1}(\mathbf{I}+\mathbf{P})\mathbf{r}^{\text{prim}},\end{aligned}$$

where the second and third lines are due to the binary structure of $\mathbf{P}$. Substituting $\mathbf{y}$ and rearranging, we obtain a linear system for $\delta\mathbf{x}$:

$$\left[\mathbf{Q} + \sigma\mathbf{I} + \mu^{-1}\mathbf{A}^\top(\mathbf{I}-\mathbf{P})\mathbf{A}\right]\delta\mathbf{x} = \mathbf{A}^\top\mathbf{P}\mathbf{y} - \mu^{-1}\mathbf{A}^\top(\mathbf{I}-\mathbf{P})\mathbf{r}^{\text{prim}} - \mathbf{r}^{\text{dual}},$$

which has a symmetric, positive definite coefficient matrix and can be solved by CHOLMOD [12]. On the one hand, this approach allows multi-rank factorization updates [15], thus avoiding the need for a full re-factorization at every inner iteration. On the other hand, sparsity pattern may be lost and significant fill-in may arise due to the matrix-matrix product $\mathbf{A}^\top\mathbf{A}$. For this reason, the current implementation may benefit from directly solving (21) via sparse symmetric linear solvers, possibly with multi-rank factorization updates. To better exploit the data sparsity pattern and the capabilities of the proposed method, we plan to add other linear solvers in future versions.

6.2 Parameters selection

Solving convex QPs via the proximal point algorithm imposes mild restrictions on the sequence of primal-dual regularization parameters $\{\boldsymbol{\Sigma}_k\}$. As mentioned in Sect. 2.2, there are no additional requirements other than being non-increasing and positive definite. However, similarly to forcing sequences in augmented Lagrangian methods [14], the sequence of regularization parameters greatly affects the behaviour of QPDO, and a careful tuning can positively impact the performance. For instance, although faster convergence rates can be expected if $\boldsymbol{\Sigma}_k \to \mathbf{0}$ [38], numerical stability and machine precision should be taken into account. Following [34, Sect. 5.3] and [59, Sect. 5.2], our implementation considers only diagonal matrices of the form $\boldsymbol{\Sigma}_k = \mathit{blockdiag}(\sigma_k\mathbf{I}, \mathit{diag}(\boldsymbol{\mu}_k))$, and we refer to the effect of σ_k and $\boldsymbol{\mu}_k$ as primal and dual regularization, respectively.

The dual regularization parameter $\boldsymbol{\mu}_k$ proves critical for the practical performance of the method since it strikes the balance between the number of inner and outer iterations, seeking easy-to-solve sub-problems, effective warm starting, and rapid constraints satisfaction. Therefore, we carefully initialize and update $\boldsymbol{\mu}_k$, guided by the interpretation as a constraint penalization offered by the augmented

Lagrangian framework; cf. Sect. 3.1. In our implementation, we consider a vector $\boldsymbol{\mu}_k$ to gain a finer control over the constraint penalization [14]. Given a (primal) initial guess $\mathbf{x}_0 \in \mathbb{R}^n$, we initialize as in [8, Sect. 12.4]:

$$\begin{aligned}\mathbf{d}_0 &:= \mathbf{A}\mathbf{x}_0 - \Pi_{\mathcal{C}}(\mathbf{A}\mathbf{x}_0),\\ \boldsymbol{\mu}_0^i &:= \Pi_{[\mu_0^{\min},\mu_0^{\max}]}\left(\kappa_\mu \frac{\max(1,(\mathbf{d}_0^i)^2/2)}{\max(1,|f(\mathbf{x}_0)|)}\right),\ i\in[1;m],\end{aligned}$$

where $\mu_0^{\max} \geq \mu_0^{\min} > 0$ and $\kappa_\mu \geq 0$. Then, following [34, Sect. 5.3], we monitor the primal residual $\mathbf{r}_{\text{prim}}(\mathbf{v}) := \mathbf{A}\mathbf{x} - \Pi_{\mathcal{C}}(\mathbf{A}\mathbf{x}+\mathbf{y})$ from (4) and update the dual regularization parameter $\boldsymbol{\mu}_k$ accordingly. If $|\mathbf{r}^i_{\text{prim}}(v_{k+1})| > \max\left(\theta_\mu |\mathbf{r}^i_{\text{prim}}(\mathbf{v}_k)|, \varepsilon_{\text{opt}}\right)$, we set

$$\boldsymbol{\mu}^i_{k+1} = \Pi_{[\mu_{\min},\boldsymbol{\mu}^i_k]}\left(\delta_\mu \frac{\|\mathbf{r}_{\text{prim}}(\mathbf{v}_{k+1})\|_\infty}{|\mathbf{r}^i_{\text{prim}}(\mathbf{v}_{k+1})|}\boldsymbol{\mu}^i_k\right),$$

where $\theta_\mu \in (0,1)$, $\mu_{\min} > 0$, and $\delta_\mu \geq 0$. Otherwise, we set $\boldsymbol{\mu}^i_{k+1} = \boldsymbol{\mu}^i_k$. These rules adapt the constraint penalization on the current residual, seeking a uniform, steady progression towards feasibility, while making sure the sequences $\{\boldsymbol{\mu}^i_k\}$, $i \in [1;m]$, are non-increasing and bounded away from zero. In our implementation, the default values are $\mu_0^{\min} = 10^{-3}$, $\mu_0^{\max} = 10^3$, $\kappa_\mu = 0.1$, $\mu_{\min} = 10^{-9}$, $\delta_\mu = 10^{-2}$ and $\theta_\mu = 0.25$.

The primal regularization turns out to be less crucial with respect to the dual counterpart. For this reason, it is associated to a scalar value and tuned independently from the residual. Starting from $\sigma_0 > 0$, we apply

$$\sigma_{k+1} = \max(\sigma_{\min}, \kappa_\sigma \sigma_k),$$

where $\sigma_{\min} > 0$ and $\kappa_\sigma \in [0,1]$. In our implementation the default values are $\sigma_0 = 10^{-3}$, $\sigma_{\min} = 10^{-7}$, and $\kappa_\sigma = 0.1$.

Early termination The inner tolerance ε_k also affects the performance of QPDO, since it balances sub-problem accuracy and early termination. In Algorithm 1, these aspects relate to the parameters ε_0 and κ_ε, which drive $\{\varepsilon_k\}$ to zero. However, finite precision should also be taken into account. In fact, although the semismooth Newton method converges in finitely many iterations, the solution provided is exact up to round-off errors and numerical precision. Therefore, we deviate from Algorithm 1 in this respect and employ the update rule

$$\varepsilon_{k+1} = \max(\varepsilon_{\min}, \kappa_\varepsilon \varepsilon_k),$$

where $0 \leq \varepsilon_{\min} \leq \varepsilon_{\text{opt}}$. In our implementation, the default values are $\varepsilon_0 = 1$, $\kappa_\varepsilon = 0.1$, $\varepsilon_{\min} = 0.1\varepsilon_{\text{opt}}$, and $\varepsilon_{\text{opt}} = 10^{-6}$.

6.3 Infeasibility detection

A routine for detecting primal and dual infeasibility of (1) is included in Algorithm 1. This allows the algorithm to terminate with either a primal-dual solution

or a certificate of primal or dual infeasibility, for some given tolerances. We adopt the mechanism developed in [4, Sect. 5.2], which holds whenever the proximal point algorithm is employed to solve the KKT conditions (3). Problem (1) is declared primal or dual infeasible based on the conditions given in Sect. 2.5 and the vectors $\Delta\mathbf{x}_k := \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\Delta\mathbf{y}_k := \mathbf{y}_{k+1} - \mathbf{y}_k$, $k \geq 0$. As in [34], we deem the problem primal infeasible if $\Delta\mathbf{y}_k \neq \mathbf{0}$ and the following two conditions hold

$$\|\mathbf{A}^\top \Delta\mathbf{y}_k\|_\infty \leq \varepsilon_{\text{pinf}} \|\Delta\mathbf{y}_k\|_\infty, \tag{27a}$$

$$\mathbf{u}^\top \max(\mathbf{y}_k, \mathbf{0}) + \mathbf{l}^\top \min(\mathbf{y}_k, \mathbf{0}) \leq -\varepsilon_{\text{pinf}} \|\Delta\mathbf{y}_k\|_\infty, \tag{27b}$$

where $\varepsilon_{\text{pinf}} > 0$ is some tolerance level. The problem is considered dual infeasible if $\Delta\mathbf{x}_k \neq \mathbf{0}$ and the following conditions hold

$$\|\mathbf{Q}\Delta\mathbf{x}_k\|_\infty \leq \varepsilon_{\text{dinf}} \|\Delta\mathbf{x}_k\|_\infty, \tag{28a}$$

$$\mathbf{q}^\top \Delta\mathbf{x}_k \leq -\varepsilon_{\text{dinf}} \|\Delta\mathbf{x}_k\|_\infty, \tag{28b}$$

$$(\mathbf{A}\Delta\mathbf{x}_k)^i \begin{cases} \in [-\varepsilon_{\text{dinf}}, \varepsilon_{\text{dinf}}] \|\Delta\mathbf{x}_k\|_\infty & \mathbf{u}^i, \mathbf{l}^i \in \mathbb{R}, \\ \geq -\varepsilon_{\text{dinf}} \|\Delta\mathbf{x}_k\|_\infty & \mathbf{u}^i = +\infty, \\ \leq \varepsilon_{\text{dinf}} \|\Delta\mathbf{x}_k\|_\infty & \mathbf{l}^i = -\infty, \end{cases} \quad , \ i \in [1;m], \tag{28c}$$

where $\varepsilon_{\text{dinf}} > 0$ is some tolerance level. In case of primal and/or dual infeasibility, we return the vectors $\Delta\mathbf{y}_k$ and $\Delta\mathbf{x}_k$ as certificates of primal and infeasibility, respectively. In our implementation, the default values are $\varepsilon_{\text{pinf}} = \varepsilon_{\text{dinf}} = 10^{-6}$. The reader may refer to [59, Sect. 3.4], [33, Sect. V.C], and [37, Sect. 4.1], and [51, Sect. 4] for analogous applications.

6.4 Preconditioning

Preconditioning, or scaling, the problem may alleviate ill-conditioning and mitigate numerical issues, especially when the problem data span across many orders of magnitude. In our implementation, we closely follow [34, Sect. 5.2] and scale the problem data by performing the Ruiz's equilibration procedure [58] on the constraint matrix $\mathbf{A}$. This procedure iteratively scales the rows and columns of a matrix in order to make their infinity norms approach one. By default, QPDO performs 10 scaling iterations. Slightly different routines are adopted, *e.g.*, in [59, Sect. 5.1] and [51, Sect. 5.1.2]. Note that, by default, if the problem is initially scaled, the termination conditions for optimality and infeasibility refer to the original, unscaled, problem.

7 Numerical results

We discuss details of our open-source C implementation of QPDO and present computational results on random problems and the Maros-Mészáros set [39]. We test and compare QPDO against the open-source, full-fledged solvers OSQP [59] and QPALM [33, 34], and the commercial interior-point solver MOSEK [43]. Indeed, "the construction of appropriate software is by no means trivial and we wish to make a thorough job of it" [14]; we plan to improve our current implementation, in particular the linear solver discussed in Sect. 6.1, and to report comprehensive numerical results in due time.

7.1 Setup

We consider the tolerance $\varepsilon_{\text{opt}} = 10^{-5}$, and set the tolerances for all the solvers accordingly. In addition, we set the maximum run time of each solver to 100 s and no limit on the maximum number of iterations. We leave all the other settings to the internal defaults. It is worth mentioning that, since no initial guess is provided, QPDO, OSQP, and QPALM start with $\mathbf{v}_0 = \mathbf{0}$.

In general it is hard to compare the solution accuracy because the solvers may verify different termination criteria. While QPDO, QPALM and OSQP monitor the residual $\mathbf{r}$ in (4) and check the condition $\|\mathbf{r}(\mathbf{v}^\star)\|_\infty \leq \varepsilon_{\text{opt}}$, MOSEK satisfies the complementarity slackness with different metrics and scalings. Therefore, we decided not to include checks on $\|\mathbf{r}(\mathbf{v}^\star)\|_\infty$. Instead, we deem optimal a primal-dual pair $\mathbf{v}^\star$ if it is returned by a solver declaring success, otherwise we consider it a failure.

All the experiments were carried out on a desktop running Ubuntu 16.04 LTS with Intel Core i7-8700, 3.20 GHz, and 16 GB RAM. The code for all the numerical examples is available online at [16].

Metrics Let S, P, and $t_{s,p}$ denote the set of solvers, the set of problems, and the time required for solver $s \in S$ to return a solution for problem $p \in P$. The shifted geometric mean (sgm) $\widehat{t}_s$ of the run times for solver $s \in S$ on P is defined by

$$\widehat{t}_s := \exp\left(\frac{1}{|P|}\sum_{p\in P}\ln\left(t_{s,p} + t_{\text{shift}}\right)\right) - t_{\text{shift}}$$

with the shift $t_{\text{shift}} = 1$ s [41]. Here, when solver s fails to solve problem p, $t_{s,p}$ is set to the time limit. We also adopt performance profiles [18] to compare the solver timings. These plot the function $f_s^{\text{r}} : \mathbb{R} \to [0, 1]$, $s \in S$, defined by

$$f_s^{\text{r}}(\tau) := \frac{|\{p \in P : t_{s,p} \leq \tau\, t_p^{\min}\}|}{|P|}, \qquad t_p^{\min} := \min_{s\in S} t_{s,p}.$$

Considering $t_{s,p} = +\infty$ when solver s fails on problem p, $f_s^{\text{r}}(\tau)$ is the fraction of problems solved by solver s within τ times the best timing. Note that, although we cannot necessarily assess the performance of one solver relative to another with

performance profiles, they still represent a tool for evaluating and comparing the performance of a solver with respect to the best one [30].

However, performance profiles do not provide the percentage of problems that can be solved (for some given tolerance ϵ_{opt}) within a given time t. Thus, on the vein of data profiles [42, Sect. 2.2], we plot the function $f_s^{\text{a}} : \mathbb{R} \to [0,1]$, $s \in S$, defined by

$$f_s^{\text{a}}(t) := \frac{|\{p \in P : t_{s,p} \leq t\}|}{|P|}.$$

Considering $t_{s,p} = +\infty$ when solver s fails on problem p, $f_s^{\text{a}}(t)$ is the fraction of problems solved by solver s within the time t. Note that, in contrast to f_s^{r}, the time profile $t \mapsto f_s^{\text{a}}(t)$ is independent from other solvers and displayed with the actual timings of s.

7.2 Random problems

We considered QPs in the form (1) with randomly generated problem data. In each problem instance, the number of variables is $n = \lceil 10^a \rceil$, with a uniformly distributed, and ranges between 10^2 and 10^3, *i.e.*, $a \sim \mathcal{U}(2,3)$. The number of constraints is $m = \lceil b\,n \rceil$, with $b \sim \mathcal{U}(2,5)$. The linear cost is normally distributed, *i.e.*, $\mathbf{q}_i \sim \mathcal{N}(0,1)$. The cost matrix is $\mathbf{Q} = \mathbf{P}\mathbf{P}^{\mathsf{T}} + \alpha \mathbf{I}_n$, where $\mathbf{P} \in \mathbb{R}^{n \times n}$ has 10% nonzero entries $\mathbf{P}_{ij} \sim \mathcal{N}(0,1)$, and $\alpha = 10^{-6}$. The constraint matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ contains 10% nonzero entries $\mathbf{A}_{ij} \sim \mathcal{N}(0,1)$. The bounds are uniformly distributed, *i.e.*, $\mathbf{l}_i \sim \mathcal{U}(-1,0)$ and $\mathbf{u}_i \sim \mathcal{U}(0,1)$. We also investigated equality-constrained QPs. For these problems, $m = \lceil n/b \rceil$, with $b \sim \mathcal{U}(2,5)$, and $\mathbf{l} = \mathbf{u} = \mathbf{A}\tilde{\mathbf{x}}$, where $\tilde{\mathbf{x}}_i \sim \mathcal{N}(0,1)$. We generated 500 instances of each problem class.

Results Computational results are summarized in Table 1 and shown in Figs. 1, 2. Both performance and time profiles suggest that, for random QPs, QPALM exhibits the best performance, with OSQP slightly slower and QPDO third. For equality-constrained QPs, instead, OSQP performs best with QPALM and QPDO slightly behind. MOSEK is generally slower than the other solvers and, for random

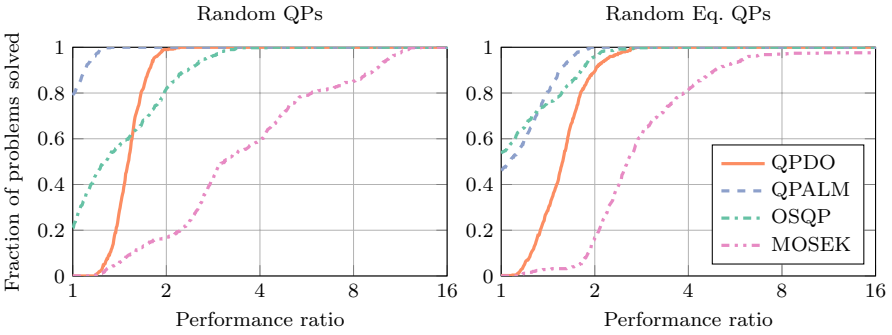

Fig. 1 Comparison on random problems with performance profiles

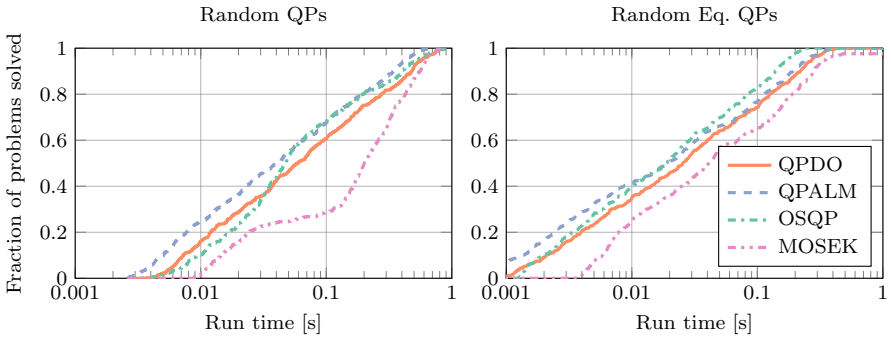

Fig. 2 Comparison on random problems with time profiles

QPs, it often declares success with a solution that does not satisfy the condition $\|\mathbf{r}(\mathbf{v}^\star)\|_\infty \leq \varepsilon_{\text{opt}}$.

7.3 Maros-Mészáros problems

We considered the Maros-Mészáros test set [39] of hard QPs and selected those with $n \leq 10^3$, due to the limitations mentioned in Sect. 6.1. This yields 73 problems, with $2 \leq n \leq 1000$, $3 \leq m \leq 1750$, and the number of nonzeros $6 \leq N \leq 22292$.

Results Computational results are summarized in Tables 1, 2 and shown in Figs. 3, 4. On this test set, QPDO demonstrates its robustness, solving all the problems. OSQP is very fast for some problems but has a high failure rate; it fails on 5 of the 20 problems reported in Table 2. As a first-order method, OSQP builds upon computationally cheap iterations, but it may take many to cope with ill-conditioning and the relatively high accuracy requirements. QPALM is still competitive but fails on the `VALUES` problem, due to linear algebra issues. MOSEK seems to perform better than the other solvers on the larger problems, but it often does not satisfy the condition $\|\mathbf{r}(\mathbf{v}^\star)\|_\infty \leq \varepsilon_{\text{opt}}$, and fails on many problems. Overall, this proves QPDO is both reliable and effective.

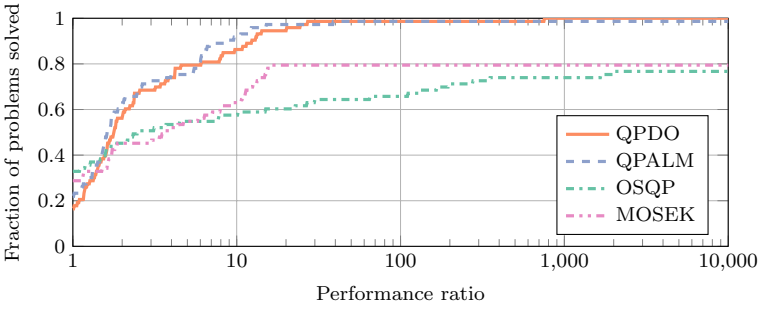

Fig. 3 Comparison on Maros-Mészáros problems with performance profiles

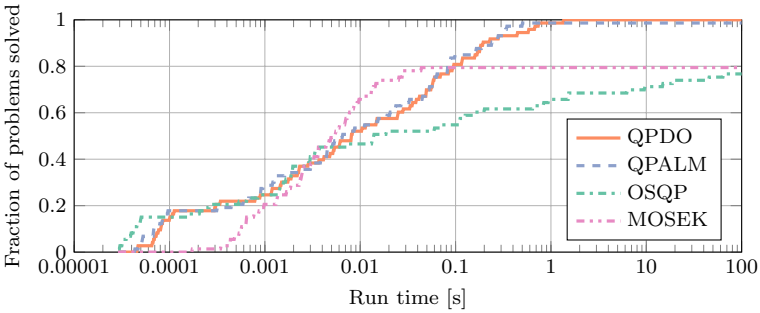

Fig. 4 Comparison on Maros-Mészáros problems with time profiles

7.4 Degenerate and infeasible problems

Consider the following parameterized QP, adapted from [37, Sect. 5.4]:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\mathbf{x}^\top \begin{bmatrix} 10 \\ 00 \end{bmatrix} \mathbf{x} + \begin{pmatrix} 1 \\ c \end{pmatrix}^\top \mathbf{x} \\ s.t. \quad & \begin{pmatrix} -\infty \\ 1 \\ 1 \end{pmatrix} \leq \begin{bmatrix} aa \\ 10 \\ 01 \end{bmatrix} \mathbf{x} \leq \begin{pmatrix} 0 \\ 3 \\ b \end{pmatrix}. \end{aligned}$$

By varying a, b and c, we can create degenerate or infeasible test problems.

First, we consider the degenerate problem obtained by setting $a = 0$, $b = 3$, and $c = 0$. This problem admits primal solutions $\mathbf{x}^\star \in \{(1, \alpha) \mid 1 \leq \alpha \leq 3\}$. Running with default settings, QPDO signals optimality after 6 proximal iterations and 14 Newton iterations, and returns $\mathbf{x} = (1.0, 1.0)$, $\mathbf{y} = (0.0, -2.0, 0.0)$, with residual $\|\mathbf{r}(\mathbf{v})\|_\infty = 1.0 \cdot 10^{-7}$.

Second, we consider a primal infeasible QP by setting $a = 1$, $b = 3$ and $c = 0$. QPDO signals primal infeasibility after 3 proximal iteration and 8 Newton iterations, and returns the certificate $\Delta\mathbf{y} = (6.6, -6.6, -6.6) \cdot 10^4$.

Finally, we consider a dual infeasible QP by setting $a = 0$, $b = +\infty$ and $c = -1$. For such problem, (0, 1) is a direction of unbounded descent. QPDO signals dual

Table 1 Comparison on different problem classes with run times, as shifted geometric means (sgm), and failure rates

		QPDO	QPALM	OSQP	MOSEK
Random QPs	Run times (sgm) [s]	0.136	0.099	0.116	0.229
	Failure rates [%]	0	0	0	0
Random Eq. QPs	Run times (sgm) [s]	0.065	0.058	0.044	0.210
	Failure rates [%]	0	0	0	2.400
Maros-Mészáros	Run times (sgm) [s]	0.074	0.119	2.993	1.594
	Failure rates [%]	0	1.370	23.288	20.548

Table 2 Comparison on the larger Maros-Mészáros problems ($N \geq 5000$) with KKT residual and run time for QPDO, QPALM, and MOSEK

Problem	n	m	N	Residual $\|\mathbf{r}(\mathbf{v})\|_\infty$			Run time [s]		
				QPDO	QPALM	MOSEK	QPDO	QPALM	MOSEK
CVXQP1M	1000	1500	9466	$1.7{\cdot}10^{-6}$	$4.4{\cdot}10^{-6}$	$1.0{\cdot}10^{-1}$	$2{\cdot}10^{-1}$	$2{\cdot}10^{-1}$	$1{\cdot}10^{-1}$
CVXQP2M	1000	1250	8717	$2.7{\cdot}10^{-6}$	$6.2{\cdot}10^{-6}$	$5.3{\cdot}10^{-1}$	$2{\cdot}10^{-1}$	$3{\cdot}10^{-1}$	$2{\cdot}10^{-1}$
CVXQP3M	1000	1750	10215	$2.0{\cdot}10^{-6}$	$1.1{\cdot}10^{-6}$	$6.7{\cdot}10^{-1}$	$3{\cdot}10^{-1}$	$3{\cdot}10^{-1}$	$2{\cdot}10^{-1}$
DUAL1	85	86	7201	$8.0{\cdot}10^{-6}$	$5.9{\cdot}10^{-7}$	$1.4{\cdot}10^{-6}$	$2{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	$4{\cdot}10^{-3}$
DUAL2	96	97	9112	$2.6{\cdot}10^{-6}$	$2.6{\cdot}10^{-6}$	$5.3{\cdot}10^{-8}$	$1{\cdot}10^{-3}$	$9{\cdot}10^{-4}$	$6{\cdot}10^{-3}$
DUAL3	111	112	12327	$1.3{\cdot}10^{-6}$	$9.7{\cdot}10^{-7}$	$4.4{\cdot}10^{-4}$	$2{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	$9{\cdot}10^{-3}$
DUAL4	75	76	5673	$2.6{\cdot}10^{-6}$	$3.6{\cdot}10^{-6}$	$8.4{\cdot}10^{-5}$	$1{\cdot}10^{-3}$	$7{\cdot}10^{-4}$	$5{\cdot}10^{-3}$
KSIP	20	1001	18871	$2.9{\cdot}10^{-6}$	$3.5{\cdot}10^{-6}$	$9.2{\cdot}10^{-6}$	$6{\cdot}10^{-3}$	$6{\cdot}10^{-3}$	$10{\cdot}10^{-3}$
PRIMAL1	325	86	6140	$6.9{\cdot}10^{-7}$	$3.6{\cdot}10^{-8}$	$4.8{\cdot}10^{-9}$	$2{\cdot}10^{-2}$	$9{\cdot}10^{-3}$	$1{\cdot}10^{-2}$
PRIMAL2	649	97	8691	$3.9{\cdot}10^{-7}$	$2.6{\cdot}10^{-6}$	$1.0{\cdot}10^{-6}$	$5{\cdot}10^{-2}$	$3{\cdot}10^{-2}$	$7{\cdot}10^{-3}$
PRIMAL3	745	112	22292	$2.0{\cdot}10^{-7}$	$7.4{\cdot}10^{-6}$	$2.1{\cdot}10^{-6}$	$2{\cdot}10^{-1}$	$8{\cdot}10^{-2}$	$2{\cdot}10^{-2}$
PRIMALC8	520	511	5182	$1.6{\cdot}10^{-7}$	$2.5{\cdot}10^{-7}$	$8.2{\cdot}10^{-8}$	$4{\cdot}10^{-2}$	$2{\cdot}10^{-2}$	$4{\cdot}10^{-3}$
QETAMACR	688	1088	11613	$1.7{\cdot}10^{-6}$	$1.3{\cdot}10^{-6}$	$2.3{\cdot}10^{-3}$	$7{\cdot}10^{-1}$	$2{\cdot}10^{-1}$	$3{\cdot}10^{-2}$
QFFFFF80	854	1378	10635	$9.8{\cdot}10^{-7}$	$8.6{\cdot}10^{-7}$	$1.5{\cdot}10^{-1}$	$6{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$7{\cdot}10^{-2}$
QFORPLAN	421	582	6112	$4.0{\cdot}10^{-6}$	$8.2{\cdot}10^{-7}$	$7.4{\cdot}10^{6}$	$7{\cdot}10^{-1}$	$3{\cdot}10^{-1}$	$3{\cdot}10^{-2}$
QGROW15	645	945	7227	$7.4{\cdot}10^{-6}$	$4.0{\cdot}10^{-7}$	$7.4{\cdot}10^{-2}$	$9{\cdot}10^{-2}$	$2{\cdot}10^{-1}$	$3{\cdot}10^{-2}$
QGROW22	946	1386	10837	$3.7{\cdot}10^{-7}$	$3.1{\cdot}10^{-6}$	$9.5{\cdot}10^{-2}$	$2{\cdot}10^{-1}$	$3{\cdot}10^{-1}$	$4{\cdot}10^{-2}$
QSCFXM2	914	1574	8285	$4.0{\cdot}10^{-6}$	$6.2{\cdot}10^{-6}$	$4.6{\cdot}10^{-1}$	$4{\cdot}10^{-1}$	$3{\cdot}10^{-1}$	$4{\cdot}10^{-2}$
QSTAIR	467	817	6287	$5.0{\cdot}10^{-6}$	$9.8{\cdot}10^{-7}$	4.5	$6{\cdot}10^{-2}$	$10{\cdot}10^{-2}$	$2{\cdot}10^{-2}$
VALUES	202	203	7846	$1.4{\cdot}10^{-6}$	NaN	NaN	$1{\cdot}10^{-2}$	$1{\cdot}10^{2}$	$4{\cdot}10^{-4}$

infeasibility after 5 proximal iterations and 12 Newton iterations, and returns the certificate $\Delta\mathbf{x} = (1.1 \cdot 10^{-5}, 1.0 \cdot 10^{7})$.

8 Conclusions

This paper presented a primal-dual Newton-type proximal method for convex quadratic programs. We build upon a simple yet crucial result: a suitable merit function for the proximal sub-problem is found in the proximal primal-dual augmented Lagrangian function. This allows us to effectively weave the proximal point method together with semismooth Newton, yielding structured symmetric linear systems, exact linesearch, and the possibility to apply sparse multi-rank factorization updates. Requiring only convexity, the method is simple and easily warm started, can exploit sparsity, is robust to early termination, and can detect infeasibility. We have implemented our method QPDO in a general-purpose solver, written in open-source C code. We benchmarked it against state-of-the-art

QP solvers, comparing run times and failure rates. QPDO proved reliable, effective, and competitive.

Appendix A: Primal-dual proximal augmented Lagrangian function

We show that the merit function $\mathcal{M}$ in (15) for the sub-problem (14) is indeed the primal-dual proximal augmented Lagrangian function, proposed and investigated in [17, 27, 54]. Let us reformulate (1) as the equivalent problem

$$\min_{\mathbf{x},\mathbf{z}} \quad f(\mathbf{x}) + g(\mathbf{z}) \qquad s.t. \quad \mathbf{A}\mathbf{x} = \mathbf{z}, \tag{29}$$

where $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^m$ are decision variables. The Lagrangian $\mathcal{L}^z$, the augmented Lagrangian $\mathcal{L}^z_\mu$, and the primal-dual augmented Lagrangian $\mathcal{M}^z_\mu$ functions for (29) are given by

$$\begin{aligned}
\mathcal{L}^z(\mathbf{x},\mathbf{z},\mathbf{y}) &:= f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{y}^\top(\mathbf{A}\mathbf{x} - \mathbf{z}),\\
\mathcal{L}^z_\mu(\mathbf{x},\mathbf{z},\mathbf{y}) &:= \mathcal{L}^z(\mathbf{x},\mathbf{z},\mathbf{y}) + \frac{1}{2\mu}\|\mathbf{A}\mathbf{x} - \mathbf{z}\|^2,\\
\mathcal{M}^z_\mu(\mathbf{x},\mathbf{z},\mathbf{y},\overline{\mathbf{y}}) &:= \mathcal{L}^z_\mu(\mathbf{x},\mathbf{z},\overline{\mathbf{y}}) + \frac{1}{2\mu}\|\mathbf{z} - \mathbf{A}\mathbf{x} + \mu(\mathbf{y} - \overline{\mathbf{y}})\|^2,
\end{aligned}$$

for some given parameter $\mu > 0$ and dual estimate $\overline{\mathbf{y}} \in \mathbb{R}^m$; cf. [7, 14] and [27, 54]. Introducing a primal proximal regularization, we define

$$\mathcal{M}^z_{\mu,\sigma}(\mathbf{x},\mathbf{z},\mathbf{y},\overline{\mathbf{x}},\overline{\mathbf{y}}) := \mathcal{M}^z_\mu(\mathbf{x},\mathbf{z},\mathbf{y},\overline{\mathbf{y}}) + \frac{\sigma}{2}\|\mathbf{x} - \overline{\mathbf{x}}\|^2 \tag{30}$$

for some given parameter $\sigma > 0$ and primal estimate $\overline{\mathbf{x}} \in \mathbb{R}^n$. In the context of primal-dual augmented Lagrangian methods, the function $\mathcal{M}^z_{\mu,\sigma}$ is to be jointly minimized with respect to $\mathbf{x}$, $\mathbf{z}$, and $\mathbf{y}$ [27, 54]. Following [17], we consider the explicit minimization over the auxiliary variable $\mathbf{z}$. The minimizer $\mathbf{z}_\mu$ of $\mathcal{M}^z_{\mu,\sigma}$ in (30) is readily obtained as

$$\begin{aligned}
\mathbf{z}_\mu(\mathbf{x},\mathbf{y},\overline{\mathbf{y}}) &:= arg\,min_{\mathbf{z}} \mathcal{M}^z_{\mu,\sigma}(\mathbf{x},\mathbf{y},\mathbf{z},\overline{\mathbf{x}},\overline{\mathbf{y}})\\
&= arg\,min_{\mathbf{z}} \left\{ g(\mathbf{z}) + \mathbf{y}^\top(\mathbf{A}\mathbf{x} - \mathbf{z}) + \frac{1}{2\mu}\|\mathbf{A}\mathbf{x} - \mathbf{z}\|^2 + \frac{1}{2\mu}\|\mathbf{z} - \mathbf{A}\mathbf{x} + \mu(\mathbf{y} - \overline{\mathbf{y}})\|^2 \right\}\\
&= arg\,min_{\mathbf{z}} \left\{ g(\mathbf{z}) + \frac{1}{2\mu}\|\mathbf{A}\mathbf{x} - \mathbf{z} + \mu\overline{\mathbf{y}}\|^2 + \frac{1}{2\mu}\|\mathbf{z} - \mathbf{A}\mathbf{x} + \mu(\mathbf{y} - \overline{\mathbf{y}})\|^2 \right\}\\
&= arg\,min_{\mathbf{z}} \left\{ g(\mathbf{z}) + \frac{1}{\mu}\|\mathbf{z} - \mathbf{A}\mathbf{x} - \mu(\overline{\mathbf{y}} - \mathbf{y}/2)\|^2 + \frac{\mu}{4}\|\mathbf{y}\|^2 - \frac{\mu}{2}\|\overline{\mathbf{y}}\|^2 \right\}\\
&= \Pi_{\mathcal{C}}\big(\mathbf{A}\mathbf{x} + \mu(\overline{\mathbf{y}} - \mathbf{y}/2)\big).
\end{aligned} \tag{31}$$

Considering $\mathcal{M}^z_{\mu,\sigma}$ on the manifold defined by $\mathbf{z}_\mu$ in (31), we get the primal-dual proximal augmented Lagrangian function $\mathcal{M}_{\mu,\sigma}$. This yields

$$\begin{aligned}\mathcal{M}_{\mu,\sigma}(\mathbf{v},\overline{\mathbf{v}}) &:= \mathcal{M}^{z}_{\mu,\sigma}(\mathbf{x},\mathbf{z}_\mu(\mathbf{x},\mathbf{y},\overline{\mathbf{y}}),\mathbf{y},\overline{\mathbf{x}},\overline{\mathbf{y}})\\ &= f(\mathbf{x}) + \frac{1}{\mu}\|\mathbf{z}_\mu(\mathbf{x},\mathbf{y},\overline{\mathbf{y}}) - \mathbf{A}\mathbf{x} - \mu(\overline{\mathbf{y}}-\mathbf{y}/2)\|^2 + \frac{\mu}{4}\|\mathbf{y}\|^2 - \frac{\mu}{2}\|\overline{\mathbf{y}}\|^2 + \frac{\sigma}{2}\|\mathbf{x}-\overline{\mathbf{x}}\|^2\\ &= f(\mathbf{x}) + \frac{1}{\mu}dist^2_C\big(\mathbf{A}\mathbf{x} + \mu(\overline{\mathbf{y}}-\mathbf{y}/2)\big) + \frac{\sigma}{2}\|\mathbf{x}-\overline{\mathbf{x}}\|^2 + \frac{\mu}{4}\|\mathbf{y}\|^2 - \frac{\mu}{2}\|\overline{\mathbf{y}}\|^2,\end{aligned}$$

which matches $\mathcal{M}_k$ in (15), up to the constant term $-\mu\|\overline{\mathbf{y}}\|^2/2$.

Appendix B: Exact linesearch coefficients

We prove that the right-hand side of (22) coincides with $\psi_k'(\tau)$ for all $\tau \in \mathbb{R}$, with the coefficients given in (23). Let $\mathbf{w}_k := \mathbf{A}\mathbf{x} + \mu_k(\mathbf{y}_k - \mathbf{y}/2)$ and $\delta\mathbf{w}_k := \mathbf{A}\delta\mathbf{x} - \mu_k\mathbf{y}/2$; cf. (23). Then, from (15), we have

$$\begin{aligned}\psi'(\tau) &= \delta\mathbf{v}^\top \nabla\mathcal{M}(\mathbf{v}+\tau\delta\mathbf{v})\\ &= \begin{pmatrix}\delta\mathbf{x}\\ \delta\mathbf{y}\end{pmatrix}^\top \begin{pmatrix}\mathbf{Q}(\mathbf{x}+\tau\delta\mathbf{x}) + \mathbf{q} + \frac{2}{\mu_k}\mathbf{A}^\top\big[\mathbf{w}_k + \tau\delta\mathbf{w}_k\\ -\Pi_C\big(\mathbf{w}_k+\tau\delta\mathbf{w}_k\big)\big] + \sigma_k(\mathbf{x}+\tau\delta\mathbf{x}-\mathbf{x}_k)\\ -\big[\mathbf{A}(\mathbf{x}+\tau\delta\mathbf{x}) + \mu_k(\mathbf{y}_k - \mathbf{y} - \tau\delta\mathbf{y}) - \Pi_C\big(\mathbf{w}_k+\tau\delta\mathbf{w}_k\big)\big]\end{pmatrix}\\ &= \delta\mathbf{x}^\top\big[\mathbf{Q}\mathbf{x} + \mathbf{q} + \sigma_k(\mathbf{x}-\mathbf{x}_k)\big] + \frac{\mu_k}{2}\delta\mathbf{y}^\top\mathbf{y} + \tau\delta\mathbf{x}^\top\big(\mathbf{Q}+\sigma_k\mathbf{I}\big)\delta\mathbf{x} + \tau\frac{\mu_k}{2}\delta\mathbf{y}^\top\delta\mathbf{y}\\ &\quad + \left[\frac{2}{\mu_k}\mathbf{A}\delta\mathbf{x} - \delta\mathbf{y}\right]^\top\big[\mathbf{w}_k + \tau\delta\mathbf{w}_k - \Pi_C\big(\mathbf{w}_k+\tau\delta\mathbf{w}_k\big)\big]\\ &= \alpha_k\tau + \beta_k + \frac{2}{\mu_k}\delta\mathbf{w}_k^\top\big[\mathbf{w}_k + \tau\delta\mathbf{w}_k - \Pi_C\big(\mathbf{w}_k+\tau\delta\mathbf{w}_k\big)\big].\end{aligned}$$

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10589-021-00342-y.

Acknowledgements This work would not have been possible without the support and enthusiasm of my supervisor, Matthias Gerdts. The author would like to thank Axel Dreves, whose comments on an early draft greatly improved the presentation. The two anonymous reviewers are thanked for critically reading the manuscript and providing comments.

Funding Open Access funding enabled and organized by Projekt DEAL. Not applicable.

Availability of data and material The datasets used, generated, and analyzed during the current study, as well as the associated code for execution and analysis, are available at https://doi.org/10.5281/zenodo.4756720.

Code availability The C code implementation of the solver is openly available on GitHub at https://github.com/aldma/qpdo.

Declaration

Competing interests The author declares that he has no conflict of interest.

References

1. Ali, A., Wong, E., Kolter, J.Z.: A semismooth Newton method for fast, generic convex programming. In: Proceedings of the 34th International Conference on Machine Learning (ICML), pp. 70–79. Sydney (2017). http://proceedings.mlr.press/v70/ali17a.html
2. Armand, P., Omheni, R.: A globally and quadratically convergent primal-dual augmented Lagrangian algorithm for equality constrained optimization. Optim. Methods Softw. **32**(1), 1–21 (2017). https://doi.org/10.1080/10556788.2015.1025401
3. Arreckx, S., Orban, D.: A regularized factorization-free method for equality-constrained optimization. SIAM J. Optim. **28**(2), 1613–1639 (2018). https://doi.org/10.1137/16M1088570
4. Banjac, G., Goulart, P., Stellato, B., Boyd, S.: Infeasibility detection in the alternating direction method of multipliers for convex optimization. J. Optim. Theory Appl. **183**(2), 490–519 (2019). https://doi.org/10.1007/s10957-019-01575-y
5. Banjac, G., Lygeros, J.: On the asymptotic behavior of the Douglas-Rachford and proximal-point algorithms for convex optimization. Optim. Lett. **15**(8), 2719–2732 (2021). https://doi.org/10.1007/s11590-021-01706-3
6. Bemporad, A.: A numerically stable solver for positive semidefinite quadratic programs based on nonnegative least squares. IEEE Trans. Autom. Control **63**(2), 525–531 (2018). https://doi.org/10.1109/TAC.2017.2735938
7. Bertsekas, D.P.: Constrained Optimization and Lagrange Multiplier Methods. Athena Scientific, Belmont (1996)
8. Birgin, E.G., Martínez, J.M.: Practical Augmented Lagrangian Methods for Constrained Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA (2014)
9. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed optimization and statistical learning via the alternating direction method of multipliers. now (2011). https://doi.org/10.1561/2200000016
10. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
11. Chen, B., Chen, X., Kanzow, C.: A penalized Fischer-Burmeister NCP-function. Math. Program. **88**(1), 211–216 (2000). https://doi.org/10.1007/PL00011375
12. Chen, Y., Davis, T.A., Hager, W.W., Rajamanickam, S.: Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. ACM Trans. Math. Softw. **35**(3), 1–14 (2008). https://doi.org/10.1145/1391989.1391995
13. Cheshmi, K., Kaufman, D.M., Kamil, S., Dehnavi, M.M.: NASOQ: numerically accurate sparsity-oriented QP solver. ACM Trans. Graph. **39**, 96 (2020)
14. Conn, A.R., Gould, N.I.M., Toint, P.L.: A globally convergent augmented Lagrangian algorithm for optimization with general constraints and simple bounds. SIAM J. Numer. Anal. **28**(2), 545–572 (1991). https://doi.org/10.1137/0728030
15. Davis, T.A., Hager, W.W.: Multiple-rank modifications of a sparse Cholesky factorization. SIAM J. Matrix Anal. Appl. **22**(4), 997–1013 (2001). https://doi.org/10.1137/S0895479899357346
16. De Marchi, A.: Benchmark examples for QPDO (2021). https://doi.org/10.5281/zenodo.4756720
17. Dhingra, N.K., Khong, S.Z., Jovanović, M.R.: The proximal augmented Lagrangian method for nonsmooth composite optimization. IEEE Trans. Autom. Control **64**(7), 2861–2868 (2019). https://doi.org/10.1109/TAC.2018.2867589
18. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002). https://doi.org/10.1007/s101070100263

19. Dontchev, A.L., Rockafellar, R.T.: Implicit Functions and Solution Mappings. Springer, Monogr. Math (2009)
20. Ferreau, H.J., Kirches, C., Potschka, A., Bock, H.G., Diehl, M.: qpOASES: a parametric active-set algorithm for quadratic programming. Math. Program. Comput. **6**(4), 327–363 (2014). https://doi.org/10.1007/s12532-014-0071-1
21. Fischer, A.: A special Newton-type optimization method. Optimization **24**, 269–284 (1992). https://doi.org/10.1080/02331939208843795
22. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval Res. Logist. Q. **3**(1–2), 95–110 (1956). https://doi.org/10.1002/nav.3800030109
23. Friedlander, M.P., Orban, D.: A primal-dual regularized interior-point method for convex quadratic programs. Math. Program. Comput. **4**(1), 71–107 (2012). https://doi.org/10.1007/s12532-012-0035-2
24. Friedlander, M.P., Tseng, P.: Exact regularization of convex programs. SIAM J. Optim. **18**(4), 1326–1350 (2008). https://doi.org/10.1137/060675320
25. Gerdts, M., Kunkel, M.: A nonsmooth Newton's method for discretized optimal control problems with state and control constraints. J. Ind. Manag. Optim. **4**(2), 247–270 (2008). https://doi.org/10.3934/jimo.2008.4.247
26. Gill, P.E., Golub, G.H., Murray, W., Saunders, M.A.: Methods for modifying matrix factorizations. Math. Comput. **28**(126), 505–535 (1974)
27. Gill, P.E., Robinson, D.P.: A primal-dual augmented Lagrangian. Comput. Optim. Appl. **51**(1), 1–25 (2012). https://doi.org/10.1007/s10589-010-9339-1
28. Gill, P.E., Robinson, D.P.: A globally convergent stabilized SQP method. SIAM J. Optim. **23**(4), 1983–2010 (2013). https://doi.org/10.1137/120882913
29. Gondzio, J.: Interior point methods 25 years later. Eur. J. Oper. Res. **218**(3), 587–601 (2012). https://doi.org/10.1016/j.ejor.2011.09.017
30. Gould, N., Scott, J.: A note on performance profiles for benchmarking software. ACM Trans. Math. Softw. (2016). https://doi.org/10.1145/2950048
31. Gould, N..I..M., Orban, D., Toint, r.P..L.: Numerical methods for large-scale nonlinear optimization. Acta Numer. **14**, 299–361 (2005). https://doi.org/10.1017/S0962492904000248
32. Gurobi Optimization Inc.: Gurobi optimizer reference manual (2021). https://www.gurobi.com/documentation/9.1/refman/refman.html. Accessed from 6 May 2021
33. Hermans, B., Themelis, A., Patrinos, P.: QPALM: a Newton-type proximal augmented Lagrangian method for quadratic programs. In: IEEE 58th Conference on Decision and Control (CDC), pp. 4325–4330. Nice, France (2019). https://doi.org/10.1109/CDC40024.2019.9030211
34. Hermans, B., Themelis, A., Patrinos, P.: QPALM: A proximal augmented Lagrangian method for nonconvex quadratic programs (2020)
35. Hintermüller, M., Ito, K., Kunisch, K.: The primal-dual active set strategy as a semismooth Newton method. SIAM J. Optim. **13**(3), 865–888 (2002). https://doi.org/10.1137/S1052623401383558
36. Izmailov, A.F., Solodov, M.V.: Newton-Type Methods for Optimization and Variational Problems. Springer, New York (2014). https://doi.org/10.1007/978-3-319-04247-3
37. Liao-McPherson, D., Kolmanovsky, I.: FBstab: a proximally stabilized semismooth algorithm for convex quadratic programming. Automatica **113**, 108801 (2020). https://doi.org/10.1016/j.automatica.2019.108801
38. Luque, F.J.: Asymptotic convergence analysis of the proximal point algorithm. SIAM J. Control Optim. **22**(2), 277–293 (1984). https://doi.org/10.1137/0322019
39. Maros, I., Mészáros, C.: A repository of convex quadratic programming problems. Optim. Methods Softw. **11**(1–4), 671–681 (1999). https://doi.org/10.1080/10556789908805768
40. Minty, G.J.: Monotone (nonlinear) operators in Hilbert space. Duke Math. J. **29**(3), 341–346 (1962). https://doi.org/10.1215/S0012-7094-62-02933-2
41. Mittelmann, H.D.: Benchmarks for optimization software. http://plato.asu.edu/bench.html. Accessed from 19 Nov 2020
42. Moré, J.J., Wild, S.M.: Benchmarking derivative-free optimization algorithms. SIAM J. Optim. **20**(1), 172–191 (2009). https://doi.org/10.1137/080724083
43. MOSEK ApS: MOSEK optimization toolbox for MATLAB. Release 9.2.42 (2021). https://docs.mosek.com/9.2/toolbox/index.html. Accessed from 5 May 2021
44. Nocedal, J., Wright, S.J.: Numerical Optimization, 2nd edn. Springer, New York, NY, USA (2006)

45. O'Connor, D., Vandenberghe, L.: Primal-dual decomposition by operator splitting and applications to image deblurring. SIAM J. Imaging Sci. **7**(3), 1724–1754 (2014). https://doi.org/10.1137/13094671X
46. O'Donoghue, B., Chu, E., Parikh, N., Boyd, S.: Conic optimization via operator splitting and homogeneous self-dual embedding. J. Optim. Theory Appl. **169**(3), 1042–1068 (2016). https://doi.org/10.1007/s10957-016-0892-3
47. Pang, J.S.: Error bounds in mathematical programming. Math. Program. **79**(1), 299–332 (1997). https://doi.org/10.1007/BF02614322
48. Parikh, N., Boyd, S.: Proximal algorithms. Found. Trends Optim. **1**(3), 127–239 (2014). https://doi.org/10.1561/2400000003
49. Patrinos, P., Bemporad, A.: An accelerated dual gradient-projection algorithm for embedded linear model predictive control. IEEE Trans. Autom. Control **59**(1), 18–33 (2014). https://doi.org/10.1109/TAC.2013.2275667
50. Pieraccini, S., Gasparo, M.G., Pasquali, A.: Global Newton-type methods and semismooth reformulations for NCP. Appl. Numer. Math. **44**(3), 367–384 (2003). https://doi.org/10.1016/S0168-9274(02)00169-1
51. Pougkakiotis, S., Gondzio, J.: An interior point-proximal method of multipliers for convex quadratic programming. Comput. Optim. Appl. (2020). https://doi.org/10.1007/s10589-020-00240-9
52. Qi, L., Jiang, H.: Semismooth Karush-Kuhn-Tucker equations and convergence analysis of Newton and quasi-Newton methods for solving these equations. Math. Oper. Res. **22**(2), 301–325 (1997). https://doi.org/10.1287/moor.22.2.301
53. Qi, L., Sun, J.: A nonsmooth version of Newton's method. Math. Program. **58**(1), 353–367 (1993). https://doi.org/10.1007/BF01581275
54. Robinson, D.P.: Primal-dual methods for nonlinear optimization. Ph.D. thesis, University of California, San Diego (2007)
55. Rockafellar, R.T.: Augmented Lagrangians and applications of the proximal point algorithm in convex programming. Math. Oper. Res. **1**(2), 97–116 (1976). https://doi.org/10.1287/moor.1.2.97
56. Rockafellar, R.T.: Monotone operators and the proximal point algorithm. SIAM J. Control Optim. **14**(5), 877–898 (1976). https://doi.org/10.1137/0314056
57. Rockafellar, R.T.: Convex Analysis. Princeton University Press, Princeton, NJ (1997)
58. Ruiz, D.: A scaling algorithm to equilibrate both rows and columns norms in matrices. Tech. Rep. RAL-TR-2001-034, Rutherford Appleton Laboratory, Oxon, UK (2001)
59. Stellato, B., Banjac, G., Goulart, P., Bemporad, A., Boyd, S.: OSQP: an operator splitting solver for quadratic programs. Math. Program. Comput. (2020). https://doi.org/10.1007/s12532-020-00179-2
60. Sun, D., Qi, L.: On NCP-functions. Comput. Optim. Appl. **13**(1), 201–220 (1999). https://doi.org/10.1023/A:1008669226453
61. Sun, J.: On piecewise quadratic Newton and trust region problems. Math. Program. **76**(3), 451–467 (1997). https://doi.org/10.1007/BF02614393
62. Themelis, A., Patrinos, P.: SuperMann: a superlinearly convergent algorithm for finding fixed points of nonexpansive operators. IEEE Trans. Autom. Control **64**(12), 4875–4890 (2019). https://doi.org/10.1109/TAC.2019.2906393
63. Vanderbei, R.J.: Symmetric quasidefinite matrices. SIAM J. Optim. **5**(1), 100–113 (1995). https://doi.org/10.1137/0805005
64. Wolfe, P.: The simplex method for quadratic programming. Econometrica **27**(3), 382–398 (1959)