An Active-Set Quadratic Programming Method Based On Sequential Hot-Starts

Travis C. Johnson[*], Christian Kirches[†], and Andreas Wächter[‡]

October 7, 2013

Abstract

A new method for solving sequences of quadratic programs (QPs) is presented. For each new QP in the sequence, the method utilizes hot-starts that employ information computed by an active-set QP solver during the solution of the first QP. This avoids the computation and factorization of the full matrices for all but the first problem in the sequence. The proposed algorithm can be seen as an extension of the iterative refinement procedure for linear systems to QP problems, coupled with the application of an accelerated linear solver method that employs hot-started QP solves as preconditioners. Local convergence results are presented. The practical performance of the proposed method is demonstrated on a sequence of QPs arising in nonlinear model predictive control and during the solution of a set of randomly generated nonlinear optimization problems using sequential quadratic programming. The results show a significant reduction in the computation time for large problems with dense constraint matrices, as well as in the number of matrix-vector products.

Keywords: nonlinear programming – quadratic programming – active set – hot starts – iterative linear solver – preconditioner – sequential quadratic programming – nonlinear model predictive control

1 Introduction

We are concerned with the solution of quadratic programs (QPs) of the form

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2}d^TWd + g^Td && \text{(1a)}\\ \text{s.t.} \quad & Ad + c = 0 && \text{(1b)}\\ & d \geq \ell, && \text{(1c)} \end{aligned}$$

where the Hessian matrix $W \in \mathbb{R}^{n\times n}$ is positive definite, $g \in \mathbb{R}^n$ is a gradient vector, and the solution $d \in \mathbb{R}^n$ is subject to equality constraints with matrix $A \in \mathbb{R}^{m\times n}$ and vector $c \in \mathbb{R}^m$ as well as lower bounds $\ell \in \mathbb{R}^n$. We assume that A has full row rank.

[*]Dept. of Eng. Sci. and App. Math., Northwestern University, Evanston, IL 60208, USA. This author is supported by a grant of the Initiative for Sustainability and Energy at Northwestern University. E-Mail: `traviscj@traviscj.com`

[†]Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Im Neuenheimer Feld 368, 69120 Heidelberg, Germany. This author was supported by the DFG Graduate School 220 (Heidelberg Graduate School of Mathematical and Computational Methods for the Sciences, HGS MathComp), funded by the German Excellency Initiative. E-Mail: `christian.kirches@iwr.uni-heidelberg.de`

[‡]Dept. of Ind. Eng. and Mgmt. Sci., Northwestern University, Evanston, IL 60208, USA. This author was supported by National Science Foundation grant DMS-1216920. Email: `andreas.waechter@northwestern.edu`

The main contribution of this paper is a novel QP algorithm that exploits information already computed during the solution of a "reference QP"

$$\min_{d\in\mathbb{R}^n} \quad \tfrac{1}{2}d^T\widetilde{W}d + \widetilde{g}^T d \tag{2a}$$
$$\text{s.t.} \quad \widetilde{A}d + \widetilde{c} = 0 \tag{2b}$$
$$d \geq \widetilde{\ell} \tag{2c}$$

in order to solve a new QP (1), where $\widetilde{W} \approx W$ and $\widetilde{A} \approx A$. Here, we assume that the reference QP (2) has been solved by a generic active-set QP solver that is capable of hot-starts. Akin to the iterative refinement procedure for linear equations, iterates converging to the optimal solution of the new QP (1) are generated by repeatedly solving (2) with different vector data $\widetilde{g}$, $\widetilde{c}$, and $\widetilde{\ell}$. In order to improve convergence speed, an accelerated linear solver method, such as SQMR [16], is applied when the active set stops changing, and the reference QP (2) then acts as a preconditioner for the linear solver method. The method is shown to converge locally if strict complementarity holds and the gradients of the active constraints are linearly independent.

The advantage of the proposed algorithm is that the factorization of the KKT matrix, which is required in the active-set QP solver and involves submatrices of $\widetilde{W}$ and $\widetilde{A}$, can be reused for solving (1). This is beneficial in particular if multiple instances of (1) need to be solved. In contrast, if the active-set solver were to be applied to solve (1) directly, a new factorization would have to be computed from scratch for each new instance with different W and A.

An additional benefit of the new QP solver is that it requires only matrix-vector products with W, A, and A^T during the solution of (1). This can lead to significant computational savings if the computation of the full matrices W and A is expensive; e.g, if the objective (1a) or the constraints (1b) involve the numerical solution of differential equations.

1.1 Motivation

This research is motivated by a number of applications where a successive resolution of QPs with similar data is required.

One example is Nonlinear Model-Predictive Control (NMPC), a numerical approach for optimally controlling a dynamic process (such as a chemical plant or a vehicle) in real-time. Here, at a given point t in time, an optimal control action is computed as the solution of a QP that is obtained from the linearization of differential equations describing the process. The initial conditions and exogenous system parameters are chosen according to the actual or estimated state of the system at time t. After a small time interval Δt, a new control action is computed, now using the initial conditions and system parameter values corresponding to $t + \Delta t$. If Δt is small and the state of the system has not changed very much, the QPs solved at t and $t + \Delta t$ are similar.

The Sequential Quadratic Programming (SQP) method for solving nonlinear programs (NLPs) represents another example. Consider an NLP of the form

$$\min_{x\in\mathbb{R}^n} \quad f(x) \tag{3a}$$
$$\text{s.t.} \quad c(x) = 0 \tag{3b}$$
$$x \geq 0, \tag{3c}$$

where the objective function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and the constraint function $c : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ with $m \leq n$ are sufficiently smooth functions (i.e., continuously differentiable). In a generic SQP method, the step d_k at an iterate x_k is obtained as the optimal solution of the QP

$$\min_{d\in\mathbb{R}^n} \quad \tfrac{1}{2}d^T W_k d + g_k^T d \tag{4a}$$
$$\text{s.t.} \quad A_k d + c_k = 0 \tag{4b}$$
$$d \geq \ell_k \tag{4c}$$

with $\ell_k := -x_k$, where $g_k = \nabla f(x_k)$ is the gradient of the objective function, $c_k = c(x_k)$ is the residual of the constraints, $A_k = \nabla c(x_k)^T$ is the Jacobian of the constraints, and W_k is (an approximation of) the Hessian of the Lagrangian function at x_k for given multiplier estimates λ_k for the equality constraints. Here, all vector and matrix data depend on the iterate x_k, and, consequently, are different at each iterate of the method. However, if the iterates are close to each other, these quantities can be expected not to change very much. This is, for example, the case, when the SQP algorithm is close to convergence.

Furthermore, suppose that we are interested in solving a sequence of NLPs (3) that differ only slightly in $f(x)$ or $c(x)$. If the optimal solution of the new NLP is close to the optimal solution of the previous one, the SQP method might require only a small number of iterations. The corresponding QPs are similar not only to each other, but also across the different nonlinear problems. In this setting, it may be beneficial to solve the QPs arising during the SQP algorithm with the algorithm proposed in this paper, where the QP from the last SQP iteration of the first NLP is taken as the reference QP (2). This can be seen as a procedure for solving a sequence of similar NLPs using hot-starts.

The solution of a sequence of closely related NLPs or QPs is also required during the execution of a branch-and-bound search for a Mixed-Integer Nonlinear Program (MINLP). Here, each node of the enumeration tree requires the solution of an NLP or QP relaxation, with different bound constraints. Moreover, during diving heuristics (see, e.g., [21] and references therein) or strong-branching (see, e.g., [1, 21]), a succession of similar NLPs or QPs has to be solved.

Structure of the Article. The remainder of this paper is organized as follows. Section 2 discusses aspects of active-set QP solvers that are relevant in the current context. In Section 3.1, we briefly review the iterative refinement procedure for solving a linear system of equations. Reinterpreting this technique in the context of equality-constrained optimization in Section 3.2, we make the connection to the use of hot-started reference QPs (2), which is then generalized in Section 3.3 to handle inequality constraints. An accelerated version is presented in Section 4 where the solution of the reference QP (2) is used as a preconditioner within an iterative linear solver method. In Section 5.1, we explore the performance of the new QP solver in the context of an optimal control application. Section 5.2 examines the performance of the new method within an SQP framework applied to sequences randomly generated NLPs with perturbed data. Final remarks are made in Section 6.

Notation. Given a vector (or vector-valued function) $x \in \mathbb{R}^n$, we denote by $x^{(i)}$ the i-th component of this vector. Given a set $\mathcal{S} \subseteq \{1,\dots,n\}$, we denote by $x^{\mathcal{S}}$ the vector composed from elements $x^{(i)}$ with indices $i \in \mathcal{S}$, and $\mathcal{S}^C$ denotes the complement of $\mathcal{S}$ in $\{1,\dots,n\}$. To simplify the notation, we write (x, y) for the concatenation $(x^T, y^T)^T$ of two vectors x and y.

2 Active-Set Solvers for Quadratic Programming

The QP algorithm proposed in this paper utilizes repeated solutions of the reference QP (2) where the matrix data $\widetilde{W}$ and $\widetilde{A}$ remains constant, and only the vector data $\widetilde{g}$, $\widetilde{c}$, and $\widetilde{\ell}$ changes. The key observation is that an active-set QP solver can often perform these repeated solutions much faster than in the case when $\widetilde{W}$ and $\widetilde{A}$ vary.

2.1 Hot Starts vs. Warm Starts

A typical active-set QP solver for (2) maintains a guess $\mathcal{A}$ of the set of variable bounds (2c) that are active at the optimal solution d_*, i.e., of $\mathcal{A}_* = \{j : d_*^{(j)} = \widetilde{\ell}^{(j)}\}$. We denote by $\mathcal{F} = \mathcal{A}^C$ the set of free variables.

An iterate of the QP solver is computed from the solution of the linear system

$$\begin{bmatrix} \widetilde{W}^{\mathcal{F}\mathcal{F}} & \widetilde{W}^{\mathcal{F}\mathcal{A}} & (\widetilde{A}^{\mathcal{F}})^T & 0 \\ \widetilde{W}^{\mathcal{A}\mathcal{F}} & \widetilde{W}^{\mathcal{A}\mathcal{A}} & (\widetilde{A}^{\mathcal{A}})^T & -I \\ \widetilde{A}^{\mathcal{F}} & \widetilde{A}^{\mathcal{A}} & 0 & 0 \\ 0 & -I & 0 & 0 \end{bmatrix} \begin{pmatrix} d^{\mathcal{F}} \\ d^{\mathcal{A}} \\ \lambda \\ \mu^{\mathcal{A}} \end{pmatrix} = - \begin{pmatrix} \widetilde{g}^{\mathcal{F}} \\ \widetilde{g}^{\mathcal{A}} \\ \widetilde{c} \\ \widetilde{\ell}^{\mathcal{A}} \end{pmatrix}. \tag{5}$$

Here, $\widetilde{W}^{\mathcal{F}\mathcal{A}}$ denotes the submatrix of $\widetilde{W}$ with rows corresponding to $\mathcal{F}$ and columns corresponding to $\mathcal{A}$ (similarly for $\widetilde{W}^{\mathcal{F}\mathcal{F}}$, $\widetilde{W}^{\mathcal{A}\mathcal{F}}$, and $\widetilde{W}^{\mathcal{A}\mathcal{A}}$), and $\widetilde{A}^{\mathcal{F}}$ is the submatrix of $\widetilde{A}$ with the columns corresponding to $\mathcal{A}$ (similarly for $\widetilde{A}^{\mathcal{A}}$). If $d^{\mathcal{F}} \geq \widetilde{\ell}^{\mathcal{F}}$ and $\mu^{\mathcal{A}} \geq 0$, the current iterate is optimal. Otherwise, $\mathcal{A}$ is updated, usually by adding or removing one variable.

Observing that $d^{\mathcal{A}} = \widetilde{\ell}^{\mathcal{A}}$, the above linear system can be reduced to

$$\begin{bmatrix} \widetilde{W}^{\mathcal{F}\mathcal{F}} & (\widetilde{A}^{\mathcal{F}})^T \\ \widetilde{A}^{\mathcal{F}} & 0 \end{bmatrix} \begin{pmatrix} d^{\mathcal{F}} \\ \lambda \end{pmatrix} = - \begin{pmatrix} \widetilde{g}^{\mathcal{F}} + \widetilde{W}^{\mathcal{F}\mathcal{A}}\widetilde{\ell}^{\mathcal{A}} \\ \widetilde{c} + \widetilde{A}^{\mathcal{A}}\widetilde{\ell}^{\mathcal{A}} \end{pmatrix}, \tag{6}$$

and the multipliers corresponding to the bound constraints are computed from the second block equation in (5),

$$\mu^{\mathcal{A}} = \widetilde{g}^{\mathcal{A}} + \widetilde{W}^{\mathcal{A}\mathcal{F}} d^{\mathcal{F}} + \widetilde{W}^{\mathcal{A}\mathcal{A}}\widetilde{\ell}^{\mathcal{A}} + (\widetilde{A}^{\mathcal{A}})^T \lambda. \tag{7}$$

Therefore, during each iteration of the active-set QP algorithm, a linear system of the form (6) has to be solved. Different methods use different techniques to solve this linear system (e.g., null space methods [13, 14], Schur complement methods [3, 17, 18]), all of which involve a factorization of matrices constructed from $\widetilde{A}^{\mathcal{F}}$ and $\widetilde{W}^{\mathcal{F}\mathcal{F}}$. To avoid large computational costs, the factorization is not computed from scratch in each iteration of the QP solver; instead, since typically only one element enters or leaves the active set $\mathcal{A}$, the factorization is updated in an efficient manner.

In this article, we say that an active-set QP solver performs a *warm start* if it uses the optimal active set $\mathcal{A}$ from a previously solved QP as the starting guess for a new QP and the internal matrix factorization to solve (6) is computed from scratch. In contrast to this, we say that a *hot start* is performed if, in addition to the active set, the internal factorization corresponding to the optimal solution of the previous QP is reused. This can only be done if the matrix data $\widetilde{A}$ and $\widetilde{W}$ remains the same, and only the vector data $\widetilde{g}$, $\widetilde{c}$, and $\widetilde{\ell}$ changes. In this case, if the new optimal active set is similar to the one from the previous QP (e.g., because $\widetilde{g}$, $\widetilde{c}$, and $\widetilde{\ell}$ have not changed much), only a few iterations of the QP solver are required, and the solution for the new QP can be obtained very quickly.

A practical QP solver has to be able to handle degeneracy, i.e., situations in which $\widetilde{A}^{\mathcal{F}}$ does not have full row rank. For this purpose, instead of constructing (5) with the set $\mathcal{A} = \{j : d^{(j)} = \widetilde{\ell}^{(j)}\}$, where d is the current QP solver iterate, it is common to use a *working set* $\mathcal{W}$ chosen as a maximal subset of $\mathcal{A}$ so that $\widetilde{A}^{(\mathcal{W}^C)}$ has full row rank. For simplicity, however, we largely assume in this paper that $\widetilde{A}^{\mathcal{F}}$ has full row rank.

2.2 Parametric QP Solvers

Hot-starting capabilities vary between different active-set QP algorithms. For example, a primal QP solver maintains feasibility of its iterates in each iteration and may have to restore feasibility first when $\widetilde{c}$ or $\widetilde{\ell}$ change. Depending on the particular method, this may require a significant amount of work. In our proposed method, we require the solution of QPs where *all* of the vector data changes, i.e., both primal and dual feasibility are destroyed.

In this context, an active-set parametric QP solver (see, e.g., [4, 10, 11]) is a suitable method. It traces optimal solutions on a homotopy path between the two QP instances. Suppose that QP (2) has been solved, with optimal solution $\widetilde{d}$ and multipliers $\widetilde{\lambda}$, $\widetilde{\mu}$, and that we now want to solve a QP with new vector data g, c, and ℓ but the same matrix data. We then consider the one-parameter family of QP problems

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2}d^T\widetilde{W}d + (\tau g + (1-\tau)\widetilde{g})^T d \\ \text{s.t.} \quad & \widetilde{A}d + \tau c + (1-\tau)\widetilde{c} = 0 \\ & d \geq \tau\ell + (1-\tau)\widetilde{\ell}, \end{aligned}$$

parametrized by $\tau \in [0,1]$. It can be shown that the optimal primal-dual solutions $z(\tau) = (d(\tau), \lambda(\tau), \mu(\tau))$ are piecewise affine-linear in τ [4]. The sets of active and inactive bound constraints, $\mathcal{A}$ and $\mathcal{F}$, are constant on each affine-linear segment. Starting with the known solution $z(0) = (\widetilde{d}, \widetilde{\lambda}, \widetilde{\mu})$, the parametric active-set algorithm generates $z(\tau)$ by performing a sequence of pivoting steps that move from the beginning of one segment to the next. At $\tau = 1$, the desired optimal solution for the new vector data is obtained.

3 Iterative Refinement

In this section, we review the iterative refinement method for linear systems and transfer the idea first to equality-constrained QPs, then to inequality-constrained QPs.

3.1 Iterative Refinement for Systems of Linear Equations

Before addressing the solution of optimization problems in Section 3.2, we first review iterative refinement for approximately solving a system of linear equations

$$Mx + b = 0. \tag{8}$$

We assume that a factorization of a reference matrix $\widetilde{M}$ is available, that operations with $\widetilde{M}^{-1}$ can hence be carried out, and that $\widetilde{M}$ is not too different from M.

After initializing $x_0 = 0$, we repeat for $i = 0, 1, 2, \ldots$

$$p_i = -\widetilde{M}^{-1}(Mx_i - b) \tag{9a}$$

$$x_{i+1} = x_i + p_i, \tag{9b}$$

until x_{i+1} is deemed to be a sufficiently good solution. After eliminating p_i and rearranging terms, we see that the recurrence (9) satisfies the fixed-point iteration

$$x_{i+1} = (I - \widetilde{M}^{-1}M)x_i - \widetilde{M}^{-1}b. \tag{10}$$

This can also be written as

$$x_{i+1} - x_* = (I - \widetilde{M}^{-1}M)(x_i - x_*), \tag{11}$$

where x_* solves (8). Therefore x_i converges to x_* if

$$\left\| I - \widetilde{M}^{-1}M \right\| < 1 \tag{12}$$

for some norm $\|\cdot\|$. For later reference, we note that we can rewrite equation (10) as

$$\widetilde{M}x_{i+1} = (\widetilde{M} - M)x_i - b. \tag{13}$$

3.2 Equality-Constrained Problems

We first discuss how the iterative refinement scheme can be applied to QPs with equality constraints only, i.e., problem (1) where (1c) is absent. In this case, the first-order optimality conditions of the QP can be stated as

$$\begin{bmatrix} W & A^T \\ A & 0 \end{bmatrix} \begin{pmatrix} d \\ \lambda \end{pmatrix} = - \begin{pmatrix} g \\ c \end{pmatrix}. \tag{14}$$

Here, and in the remainder of this paper we assume that W is positive definite, so that the solution of (14) is a global solution of the QP. Choosing M and b in (8) appropriately, the iterative refinement procedure (9) applied to (14) becomes

$$\begin{bmatrix} \widetilde{W} & \widetilde{A}^T \\ \widetilde{A} & 0 \end{bmatrix} \begin{pmatrix} p_i \\ p_i^\lambda \end{pmatrix} = - \begin{pmatrix} g \\ c \end{pmatrix} - \begin{pmatrix} Wd_i + A^T\lambda_i \\ Ad_i \end{pmatrix} \tag{15}$$

with iterates

$$d_{i+1} = d_i + p_i, \tag{16a}$$
$$\lambda_{i+1} = \lambda_i + p_i^\lambda. \tag{16b}$$

The linear system (15) states the first-order optimality conditions for the QP

$$\min_p \quad \tfrac{1}{2}p^T\widetilde{W}p + \left(g + Wd_i + A^T\lambda_i\right)^T p \tag{17a}$$
$$\text{s.t.} \quad \widetilde{A}p + (c + Ad_i) = 0. \tag{17b}$$

Therefore, p_i can be obtained equivalently as the optimal solution of this QP, and p_i^λ are the optimal multipliers for (17b). In summary, the iterative refinement procedure consists of generating iterates using the update (16), where the steps are computed as the solution of the QP (17).

We stress that the matrix data in (17), i.e., $\widetilde{W}$ and $\widetilde{A}$, remains unchanged over the iterations i, and only the vector data in the objective gradient and constraint right-hand side of this reference QP vary. Therefore, a QP solver capable of hot-starts will often be able to compute solutions for each QP (17) very rapidly, once an initial solution (requiring an internal factorization of matrices involving $\widetilde{W}$ and $\widetilde{A}$) has been computed for $i = 0$ (see Section 2).

In order to obtain a geometric interpretation of QP (17), we note that, analogously to (13), (15) and (16) can be rearranged to give

$$\begin{bmatrix} \widetilde{W} & \widetilde{A}^T \\ \widetilde{A} & 0 \end{bmatrix} \begin{pmatrix} d_{i+1} \\ \lambda_{i+1} \end{pmatrix} = - \begin{pmatrix} g \\ c \end{pmatrix} - \begin{pmatrix} (W - \widetilde{W})d_i + (A - \widetilde{A})^T\lambda_i \\ (A - \widetilde{A})d_i \end{pmatrix}, \tag{18}$$

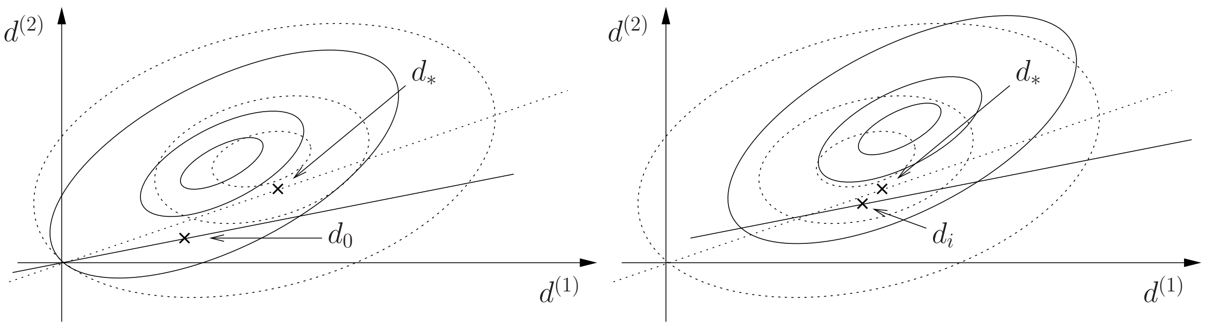


Figure 1: Geometric interpretation of iterative refinement QP

Algorithm 1 Solving QP (1) using hot-starts for QP (20) based on iterative refinement

```
1: Given: Initial iterate $(d_0, \lambda_0, \mu_0)$ with $d_0 \geq \ell$.
2: for $i = 0, 1, 2, \ldots$ do
3:     Solve QP (20).
4:     Let $p_i$ be the optimal solution of QP (20) and set $d_{i+1} = d_i + p_i$.
5:     Let $p_i^\lambda$ be the optimal multipliers for (20b) and set $\lambda_{i+1} = \lambda_i + p_i^\lambda$.
6:     Let $\mu_{i+1}$ be the optimal multipliers for (20c).
7: end for
```

or, expressed as a QP,

$$\min_{d} \quad \tfrac{1}{2} d^T \widetilde{W} d + \left(g + (W - \widetilde{W}) d_i + (A - \widetilde{A})^T \lambda_i\right)^T d \tag{19a}$$

$$\text{s.t.} \quad \widetilde{A} d + (c + (A - \widetilde{A}) d_i) = 0, \tag{19b}$$

where the optimal multipliers for (19b) are the new iterates λ_{i+1}. Here the objective gradient and constraint right-hand side are modified to compensate the error in the matrices.

This correction is depicted in Figure 1. Here, the lines and contours represent the constraints and objective; those of the original problem (1) are indicated by dashes, and the solid lines represent (19). In the left plot, which corresponds to the first iteration $i = 0$, the solution d_0 for (19) is away from the desired solution d_*. As the iteration progresses, the terms involving $W - \widetilde{W}$ and $A - \widetilde{A}$ act as corrections for the positions of the objective and constraints of the QP (19), so that, as demonstrated in the right plot, the iterates d_i approach d_*.

3.3 Inequality-Constrained Problems

To handle the general class of QPs (1), we augment QP (17) with bound constraints that ensure that the iterative refinement iterates (16a) always satisfy the original bound constraints (1c); the resulting method is formally stated as Algorithm 1.

The iterates (d_i, λ_i, μ_i) now include the bound multipliers μ_i corresponding to (1c). The optimal solution of

$$\min_{p} \quad \tfrac{1}{2} p^T \widetilde{W} p + \left(g + W d_i + A^T \lambda_i\right)^T p \tag{20a}$$

$$\text{s.t.} \quad \widetilde{A} p + (c + A d_i) = 0 \tag{20b}$$

$$d_i + p \geq \ell \tag{20c}$$

provides the step p_i, the optimal multipliers for (20b) provide the step p_i^λ, and the iterates d_{i+1} and λ_{i+1} are updated according to (16).

The iterates for the bound multipliers μ_{i+1} are updated in a different manner; they are simply set to the optimal multipliers corresponding to (20c) and, in contrast to λ_i, do not appear in the QP gradient (20a). In fact, the updates for the d_i and λ_i iterates can be performed without the knowledge of the μ_i iterates, and the algorithm can be executed without explicitly tracking μ_i. To see that this is reasonable, suppose for the moment that the active set $\mathcal{A}_*$ at the optimal solution of the original QP (1) is known, and that we apply the procedure in Section 3.2 directly to the equality-constrained QP with the original equality constraints and active bound constraints,

$$\begin{aligned} \min_{d\in\mathbb{R}^n} \quad & \tfrac{1}{2}d^TWd + g^Td && \text{(21a)}\\ \text{s.t.} \quad & Ad + c = 0 && \text{(21b)}\\ & d^{\mathcal{A}_*} = \ell^{\mathcal{A}_*}. && \text{(21c)} \end{aligned}$$

Then, the multiplier iterates λ_i in (16b) consist of the multipliers for the original constraints (1b), λ^{orig}, and the multipliers for the active bound constraints (1c), $\mu^{\mathcal{A}_*}$; i.e., $\lambda_i = (\lambda_i^{\text{orig}}, \mu_i^{\mathcal{A}_*})$. For the extended QP (21), the term involving λ_i in the right-hand side of (18) then becomes

$$\left(\begin{bmatrix} A \\ I \end{bmatrix} - \begin{bmatrix} \widetilde{A} \\ I \end{bmatrix}\right)^T \begin{pmatrix} \lambda_i^{\text{orig}} \\ \mu_i^{\mathcal{A}_*} \end{pmatrix} = (A - \widetilde{A})^T \lambda_i^{\text{orig}}.$$

Therefore, the bound multipliers $\mu_i^{\mathcal{A}_*}$ are not needed in order to compute the new iterate. Consequently, if the active set of (20) has settled to the optimal active set $\mathcal{A}_*$, it is not necessary to track the bound multipliers explicitly in order to execute the algorithm proposed in the previous section. Our algorithm uses this update strategy also when the active set of (20) may not have settled yet.

In a practical setting, a suitable termination criterion is required. In this paper, we use the semi-smooth function

$$\Phi(d,\lambda) = \left\| \begin{pmatrix} \min\{d - l, g + Wd + A^T\lambda\} \\ Ad + c \end{pmatrix} \right\|_2 . \tag{22}$$

It is easy to verify that $\Phi(d_*, \lambda_*) = 0$ if and only if (d_*, λ_*) is optimal. Note that an explicit knowledge of the bound multipliers μ is again not required. We may also use $\Phi(d,\lambda)$ as a means to monitor whether Algorithm 1 is diverging or cycling.

Before proving the main theorem of this section, we provide a lemma discussing local properties of the iterates under some regularity assumptions. Here and throughout the rest of the paper we define the active set for a given vector $d \in \mathbb{R}^n$ as $\mathcal{A}(d) := \{j : d^{(j)} = \ell^{(j)}\}$, and the corresponding set of free variables as $\mathcal{F}(d) := \mathcal{A}(d)^C$.

Lemma 1 *Suppose that $\widetilde{W}$ and W are positive definite and that QP (20) is feasible for each i (so that Algorithm 1 is well-defined). Further assume that (d_*, λ_*, μ_*) is the unique optimal primal-dual solution of QP (1) with active set $\mathcal{A}_* = \mathcal{A}(d_*)$, that strict complementarity holds, and that $\widetilde{A}^{\mathcal{F}_*}$ and $A^{\mathcal{F}_*}$ have full row rank.*

Then there exists $\epsilon > 0$ and $c_1 > 0$ so that for all i with $(d_i, \lambda_i) \in B_\epsilon := \{(d,\lambda) \mid \|(d,\lambda) - (d_*, \lambda_*)\| \le \epsilon\}$ we have $\mathcal{A}(d_{i+1}) = \mathcal{A}_*$ and $\|(d_{i+1}, \lambda_{i+1}) - (d_*, \lambda_*)\| \le c_1 \|(d_i, \lambda_i) - (d_*, \lambda_*)\|$, if (d_{i+1}, λ_{i+1}) is obtained from the updates in Steps 4 and 5 of Algorithm 1.

Proof. First note that (d_*, λ_*, μ_*) satisfies the KKT conditions for (1)

$$\begin{aligned} g + W d_* + A^T\lambda_* - \mu_* &= 0 && \text{(23a)}\\ c + A d_* &= 0 && \text{(23b)}\\ \mu_* &\geq 0 && \text{(23c)}\\ d_* &\geq \ell && \text{(23d)}\\ \mu_*^T(d_* - \ell) &= 0. && \text{(23e)} \end{aligned}$$

Considering QP (20), and following arguments similar to the derivation of (19), one can show that d_{i+1} is the solution of

$$\begin{aligned} \min_d \quad & \tfrac{1}{2} d^T \widetilde{W} d + \left(g + (W - \widetilde{W}) d_i + (A - \widetilde{A})^T \lambda_i \right)^T d && \text{(24a)}\\ \text{s.t.} \quad & \widetilde{A} d + (c + (A - \widetilde{A}) d_i) = 0 && \text{(24b)}\\ & d \geq \ell, && \text{(24c)} \end{aligned}$$

and $(\lambda_{i+1}, \mu_{i+1})$ are the corresponding optimal multipliers. The KKT conditions for this QP are given as

$$\begin{aligned} \widetilde{W} d_{i+1} + g + (W - \widetilde{W}) d_i + (A - \widetilde{A})^T \lambda_i + \widetilde{A}^T \lambda_{i+1} - \mu_{i+1} &= 0 && \text{(25a)}\\ \widetilde{A} d_{i+1} + c + (A - \widetilde{A}) d_i &= 0 && \text{(25b)}\\ \mu_{i+1} &\geq 0 && \text{(25c)}\\ d_{i+1} &\geq \ell && \text{(25d)}\\ (\mu_{i+1})^T (d_{i+1} - \ell) &= 0. && \text{(25e)} \end{aligned}$$

Suppose for the moment that $d_i = d_*$ and $\lambda_i = \lambda_*$, and substitute (23a) and (23b) into (25a) and (25b). Rearranging terms one can verify that then $(d_{i+1}, \lambda_{i+1}, \mu_{i+1}) = (d_*, \lambda_*, \mu_*)$ satisfies the KKT conditions (25) and is therefore the optimal solution of (24). Since (24c) and (1c) are identical, the active set for (24) is $\mathcal{A}_*$.

Because of the full row rank of $\widetilde{A}^{\mathcal{F}_*}$, the linear independence constraint qualification (LICQ) holds at the solution of (24) if $(d_i, \lambda_i) = (d_*, \lambda_*)$. Since strict complementarity also holds and $\widetilde{W}$ is positive definite by assumption, standard sensitivity results hold (see, e.g., [12]), and there exists $\epsilon > 0$ so that for $(d_i, \lambda_i) \in B_\epsilon$ (i.e., the gradient in (24a) and the constant term in (24b) vary sufficiently little), the active set of (24) does not change and is identical to that of QP (1). Hence $\mathcal{A}(d_{i+1}) = \mathcal{A}_*$.

Let $(d_i, \lambda_i) \in B_\epsilon$. Noting that $(\mu_{i+1})^{\mathcal{F}_*} = 0$ and $d_{i+1}^{\mathcal{A}_*} = d_*^{\mathcal{A}_*}$, substituting again (23a) and (23b) into (25a) and (25b), and rearranging terms, it can be seen that

$$\begin{bmatrix} \widetilde{W}^{\mathcal{F}_*\mathcal{F}_*} & \widetilde{W}^{\mathcal{F}_*\mathcal{A}_*} & (\widetilde{A}^{\mathcal{F}_*})^T & 0 \\ \widetilde{W}^{\mathcal{A}_*\mathcal{F}_*} & \widetilde{W}^{\mathcal{A}_*\mathcal{A}_*} & (\widetilde{A}^{\mathcal{A}_*})^T & -I \\ \widetilde{A}^{\mathcal{F}_*} & \widetilde{A}^{\mathcal{A}_*} & 0 & 0 \\ 0 & -I & 0 & 0 \end{bmatrix} \begin{pmatrix} d_{i+1}^{\mathcal{F}_*} - d_*^{\mathcal{F}_*} \\ d_{i+1}^{\mathcal{A}_*} - d_*^{\mathcal{A}_*} \\ \lambda_{i+1} - \lambda_* \\ (\mu_{i+1})^{\mathcal{A}_*} - (\mu_*)^{\mathcal{A}_*} \end{pmatrix} = - \begin{pmatrix} \hat{g}_i^{\mathcal{F}_*} \\ \hat{g}_i^{\mathcal{A}_*} \\ (A - \widetilde{A})^T (d_i - d_*) \\ 0 \end{pmatrix}, \qquad (26)$$

where $\hat{g}_i = (W - \widetilde{W})(d_i - d_*) + (A - \widetilde{A})^T(\lambda_i - \lambda_*)$. Since $\widetilde{W}$ is positive definite and $\widetilde{A}^{\mathcal{F}_*}$ has full row rank, the matrix $\widetilde{K}_{\mathcal{A}}$ in this linear system is non-singular. The claim then follows with $c_1 = \|I - \widetilde{K}_{\mathcal{A}}^{-1} K_{\mathcal{A}}\|$, where $K_{\mathcal{A}}$ is the matrix in (26) with $\widetilde{W}$ and $\widetilde{A}$ replaced by W and A, respectively. $\square$

The next theorem shows that Algorithm 1 has desirable convergence properties. In particular, the method cannot converge to spurious solutions, and local convergence is guaranteed under regularity assumptions if the matrix data is not too different.

Theorem 1 *Suppose that $\widetilde{W}$ and W are positive definite and that QP (20) is feasible for each i. Then the following statements hold true for the sequence (d_i, λ_i, μ_i) generated by Algorithm 1.*

i) If (d_i, λ_i) converges to (d_*, λ_*), then d_* is the unique optimal solution of QP (1), and λ_* are optimal multipliers.

ii) Suppose that the assumptions of Lemma 1 hold, and that in addition

$$c_2 := \left\| I - \begin{bmatrix} \widetilde{W}^{\mathcal{F}_*\mathcal{F}_*} & (\widetilde{A}^{\mathcal{F}_*})^T \\ \widetilde{A}^{\mathcal{F}_*} & 0 \end{bmatrix}^{-1} \begin{bmatrix} W^{\mathcal{F}_*\mathcal{F}_*} & (A^{\mathcal{F}_*})^T \\ A^{\mathcal{F}_*} & 0 \end{bmatrix} \right\| < 1. \tag{27}$$

Then, if (d_i, λ_i) is sufficiently close to (d_*, λ_*), the sequence generated by Algorithm 1 converges to (d_*, λ_*, μ_*).

Proof. To *i)*: Recall that the iterates satisfy the optimality condition (25) of the QP (24). Because (d_i, λ_i) converges to (d_*, λ_*), we have that μ_i is bounded. Consequently, there is a subsequence μ_{i_j} that converges to μ_*. Taking the limit in (25) as $i_j \to \infty$, we see that the limit point satisfies the optimality conditions (23). Hence, d_* is an optimal solution of QP (1) with optimal multipliers λ_* and μ_*. Uniqueness of d_* follows from the positive-definiteness of W.

To *ii)*: Let $\epsilon > 0$ and $c_1 > 0$ be the constants from Lemma 1. Set $\tilde{\epsilon} = \epsilon / \max\{1, c_1\}$ and let $(d_i, \lambda_i) \in B_{\tilde{\epsilon}} \subseteq B_\epsilon$. Then $\mathcal{A}(d_{i+1}) = \mathcal{A}_*$ and $(d_{i+1}, \lambda_{i+1}) \in B_{c_1\tilde{\epsilon}} \subseteq B_\epsilon$ by Lemma 1. Applying this argument a second time, it follows $\mathcal{A}(d_{i+2}) = \mathcal{A}_*$.

Therefore, (26) holds with i replaced by $i+1$, and $d_{i+2}^{\mathcal{A}_*} = d_{i+1}^{\mathcal{A}_*} = d_*^{\mathcal{A}_*}$. Because then $d_{i+2}^{\mathcal{A}_*} - d_*^{\mathcal{A}_*} = d_{i+1}^{\mathcal{A}_*} - d_*^{\mathcal{A}_*} = 0$, (26) can be reduced to

$$\begin{bmatrix} \widetilde{W}^{\mathcal{F}_*\mathcal{F}_*} & (\widetilde{A}^{\mathcal{F}_*})^T \\ \widetilde{A}^{\mathcal{F}_*} & 0 \end{bmatrix} \begin{pmatrix} d_{i+2}^{\mathcal{F}_*} - d_*^{\mathcal{F}_*} \\ \lambda_{i+2} - \lambda_* \end{pmatrix} = - \left(\begin{bmatrix} W^{\mathcal{F}_*\mathcal{F}_*} & (A^{\mathcal{F}_*})^T \\ A^{\mathcal{F}_*} & 0 \end{bmatrix} - \begin{bmatrix} \widetilde{W}^{\mathcal{F}_*\mathcal{F}_*} & (\widetilde{A}^{\mathcal{F}_*})^T \\ \widetilde{A}^{\mathcal{F}_*} & 0 \end{bmatrix} \right) \begin{pmatrix} d_{i+1}^{\mathcal{F}_*} - d_*^{\mathcal{F}_*} \\ \lambda_{i+1} - \lambda_* \end{pmatrix}.$$

Using (27) we obtain $\|(d_{i+2}, \lambda_{i+2}) - (d_*, \lambda_*)\| \le c_2 \|(d_{i+1}, \lambda_{i+1}) - (d_*, \lambda_*)\|$. Because $c_2 < 1$, also the new iterate (d_{i+2}, λ_{i+2}) lies in B_ϵ. Repeating this argument, we see that $(d_j, \lambda_j) \in B_\epsilon$ for all $j \ge i+1$, and we obtain $(d_j, \lambda_j) \to (d_*, \lambda_*)$. $\square$

4 Acceleration By Preconditioned Linear Solver

Let us assume for the moment that the optimal active set $\mathcal{A}_*$ is known. In that case, we seek the solution of the linear system

$$\underbrace{\begin{bmatrix} W^{\mathcal{F}\mathcal{F}} & (A^{\mathcal{F}})^T \\ A^{\mathcal{F}} & 0 \end{bmatrix}}_{=:\mathcal{M}} \begin{pmatrix} d^{\mathcal{F}} \\ \lambda \end{pmatrix} = - \begin{pmatrix} g + W^{\mathcal{F}\mathcal{A}} \ell^{\mathcal{A}} \\ c + A^{\mathcal{A}} \ell^{\mathcal{A}} \end{pmatrix} \tag{28}$$

with $\mathcal{A} = \mathcal{A}_*$, $\mathcal{F} = \mathcal{A}^C$, and set $d^{\mathcal{A}} = \ell^{\mathcal{A}}$ (see also (6)). The disadvantage of the fixed-point iteration (9), which is the basis of the algorithm proposed so far, lies in its potentially slow linear convergence rate. To speed up the convergence, we may use some accelerated iterative linear solver method, such as, for example, GMRES [23], LSQR [22], LSMR [15], or SQMR [16]. Note that these methods converge independent of some contraction condition (cf. (12)), but for good practical performance, they require the application of a preconditioner. In our case, the preconditioner replaces W and A in (28) by $\widetilde{W}$ and $\widetilde{A}$, respectively, and its application requires the solution of

$$\begin{bmatrix} \widetilde{W}^{\mathcal{F}\mathcal{F}} & (\widetilde{A}^{\mathcal{F}})^T \\ \widetilde{A}^{\mathcal{F}} & 0 \end{bmatrix} \begin{pmatrix} z_j^{\mathcal{F}} \\ z_j^{\lambda} \end{pmatrix} = \begin{pmatrix} r_j \\ s_j \end{pmatrix} \tag{29}$$

during iteration j of the iterative linear solver. In our context, the precise definition of the right-hand side r_j and s_j is not important; these vectors are defined by the specific iterative linear solver method. The solution $(z_j^{\mathcal{F}}, z_j^{\lambda})$ provides preconditioned quantities required by the particular linear solver. As before we make the key observation that the solution of (29) can be obtained equivalently by solving the QP

$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z^T \widetilde{W} z - r_j^T z^{\mathcal{F}} && \text{(30a)} \\ \text{s.t.} \quad & \widetilde{A} z - s_j = 0 && \text{(30b)} \\ & z^{\mathcal{A}} = 0. && \text{(30c)} \end{aligned}$$

However, the optimal active set $\mathcal{A}_*$ is not known in advance. In our algorithm, we start the iterative linear solver for (28) if the active set no longer changes in Algorithm 1 because we might have found the optimal active set. Still, it is important to detect non-optimal active sets. For this purpose, we modify the previous QP, and instead solve

$$\begin{aligned} \min_z \quad & \tfrac{1}{2} z^T \widetilde{W} z - r_j^T z^{\mathcal{F}} + \left(g^{\mathcal{A}} + W^{\mathcal{A}\mathcal{F}} \bar{d}_j^{\mathcal{F}} + W^{\mathcal{A}\mathcal{A}} \ell^{\mathcal{A}} + (A^{\mathcal{A}})^T \bar{\lambda}_j \right)^T z^{\mathcal{A}} && \text{(31a)} \\ \text{s.t.} \quad & \widetilde{A} z - s_j = 0 && \text{(31b)} \\ & z^{\mathcal{A}} \geq 0 && \text{(31c)} \end{aligned}$$

to obtain the preconditioned quantities $z^{\mathcal{F}}$ and z^{λ} for the iterative linear solver. Here, $(\bar{d}_j^{\mathcal{F}}, \bar{\lambda}_j)$ are the iterates of the linear solver for the system (28). Note that this QP can be solved using a hot-start for the reference QP (2). If $z^{\mathcal{A}} \neq 0$, we take this as an indication that too many variables might have been considered active. Theorem 2 below shows this strategy indeed prevents convergence to a point with incorrect active set.

Algorithm 2 details the overall method. In an outer loop (indexed by i), the algorithm applies the iterative refinement steps (Steps 4 and 25) in the same way as in Algorithm 1. Recall that the iterates μ_i for the bound multipliers are not required to execute the algorithm and are omitted here for simplicity. During this procedure, the method keeps track of the active set $\mathcal{A}_i$. If the active set is identical in two consecutive iterations and the most recent step was an iterative refinement step (i.e., `ref_flag` = `true`), the algorithm starts the accelerated iterative linear solver (in an inner loop, Steps 10–23) from the current outer iterate in order to solve (28) for the current active set. Here, applications of the preconditioner are performed by solving the QP (30). The iterative linear solver method is interrupted if either $z^{\mathcal{A}} \neq 0$ (indicating that the active set might be too large) or if the inner iterate violates the (ignored) bounds on the free variables (indicating that the active set might be too small). In either case, the algorithm reverts to the outer iterative refinement procedure from the most recent feasible iterate of the inner iterative linear solver method. Note that the details of the iterative linear solver computations in Steps 11 and 16 are left vague, since they depend on the particular method. For concreteness, Appendix A provides an explicit version of Algorithm 2 for the SQMR linear solver method.

Algorithm 2 (`iQP`) Solving QP (1) using hot-starts for QPs (20) and (31), accelerated version

```
1: Given: Initial iterates $d_0 \geq \ell$ and $\lambda_0$.
2: Initialize: $\mathcal{A}_{-1} = \{l \mid d_0^{(l)} = \ell^{(l)}\}$ and ref_flag $\leftarrow$ false.
3: for $i = 0, 1, 2, \dots$ do
4:   Solve QP (20) to obtain optimal solution $p_i$ with optimal multipliers $p_i^\lambda$.
5:   Determine the active set $\mathcal{A}_i = \{\, l \mid d_i^{(l)} + p_i^{(l)} = \ell^{(l)}\}$.
6:   if $\mathcal{A}_i = \mathcal{A}_{i-1}$ and ref_flag $=$ true then
7:     Set ref_flag $\leftarrow$ false.
8:     Fix active set $\mathcal{A} \leftarrow \mathcal{A}_i$ and $\mathcal{F} \leftarrow \mathcal{A}^C$.
9:     Initialize iterative linear solver for solving (28) with iterate $(\bar{d}_0^{\mathcal{F}}, \bar{\lambda}_0) \leftarrow (d_i^{\mathcal{F}}, \lambda_i)$.
10:    for $j = 0, 1, 2, \dots$ do
11:      Perform iterative linear solver computations up to application of preconditioner.
12:      Apply preconditioner by solving QP (31).
13:      if $z_j^{\mathcal{A}} \neq 0$ in optimal solution of QP (31) then
14:        Update outer iterate $(d_{i+1}^{\mathcal{F}}, d_{i+1}^{\mathcal{A}}, \lambda_{i+1}) \leftarrow (\bar{d}_j^{\mathcal{F}}, \ell^{\mathcal{A}}, \bar{\lambda}_j)$; break.
15:      end if
16:      Continue to perform iterative linear solver computations to obtain $(\bar{d}_{j+1}^{\mathcal{F}}, \bar{\lambda}_{j+1})$.
17:      if $\bar{d}_{j+1}^{\mathcal{F}} \ngeq \ell^{\mathcal{F}}$ then
18:        Update outer iterate $(d_{i+1}^{\mathcal{F}}, d_{i+1}^{\mathcal{A}}, \lambda_{i+1}) \leftarrow (\bar{d}_j^{\mathcal{F}}, \ell^{\mathcal{A}}, \bar{\lambda}_j)$; break.
19:      end if
20:      if $(\bar{d}_{j+1}^{\mathcal{F}}, \bar{\lambda}_{j+1})$ solves (28) then
21:        Return optimal solution $(d_*, \bar{\lambda}_{j+1})$ with $d_*^{\mathcal{F}} = \bar{d}_{j+1}^{\mathcal{F}}$ and $d_*^{\mathcal{A}} = \ell^{\mathcal{A}}$.
22:      end if
23:    end for
24:  else
25:    Update $d_{i+1} = d_i + p_i$ and $\lambda_{i+1} = \lambda_i + p_i^\lambda$.
26:    Set ref_flag $\leftarrow$ true.
27:  end if
28:  if $(d_{i+1}, \lambda_{i+1})$ solves (1) then
29:    Return optimal solution $(d_{i+1}, \lambda_{i+1})$.
30:  end if
31: end for
```

The flag `ref_flag` is included for two reasons: First, it ensures that at least one iterative refinement step is taken between two executions of the iterative linear solver. In this way, the initial iterate in the second execution is different from the final iterate in the first execution. This prevents cycling in case the iterative linear solver is interrupted in its first iteration without taking any step.

Secondly, if $\mathcal{A}_i = \mathcal{A}_{i-1}$ and the most recent iteration was an iterative refinement step (Step 25), it is guaranteed that $p_i^{\mathcal{A}_i} = 0$. This is desirable if p_i from (20) happens to be identical to the solution z of the preconditioning QP (31) in the first iteration of the iterative linear solver methods. This is the case for SQMR, and therefore a renewed solution of the iterative refinement QP in Step 12 of the first SQMR iteration can then be skipped.

Finally, we note that, if the iterative linear solver is interrupted before having taken a step in outer iteration i, we have $(d_{i+1}, \lambda_{i+1}) = (d_i, \lambda_i)$ and the iterative refinement QP solution (p_i, p_i^λ) can be used in the next iteration $i + 1$ without solving QP (20) again.

Before stating the convergence properties of this method, we make the following assumptions on the iterative linear solver.

Assumptions 1 *Assume that the iterative linear solver in the inner loop of Algorithm 2 (Steps 9–12, 16, and 23) has the following properties:*

i) If the iterates generated by the linear solver method converge and the matrix in (29) is nonsingular, the limit point of the iterates satisfies the linear system (28) and the quantities $z_j^{\mathcal{F}}$, z_j^{λ}, r_j, and s_j in the preconditioning system (29) converge to zero.

ii) If the matrices in (28) and (29) are nonsingular, the iterates generated by the linear solver converge to the solution of (28) from any starting point.

iii) If the matrices in (28) and (29) are nonsingular, there exists a constant $c_3 > 0$ so that

$$\max\left\{\|z_j^{\mathcal{F}}\|, \|z_j^{\lambda}\|, \|r_j\|, \|s_j\|, \left\|\begin{pmatrix}\bar{d}_j^{\mathcal{F}}\\ \bar{\lambda}_j\end{pmatrix} - \begin{pmatrix}\bar{d}_*^{\mathcal{F}}\\ \bar{\lambda}_*\end{pmatrix}\right\|\right\} \leq c_3 \left\|\begin{pmatrix}\bar{d}_0^{\mathcal{F}}\\ \bar{\lambda}_0\end{pmatrix} - \begin{pmatrix}\bar{d}_*^{\mathcal{F}}\\ \bar{\lambda}_*\end{pmatrix}\right\| \tag{32}$$

for all $\bar{d}_0^{\mathcal{F}}$ and $\bar{\lambda}_0$, and for all j, where $(\bar{d}_*^{\mathcal{F}}, \bar{\lambda}_*)$ is the solution of (28).

We point out that the LSQR [22] and LSMR [15] solvers satisfy these assumptions, as proved in [15].

In the description of the algorithm, we implicitly assumed that the matrices in (28) and (29) are nonsingular. However, it is possible that the solution of (20) is degenerate and that the gradients of the active constraints are linearly dependent. In this case, the active set $\mathcal{A}$ identified in Step 5 leads to a preconditioning system (29) with a singular matrix because $A^{\mathcal{F}}$ does not have full row rank. Nevertheless, as mentioned at the end of Section 2.1, usually active-set QP solvers maintain a "working set" $\mathcal{W}$ of linearly independent constraints that identify the optimal solution. Using this working set in place of the active set $\mathcal{A}$ in Step 5, we are guaranteed to always obtain a nonsingular preconditioning system (29) if $\mathcal{F} = \mathcal{W}^C$. In addition, if we assume that the QP solver returns the same working set whenever the active set has not changed during a hot-start for (20), Algorithm 2 will still detect when the active set remains unchanged and enter the inner loop in Step 6. Therefore, the nonsingularity Assumption 1*i)* is not restrictive if we consistently replace $\mathcal{A}$ by a working set.

The following theorem shows that Algorithm 2 cannot converge to spurious solutions and that local convergence is guaranteed under certain regularity assumptions.

Theorem 2 *Suppose that Assumption 1 holds, that $\widetilde{W}$ and W are positive definite, that QP (20) and QP (31) are always feasible (so that Algorithm 2 is well-defined), and that the matrix in (29) is nonsingular whenever the preconditioning QP (31) is solved in Step 12. Furthermore assume that the algorithm does not terminate at an optimal solution in Step 21 or Step 29.*

i) If Step 25 is executed infinitely many times and the sequence (d_i, λ_i) converges to some limit point (d_*, λ_*), then d_* is the unique optimal solution of QP (1).

ii) If Step 25 is executed finitely many times, the algorithm eventually stays in the inner iterative linear solver loop (Steps 10–23) for some active set $\mathcal{A}$. If the corresponding sequence of iterative linear solver iterates $(\bar{d}_j^{\mathcal{F}}, \bar{\lambda}_j)$ converges to some limit point $(\bar{d}_*^{\mathcal{F}}, \bar{\lambda}_*)$, then d_* defined by $d_*^{\mathcal{F}} = \bar{d}_*^{\mathcal{F}}$ and $d_*^{\mathcal{A}} = \ell^{\mathcal{A}}$ is the unique optimal solution of QP (1) with corresponding optimal multipliers $\lambda_* = \bar{\lambda}_*$.

iii) Suppose that (d_*, λ_*, μ_*) is the unique optimal primal-dual solution of QP (1) with active set $\mathcal{A}_* = \mathcal{A}(d_*)$, that strict complementarity holds, and that $\widetilde{A}^{\mathcal{F}_*}$ and $A^{\mathcal{F}_*}$ have full row rank. Then, if (d_0, λ_0) is sufficiently close to (d_*, λ_*), Algorithm 2 eventually remains in the inner iterative linear solver loop with active set $\mathcal{A} = \mathcal{A}_*$, and the inner iterates $(\bar{d}_j^{\mathcal{F}}, \bar{\lambda}_j)$ converge to $(d_*^{\mathcal{F}_*}, \lambda_*)$.

Proof. To *i)*: If Algorithm 2 executes the iterative refinement update in Steps 4 and 25 infinitely many times, we can argue as in the proof of Theorem 1*i)* that the limit point (d_*, λ_*) satisfies the optimality conditions for QP (1).

To *ii)*: If Step 25 is executed only a finite number of times, the algorithm eventually performs only iterative linear solver iterations in the inner j-loop with some fixed active set $\mathcal{A}$. Because Step 18 is not reached, the test in Step 17 is not true, and therefore $\bar{d}_j^{\mathcal{F}} \geq \ell^{\mathcal{F}}$ for all j, so that in the limit $\bar{d}_*^{\mathcal{F}} \geq \ell^{\mathcal{F}}$ and hence by construction $d_* \geq \ell$. Similarly, because Step 14 is not reached, we have $z_j^{\mathcal{A}} = 0$ for all j.

Furthermore, the preconditioning QP (31) is solved in each iteration, and its optimality conditions imply

$$\begin{bmatrix} \widetilde{W}^{\mathcal{F}\mathcal{F}} & \widetilde{W}^{\mathcal{F}\mathcal{A}} & (\widetilde{A}^{\mathcal{F}})^T & 0 \\ \widetilde{W}^{\mathcal{A}\mathcal{F}} & \widetilde{W}^{\mathcal{A}\mathcal{A}} & (\widetilde{A}^{\mathcal{A}})^T & -I \\ \widetilde{A}^{\mathcal{F}} & \widetilde{A}^{\mathcal{A}} & 0 & 0 \\ 0 & -I & 0 & 0 \end{bmatrix} \begin{pmatrix} z_j^{\mathcal{F}} \\ z_j^{\mathcal{A}} \\ z_j^{\lambda} \\ \mu_j^{\mathcal{A}} \end{pmatrix} = \begin{pmatrix} r_j \\ -\left(g^{\mathcal{A}} + W^{\mathcal{A}\mathcal{F}}\bar{d}_j^{\mathcal{F}} + W^{\mathcal{A}\mathcal{A}}\ell^{\mathcal{A}} + (A^{\mathcal{A}})^T\bar{\lambda}_j\right) \\ s_j \\ 0 \end{pmatrix}, \quad (33)$$

where $\mu_j^{\mathcal{A}} \geq 0$ are the multipliers for the bound constraints (31c). Because we assume that the iterates converge, it follows from Assumption 1*i)* that $(z_j^{\mathcal{F}}, z_j^{\lambda})$ and residuals (r_j, s_j) converge to zero, and the optimality conditions above become in the limit

$$\begin{bmatrix} \widetilde{W}^{\mathcal{F}\mathcal{F}} & \widetilde{W}^{\mathcal{F}\mathcal{A}} & (\widetilde{A}^{\mathcal{F}})^T & 0 \\ \widetilde{W}^{\mathcal{A}\mathcal{F}} & \widetilde{W}^{\mathcal{A}\mathcal{A}} & (\widetilde{A}^{\mathcal{A}})^T & -I \\ \widetilde{A}^{\mathcal{F}} & \widetilde{A}^{\mathcal{A}} & 0 & 0 \\ 0 & -I & 0 & 0 \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ \mu_*^{\mathcal{A}} \end{pmatrix} = \begin{pmatrix} 0 \\ -\left(g^{\mathcal{A}} + W^{\mathcal{A}\mathcal{F}}\bar{d}_*^{\mathcal{F}} + W^{\mathcal{A}\mathcal{A}}\ell^{\mathcal{A}} + (A^{\mathcal{A}})^T\bar{\lambda}_*\right) \\ 0 \\ 0 \end{pmatrix} \quad (34)$$

with $\mu_*^{\mathcal{A}} \geq 0$.

Finally, because we assume that the iterates converge, it follows from Assumption 1*i)* that $(\bar{d}_*^{\mathcal{F}}, \bar{\lambda}_*)$ satisfy (28). Together with (34) this implies that the optimality conditions (23) for QP (1) are satisfied by $(d_*, \bar{\lambda}_*)$.

To *iii)*: Note that $(\bar{d}_*^{\mathcal{F}}, \bar{\lambda}_*) = (d_*^{\mathcal{F}_*}, \lambda_*)$ is the solution of (28). Due to strict complementarity, we have $\bar{d}_*^{\mathcal{F}} > \ell^{\mathcal{F}_*}$, and from Assumption 1*iii)* we then have that $\bar{d}_j^{\mathcal{F}} \geq \ell_k^{\mathcal{F}_*}$ for all j if $(\bar{d}_0^{\mathcal{F}}, \bar{\lambda}_0)$ is sufficiently close to $(\bar{d}_*^{\mathcal{F}}, \bar{\lambda}_*)$. Similarly, $\mu_*^{\mathcal{A}_*} = g^{\mathcal{A}_*} + W^{\mathcal{A}_*\mathcal{F}_*}d_*^{\mathcal{F}_*} + W^{\mathcal{A}_*\mathcal{A}_*}\ell^{\mathcal{A}_*} + (A^{\mathcal{A}_*})^T\lambda_* > 0$.

Because $\widetilde{W}$ is positive definite and $\widetilde{A}^{\mathcal{F}}$ has full row rank, the matrix in (33) is nonsingular. Therefore, we then have from (33) and (32) that $\mu_j^{\mathcal{A}_*} > 0$ for all j if $(\bar{d}_0^{\mathcal{F}}, \bar{\lambda}_0)$ is sufficiently close to $(\bar{d}_*^{\mathcal{F}}, \bar{\lambda}_*)$, and due to complementarity, $z_j^{\mathcal{A}_*} = 0$ for such j. In summary, we conclude that Algorithm 2 remains in the inner iterative linear solver loop if it is invoked at an iterate $(d_i, \lambda_i) \in B_{\epsilon_1}$ for a sufficiently small $\epsilon_1 > 0$ in Step 9, because then the conditions in Steps 13 and 17 will never be met.

Let $\epsilon > 0$ and $c_1 > 0$ be the constants from Lemma 1. Set $\epsilon_2 = \min\{\epsilon, \epsilon_1\}/\max\{1, c_1\}$ and let $(d_0, \lambda_0) \in B_{\epsilon_2} \subseteq B_\epsilon$. Then $\mathcal{A}(d_1) = \mathcal{A}_*$ and $(d_1, \lambda_1) \in B_{c_1\epsilon_2} \subseteq B_\epsilon$ by Lemma 1. Applying this argument a second time, it follows $\mathcal{A}(d_2) = \mathcal{A}_*$. As a consequence, Algorithm 2 will enter the inner loop with active set $\mathcal{A}_*$ at the iterate $(d_1, \lambda_1) \in B_{c_1\epsilon_2} \subseteq B_{\epsilon_1}$. As shown in the previous paragraph, the method will then remain in the inner loop, and from Assumption 1*ii)* it follows that $(\bar{d}_j^{\mathcal{F}}, \bar{\lambda}_j)$ converges to $(d_*^{\mathcal{F}_*}, \lambda_*)$. □

We point out that, in contrast to Theorem 1*ii)*, Theorem 2*iii)* does not require a contraction condition (27).

5 Numerical Results

To examine the practical performance of the proposed approach, a prototype implementation of Algorithm 2, which we will refer to as `iQP` (inexact QP solver), was created in Matlab R2012b. SQMR [16] was chosen as the iterative linear solver for the inner loop, because it exploits the symmetry of the matrix and allows indefinite preconditioners. For completeness, the detailed description of Algorithm 2 using SQMR is provided in Appendix A. We note that SQMR does not have theoretical convergence guarantees; in our implementation, SQMR is simply restarted if it breaks down, but this fall-back was triggered in our experiments very rarely. The QPs (20) and (31) were solved using the open-source active-set parametric QP solver `qpOASES` [10, 11]. All experiments were performed on an 8-core Intel-i7 3.4GHz 64bit Linux server with 32GB RAM. Matlab was set to use only a single computational thread.

We present two sets of numerical experiments. In Section 5.1, Algorithm 2 is used to solve a sequence of QPs that arise in certain nonlinear model predictive control (NMPC) applications. In Section 5.2, a sequence of randomly perturbed quadratically-constrained quadratic programs (QCQPs) is solved.

The goal of these experiments is two-fold. First, we explore the reliability of the new QP method in practice, given that convergence is not guaranteed. Second, we compare the performance of the new iterative `iQP` method implementing Algorithm 2 (as the hot-start approach) with that of a standard active set solver, `qpOASES` (as the warm-start approach). Because `qpOASES` uses dense linear algebra in its current implementation, our experiments are carried out for problems with dense matrix data.

Whether the new method requires overall less computation time for the solution of a new QP depends on a number of factors. The warm start approach requires the factorization of the KKT matrix in (6); this costs roughly $O((n_{\mathcal{F}})^3)$ floating-point operations for dense matrices, where $n_{\mathcal{F}}$ is the number of free variables. In addition, for each iteration of the active-set QP solver, in which one variable leaves or enters the active set, the linear system (6) is solved twice, and the factorization of the KKT matrix is updated for a new active set; this requires roughly $O((n_{\mathcal{F}})^2)$ operations. On the other hand, the hot-start approach does not require a factorization with work $O((n_{\mathcal{F}})^3)$, but the number of solves of the linear system (6) increases because a small number of reference QPs have to be solved per `iQP` iteration, each of which might require several active-set changes, particularly in the first `iQP` iterations. Therefore, whether the new approach requires less computational effort depends on the number of `iQP` iterations and the relative cost of factorizing the KKT matrix in (6) and the solution of the corresponding linear system.

In some applications, the computation of the matrices A and W in (1) dominate the computational time. This is, for example, the case when the constraints involve the integration of differential equations, as in the NMPC context (see Section 5.1.3). In that case, the effort for evaluating the entire A matrix might be equivalent to computing n products of A (or m products of A^T) with a vector. Therefore, we also report the number of matrix-vector products involving both $A \cdot v$ and $A^T \cdot v$ in our statistics. If this count is significantly smaller than n or m over the entire execution of an `iQP` run, a large reduction in overall computation time can be expected.

Symbol	Value	Unit	Symbol	Value	Unit	Symbol	Value	Unit
g	$(0,0,9.81)^T$	$\mathrm{m/s^2}$	x_e	$(7.5,0,0)^T$	m	w_sl	0.1	–
k	0.1	N/m	w_v	0.25	–	x_1^min	-0.5	m
l_r	0.55	m	w_x	25	–	x_1^max	8	m
m	0.45	kg	w_u	0.01	–	x_3^min	-10	m

Table 1: Parameter values for the chain of point masses model (35).

5.1 QPs from Nonlinear Model-Predictive Control

In this section, we investigate the performance of `iQP` on a sequence of QPs from a nonlinear model predictive control (NMPC) application.

5.1.1 Chain of Point Masses Problem

Our NMPC case study involves a motion control problem for a chain of N_PM free point masses, indexed by $1 \le i \le N_\mathrm{PM}$, that are connected by springs and subject to gravity. An additional point mass, indexed by 0, is fixed at the origin. Point mass positions at time t are denoted by $x_i(t) = (x_{i,\mathrm{x}}(t), x_{i,\mathrm{y}}(t), x_{i,\mathrm{z}}(t)) \in \mathbb{R}^3$ and velocities by $v_i(t) \in \mathbb{R}^3$. Starting with initial conditions $x_i(0) = (7.5i/N_\mathrm{PM}, 0, 0)^T$, $v_i(0) = (0,0,0)^T$, the point masses are accelerated by gravity and the chain's springs expand. The dynamic model is free of friction such that, once accelerated by gravity, it does not return to rest without appropriate application of external forces. The velocity $v_{N_\mathrm{PM}}(t)$ of the final point mass may be controlled through $u(t) = (u_\mathrm{x}(t), u_\mathrm{y}(t), u_\mathrm{z}(t)) \in \mathbb{R}^3$. The goal of the controller is to determine velocities $u(t)$ for the final point mass that bring the chain to rest. This optimal control problem can be written as

$$\begin{aligned}
\min_{x(\cdot),u(\cdot)} \quad & \int_0^T w_\mathrm{v} \sum_{i=1}^{N_\mathrm{PM}} \|v_i(t)\|_2^2 + w_\mathrm{x}\|x_{N_\mathrm{PM}}(t) - x_\mathrm{e}\|_2^2 + w_\mathrm{u}\|u(t)\|_2^2 + w_\mathrm{sl}\|u_\mathrm{sl}(t)\|_2^2 \,\mathrm{d}t && (35a)\\
\text{s.t.} \quad & \dot{x}_i(t) = v_i(t) \qquad t \in [0,T],\ 1 \le i < N_\mathrm{PM} && (35b)\\
& \dot{v}_i(t) = (F_{i+1}(t) - F_i(t)) \cdot N_\mathrm{PM}/m - g \qquad t \in [0,T],\ 1 \le i < N_\mathrm{PM} && (35c)\\
& \dot{x}_{N_\mathrm{PM}}(t) = u(t) \qquad t \in [0,T] && (35d)\\
& x(0) = \hat{x}_0 && (35e)\\
& u(t) \in [-1,1]^3 \qquad t \in [0,T] && (35f)\\
& x_1^\mathrm{low} \le x_1(t) \le x_1^\mathrm{high}, \quad x_3^\mathrm{low} - u_\mathrm{sl}(t) \le x_3(t) \qquad t \in [0,T], && (35g)\\
& u_\mathrm{sl}(t) \ge 0, \qquad t \in [0,T]. && (35h)
\end{aligned}$$

We denote by $F_i(t)$ the forces $F_i(t) := \big(x_i(t) - x_{i-1}(t)\big) \cdot k\big(n - l_\mathrm{r}/\|x_i(t) - x_{i-1}(t)\|_2\big)$. The slack variables $u_\mathrm{sl}(t)$ penalizes violation of the lower bound on $x_3(t)$. Characteristics and weights are given in Table 1. This problem has been considered in similar form in, e.g., [19].

5.1.2 Nonlinear Model-Predictive Control

To simplify the notation, we rewrite (35b)–(35e) as

$$\dot{y}(t) = D(y(t), u(t)),\ t \in [0,T], \qquad y(0) = \hat{y}_0. \tag{36}$$

In the online NMPC setting, one considers a sequence of sample times τ^k, indexed by k. The state $\hat{y}_0(\tau^k)$ of a physical system is monitored (sampled) at τ^k. Following the idea of real-time iterations

[9], the optimal control answer is computed as the solution of a feedback QP, see Section 5.1.3. From this solution, the initial optimal control action $u^*(\tau^k)$ is applied to the system. After the feedback interval $\Delta\tau$ has elapsed, the system state $\hat{y}_0(\tau^{k+1})$ is remeasured at $\tau^{k+1} = \tau^k + \Delta\tau$ and the feedback QP is resolved with the new value of $\hat{y}_0$. Hence, the number of NMPC samples is equal to N_{QP}, the number of QPs solved.

In order to simulate the change of the point masses process from one sample time τ^k to the next τ^{k+1} in our experiments, the forward problem (35b)–(35d) is solved, starting in the previous initial value $\hat{y}_0(\tau^k)$ and applying the most recent feedback control u_0^k for the duration of the sampling interval. The state at the end of this simulation is then taken as the (unperturbed) initial conditions $\hat{y}_0(\tau^{k+1})$ for the next sample.

5.1.3 Feedback QP Problem

The feedback QP is obtained from the optimal control problem (35) by discretizing the ordinary differential equation (ODE) (35b)–(35d) and linearizing the resulting NLP. In our experiment, we follow the direct multiple shooting approach [6, 20]. To this end, we choose an equidistant time discretization $0 = t_0 < t_1 < \ldots < t_{N_{\text{MS}}-1} < t_{N_{\text{MS}}} = T$ of the prediction horizon $[0, T]$ into N_{MS} shooting intervals $[t_j, t_{j+1}]$, $0 \le j < N_{\text{MS}}$. We introduce control parameters $u_j \in \mathbb{R}^3$ for a piecewise constant control discretization, $u(t)\mid_{t\in[t_j,t_{j+1})} = u_j$ for $0 \le j < N_{\text{MS}}$. In addition, we apply the multiple shooting state parametrization

$$\dot{y}(t) = D(y(t), u_j),\ t \in [t_j, t_{j+1}], \qquad y(t_j) = s_j \tag{37}$$

that decouples the forward problem (36) into N_{MS} initial value problems. In order to ensure consistency of the optimal solution, we introduce the additional matching conditions

$$y(t_{j+1};\ t_j, s_j, u_j) - s_{j+1} = 0, \quad 0 \le j < N_{\text{MS}}. \tag{38}$$

Herein, $y(t_{j+1};\ t_j, s_j, u_j)$ denotes the solution of (37) on $[t_j, t_{j+1}]$ evaluated in t_{j+1} when started with initial value s_j, applying the control u_j. Inequality path constraints (35g) are enforced on the shooting grid $\{t_j\}_{0\le j\le N_{\text{MS}}}$, resulting in constraints of the form

$$0 \le r_j(s_j, u_j), \quad 0 \le j \le N_{\text{MS}}. \tag{39}$$

We set $u_{N_{\text{MS}}} = u_{N_{\text{MS}}-1}$ for simplicity of notation in (39). In summary, this discretization and parametrization transforms problem (35) into a discrete-time control problem that is a finite-dimensional NLP.

The linearization of this NLP about a reference point $(\bar{s}, \bar{u})$ is

$$\begin{aligned}
\min_{s_j, u_j} \quad & \sum_{j=0}^{N_{\text{MS}}} \tfrac{1}{2} z_j^T W_j z_j + g_j^T z_j && && \text{(40a)}\\
\text{s.t.} \quad & s_{j+1} = D_j s_j + E_j u_j + f_j && 0 \le j < N_{\text{MS}} && \text{(40b)}\\
& s_0 = \hat{y}_0 && && \text{(40c)}\\
& u_j \in [0,1]^3 && 0 \le j \le N_{\text{MS}} && \text{(40d)}\\
& 0 \le P_j s_j + Q_j u_j + p_j && 0 \le j \le N_{\text{MS}}, && \text{(40e)}
\end{aligned}$$

with the block Gauß-Newton Hessian approximations W_j, the gradient of the objective parts g_j, constraint matrices $D_j = \nabla_s y(t_{j+1};\ t_j, \bar{s}_j, \bar{u}_j)$, $E_j = \nabla_u y(t_{j+1};\ t_j, \bar{s}_j, \bar{u}_j)$, $P_j = \nabla_s r_j(\bar{s}_j, \bar{u}_j)$, and $Q_j = \nabla_u r_j(\bar{s}_j, \bar{u}_j)$, and constraint vectors $f_j = y(t_{j+1};\ t_j, \bar{s}_j, \bar{u}_j)$ and $p_j = r_j(\bar{s}_j, \bar{u}_j)$. Therein, the "expensive" matrices D_j, E_j, and W_j are usually dense and require the numerical computation of sensitivities of the solution of the initial value problem (37) with respect to all independent variables s_j and u_j, and capture the ODE dynamics of the process. Typically, computing the sensitivity of $y(t_{j+1};\ t_j, s_j, u_j)$ with respect to a single variable $s_j^{(i)}$ or $u_j^{(i)}$ is about as time-consuming as a forward integration [2]. As a consequence, the computation of the constraint Jacobians D_j, E_j, and the Hessian (approximation) W_j can become the computational bottleneck and can take up more than 90 percent of the CPU time [19]. To address this, [5] proposed an NMPC algorithm named "Mode C" that solves a single QP (24), i.e., it performs one iteration of Algorithm 1. The `iQP` approach proposed in this article improves over this idea by employing a preconditioner and performing multiple iterations to compute an improved solution.

5.1.4 Results

We consider the NMPC scenario for chains with $N_{\text{PM}} \in \{6, 8, 10, 12, 14\}$ point masses, and for a prediction horizon of $T = 8$ seconds discretized into $N_{\text{MS}} \in \{15, 20\}$ intervals. We give feedback every $\Delta\tau = T/N_{\text{MS}} \in \{0.5333, 0.4\}$ seconds, and run this scenario for $\tau_{\max} = 30$ seconds, computing $N_{\text{QP}} = \lceil \tau_{\max} N_{\text{MS}}/T \rceil \in \{57, 76\}$ samples. This duration and sampling rate was sufficient for the NMPC controller to successfully settle the system in all investigated scenarios. The resulting dimensions are listed in Table 2, where n_{y} is the number of state variables, $n_{\text{u}} = 3 + (N_{\text{PM}} - 1)$ is the number of control and slack variables, and n and m are the number of QP variables and constraints. We obtain the reference QP as the linearlization (40) at the steady state of the system, which is obtained by setting $u(t) \equiv 0$ and $y(t) \equiv \hat{y}_0$ such that the chain is at rest, satisfying $D(\hat{y}_0, 0) \equiv 0$.

In Table 2, we report the performance of the `iQP` algorithm on the ten QP sequences. All QPs were solved successfully by `iQP`, with the exception of one QP in the $N_{\text{MS}} = 20$, $N_{\text{PM}} = 6$ series. The metric for this experiment is the number of matrix-vector products with the constraint matrix A in (1) and its transpose, which are computationally expensive because these involve the sensitivity matrices D_j and E_j and require computations by the ODE solver. The minimum, maximum, and mean of the total number of products required by `iQP` during the solution of the QPs (20) and (31) is compared with the equivalent number of matrix-vector products that are necessary to compute the entries of the almost block-diagonal matrix A_k. Here we assume that each of the N_{MS} blocks in A can be obtained by n_{y} products of the transpose of the sensitivity matrices with unit vectors. As can be seen, `iQP` reduces this effort by a factor of up to 2.4 on average.

As the physical system settles and gets closer to the desired steady-state solution, the differences between the QPs from one sample time to the next become smaller, and `iQP` requires fewer iterations. This can be seen in Figure 2, which illustrates the diminishing number of active set changes and matrix-vector products over the QP sequence for a typical case.

5.2 Solving Sequences of Similar NLPs with SQP

As briefly discussed in the introduction, a sequence of QPs with similar data also arises when the SQP algorithm is applied to a sequence of similar NLPs. In this section, we consider the sequential solution of quadratically-constrained quadratic problems, which we will refer to as QCQP(t) with $t = 0, 1, 2, \ldots$, of the form

$$\min_{x \in \mathbb{R}^n} \quad \tfrac{1}{2} x^T H_0 x + (q_0^t)^T x + r_0^t \tag{41a}$$

$$\text{s.t.} \quad \tfrac{1}{2} x^T H_j x + (q_j^t)^T x + r_j^t \le 0 \qquad \text{for all } j = 1, \ldots, m \tag{41b}$$

$$x \ge 0. \tag{41c}$$

Dimensions		**Sizes**				**QPs**		**Matrix-Vector Products**			
N_{MS}	N_{PM}	n_{y}	n_{u}	n	m	Total	Success	Min	Mean	Max	Full A_k
15	6	33	8	656	575	57	57	16	29.8	128	33
15	8	45	10	880	787	57	57	20	33.5	130	45
15	10	57	12	1,104	999	57	57	24	36.9	102	57
15	12	69	14	1,328	1,211	57	57	32	38.2	90	69
15	14	81	16	1,522	1,423	57	57	28	35.8	74	81
20	6	33	8	861	765	76	75	16	32.4	148	33
20	8	45	10	1,155	1,047	76	76	16	30.0	112	45
20	10	57	12	1,449	1,329	76	76	24	34.8	124	57
20	12	69	14	1,743	1,611	76	76	24	33.0	104	69
20	14	81	16	1,977	1,893	76	76	24	33.8	108	81

Table 2: `iQP` statistics for the NMPC case study.

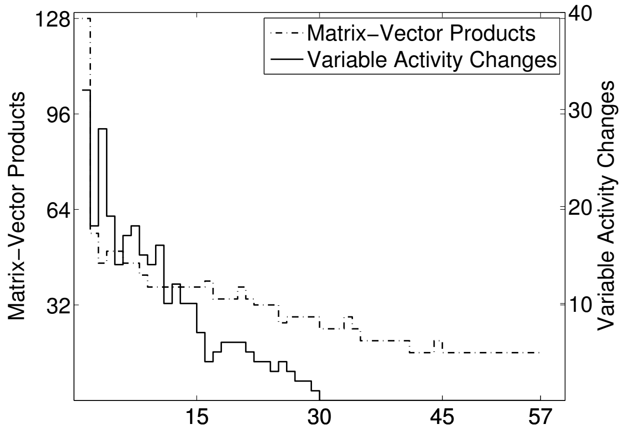


Figure 2: Number of matrix-vector products and active set changes over the course of 57 solved QPs for the $N_{\mathrm{PM}} = 6$, $N_{\mathrm{MS}} = 15$ chain simulation.

5.2.1 Experimental Setup

For a chosen problem size of n variables and m constraints, the base (or reference) problem, indexed by $t = 0$, is generated using the following steps.

1. Choose optimal solution values $x_* \sim \mathcal{U}(-1,1)^n$, $\lambda_* \sim \mathcal{U}(0,1)^m$, and $\mu_* \sim \mathcal{U}(0,1)^n$, where $\mathcal{U}(a,b)$ is the uniform distribution on the interval $[a,b]$.

2. Adapt the solution so that the first κ_x variable bounds and the first κ_c inequality constraints are active: Reset the first κ_x elements in x_*, the last $n-\kappa_x$ elements in μ_*, and the last $m-\kappa_c$ elements in λ_* to zero.

   Here, $0 \le \kappa_c \le m$ and $0 \le \kappa_x \le n$ denote the number of variable bound constraints (41b) and inequality constraints (41c) that are active at the optimal solution. For our experiments, we choose $\kappa_c = \lceil 1/3m \rceil$ and $\kappa_x = \lceil 1/3n \rceil$.

3. For $j = 0,\ldots,m$, generate random symmetric positive definite matrices H_j $(j = 0,\ldots,m)$ of dimension $n \times n$ with condition number 1000. For our experiments, we use the Matlab function `sprandsym(n,.2,1e-3,1)`.

4. For $j = 1,\ldots,m$, choose $q_j^0 \sim \mathcal{U}(-1,1)^n$ and set $r_j^0 := -\left(\frac{1}{2}x_*^T H_j x_* + (q_j^0)^T x_*\right)$. Add a uniform random number from $\mathcal{U}(0,1)$ to each of the last $m-\kappa_c$ elements in r_j^0 to make the corresponding constraints inactive.

5. Set $q_0^0 := -\left((H_0 + \lambda_*^{(j)} H_j)x_* + \sum_{j=1}^m \lambda_*^{(j)} q_j^0 - \mu_*\right)$ and $r_0^0 = -\left(\frac{1}{2}x_*^T H_0 x_* + (q_0^0)^T x_*\right)$.

It can easily be verified that then x_* is the optimal solution of (41) for $t = 0$, and that λ_* and μ_* are the corresponding multipliers. By construction, strict complementarity holds, and the objective and constraint functions are convex. Note that this problem can be reformulated into the standard form (3) by introducing slack variables.

From the QCQP(0) data, we generate the nearby problem instance QCQP(t) by perturbing each entry in the problem data q_j^t and r_j^t via

$$(q_j^t)^{(i)} \sim \mathcal{N}\left((q_j^0)^{(i)}, \sigma^2\right), \quad r_j^t \sim \mathcal{N}\left(r_j^0, \sigma^2\right), \qquad i = 1,\ldots,n \text{ and } j = 0,1,\ldots,m. \tag{42}$$

Here, $\mathcal{N}(\mu,\sigma^2)$ denotes the normal distribution with mean μ and variance σ^2, and $\sigma > 0$ is a fixed parameter controlling the size of the perturbation.

These QCQPs are solved using the MATLAB implementation "`p-sqp`", developed by Frank E. Curtis, of the Sℓ_1QP method proposed in [7] with minor modifications. Here, at an SQP iterate (x_k, λ_k), the search direction for the line search is obtained as the optimal solution of the ℓ_1QP

$$\min_{d,p,s} \quad \rho_k \nabla f(x_k)^T d + \tfrac{1}{2} d^T W_k d + e^T p \tag{43a}$$

$$\text{s.t.} \quad \nabla c(x_k)^T d + c(x_k) + s - p = 0 \tag{43b}$$

$$x_k + d \ge 0 \tag{43c}$$

$$s, p \ge 0, \tag{43d}$$

where $e = (1,\ldots,1)^T$, $\rho_k \ge 0$ is the current value of the (inverse of the) penalty parameter, and $W_k = \rho_k \nabla^2 f(x_k) + \sum_{j=1}^m \lambda_k^{(j)} \nabla^2 c^{(j)}(x_k)$. Due to the convexity assumption, W_k is always positive definite, because (ρ_k, λ_k) remains non-negative and non-zero throughout the optimization. The

	Size of perturbation σ			
	0.01	0.05	0.1	0.2
Successfully solved QCQPs	99.3%	99.3%	97.7%	42.9%
Average number of `iQP` iterations	4.9	8.5	12.5	20.6
Average change in active set for (41b)	5.3%	19.0%	25.7%	32.4%
Average change in active set for (41c)	3.4%	9.5%	13.8%	19.2%

Table 3: Statistical observations for QCQP experiments

details of the SQP method are not relevant here; the interested reader is referred to [7]. We only point out that the ℓ_1QP (43) is always feasible by construction. Therefore, for these problems, the assumptions pertaining to feasibility of the QPs (1), (20) and (31) and the positive definiteness of their Hessian matrices made for Theorems 1 and 2 are satisfied in each SQP iteration.

In our experimental setup, we initially solve QCQP(0) with the SQP method using a standard active-set QP solver (`qpOASES` in our context), and then "fix" the QP (43) corresponding to the instance QCQP(0) at the returned solution x_* and λ_* as the reference QP (2), i.e., A_0 and W_0 are chosen to be the matrices corresponding to x_* and λ_*. The internal state of the QP solver is also stored. We then apply the Sℓ_1QP algorithm to the perturbed QCQP(t) problems with $t = 1, 2, 3, \ldots$, using x_* and λ_* as initial iterates. The termination tolerance for the Sℓ_1QP algorithm is set to 10^{-6}.

The QPs (43) are solved using our `iQP` implementation of Algorithm 2. At the beginning of each SQP run, the QP solver for (2) is restored to the internal state corresponding to (x_*, λ_*), and subsequently only hot-starts are used for any solution of (20) and (31) required for Algorithm 2. In each SQP iteration k, Algorithm 2 is terminated when the residual function (22) satisfies

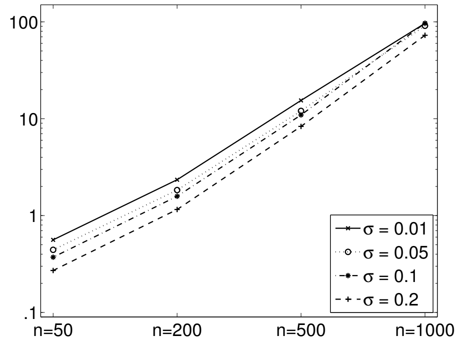
$$\Phi_k((d, s, p), \lambda) \leq \min\{10^{-8}, 10^{-5}\epsilon_k\}, \tag{44}$$

where ϵ_k is the current KKT error for QCQP(t). This tight tolerance is necessary because the convergence analysis for Sℓ_1QP method in [7] assumes the exact solution of (43), and `p-sqp` frequently fails to converge if less accurate solutions are returned. The experiments below show that even such highly accurate solutions are obtained by Algorithm 2 with a reasonable amount of work. If Algorithm 2 fails to satisfy condition (44) within 100 `iQP` iterations in some SQP iteration k, the SQP algorithm is terminated with an error message.

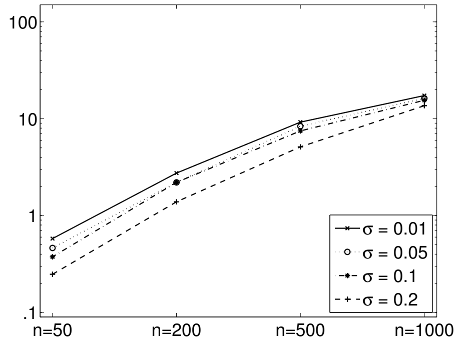
5.2.2 Results

The detailed results of our numerical experiments are reported in Appendix B. A total of 64 combinations of sizes and perturbation levels σ were considered: The numbers of variables were chosen as $n \in \{50, 200, 500, 1000\}$, and the numbers m of inequality constraints took the values $20\%n$, $50\%n$, $80\%n$, and $150\%n$. The perturbations were chosen as $\sigma \in \{0.01, 0.05, 0.1, 0.2\}$. For each such combination, one QCQP(0) was generated, and then 10 perturbed instances were solved (only 3 perturbed instances for $n = 1000$ due to the excessive computation times). The values reported in Appendix B are the averages over those 10 (or 3) runs.

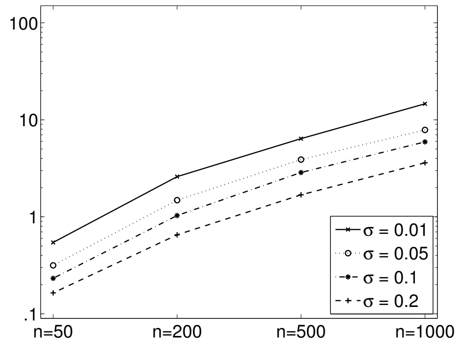
The SQP algorithm was run twice on each instance, once with the standard active-set QP solver `qpOASES` and once with the new method `iQP` to solve the step computation QP (43). Except for eight out of about 600 successfully solved instances, the number of SQP iterations was identical for both QP solvers.



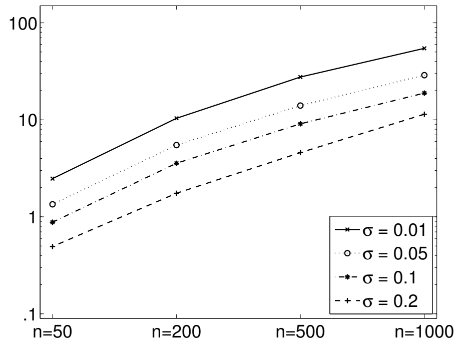
(a) CPU time; $m = 0.2n$



(b) CPU time; $m = 0.8n$



(c) Matrix-vector products; $m = 0.2n$



(d) Matrix-vector products; $m = 0.8n$

Figure 3: Relative performance of `qpOASES` vs. `iQP`. For a given number of constraints m, number of variables n and perturbation σ, the graphs show the ratio $r = \text{metric}_{\texttt{qpOASES}}/\text{metric}_{\texttt{iQP}}$, where "metric" is the average CPU time or number of matrix-vector products.

Table 3 presents a summary of the performance of `iQP`. It is noteworthy that `iQP`, despite the lack of a convergence guarantee, is able to solve most of the arising QPs and exhibits a considerable level of reliability except when the problem perturbation becomes large. Furthermore, the number of `iQP` iterations is on average only between 5 and 20 per QP, even though the level of accuracy is rather tight (the right-hand side in (44) is between 10^{-8} and 10^{-11}). We also observe that the sets of inequality constraints (41b) and variable bounds (41c) that are active at the optimal solution of the reference QCQP(0) and the perturbed QCQP(t) are significantly different, showing that the problem perturbations are non-trivial.

Figures 3(a) and 3(b) compare the performance of `qpOASES` and `iQP` in terms of CPU time. As the problem size increases, the new method becomes increasingly faster compared to the standard approach; for $n = 1000$, we see a reduction of up to two orders of magnitude in CPU time. However, we point out that in this experiment all matrices in (1) are dense, and the balance is likely to shift in favor of `qpOASES` if sparse linear algebra methods can be used.

We also compare the number of matrix-vector products involving $\nabla c(x_k)$ and $\nabla c(x_k)^T$ required by the algorithm. This is relevant if the evaluation of the constraint Jacobians are the bottleneck of the computation; for example, when the constraints involve the numerical integration of differential equations. Figures 3(c) and 3(d) show a significant improvement, with a reduction of up to more than one order of magnitude for the large instances. Here, the number of matrix-vector products is obtained by counting the products with both $\nabla c(x_k)$ and $\nabla c(x_k)^T$ during an `iQP` run. For the `qpOASES` case, we consider m matrix-vector products with $\nabla c(x_k)$ to be equivalent to the

		iQP iteration															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
SQP iteration	0	13	6	$5^{\pm}$	5	$2^{\pm}$	1	2	0^{+}	0	0	0	0	0	0	0	0
	1	6	2	0^{+}	0	0	0	0	0	0	0	0	0	0	0		
	2	1	0^{+}	0	0	0	0	0	0	0	0	0	0				
	3–7	1	0^{+}	0	0	0	0	0	0	0	0						

Table 4: Number of `qpOASES` pivots in an `iQP` iteration within the given SQP iteration. The superscript $^{+}$ ($^{\pm}$) indicates that the inner SQMR loop in Algorithm 2 was started (started and immediately terminated) in an iteration.

computation of the full matrix $\nabla c(x_k)$, since all matrix elements can be obtained by products of $\nabla c(x_k)$ with the m unit vectors.

Finally, to see the progression over an entire SQP run, we report one typical case in more detail. Table 4 lists the total number of pivots taken by `qpOASES` when solving the QPs (20) or QP (31) for each `iQP` iteration over the course of the SQP algorithm. Most pivots are taken in the first `iQP` iteration of the first SQP iteration, indicating that the active set changes significantly compared to the reference QP. In the later `iQP` iterations, in which SQMR updates are performed, no pivots are required because then the active set remains constant.

6 Conclusions

We proposed a new QP algorithm that uses hot-starts of an active-set QP solver for a previously solved reference QP in order to accelerate the solution of a similar QP. The numerical study showed that this approach can reduce the computational effort when a sequence of similar QPs or NLPs is solved. Our approach has two advantages.

First, when the computation of the constraint matrix of the QP requires expensive calculations, such as the integration of differential equations, the evaluation of the full constraint matrix can be avoided, and only matrix-vector products (obtained using adjoint calculations or automatic differentiation techniques) are required. This benefit was demonstrated on a nonlinear model-predictive control example.

Secondly, as shown for a set of randomly perturbed NLPs, speedup can be obtained when, for each new SQP iteration, the factorization of the KKT matrix inside an active-set QP solver is replaced by a sequence of hot-starts. This observations was made for instances with dense matrices. It remains subject of future research whether this advantage is also observed when sparse linear algebra techniques can be used. Furthermore, we postulate that additional computation time could be saved if the SQP algorithm is designed to handle inexact QP solutions so that our method could be terminated after fewer iterations. A candidate of such an algorithm has been proposed in [8].

The proposed algorithm is proven to converge if it is started sufficiently close to a non-degenerate QP solution. The method may diverge or cycle, however, as is the case with any iterative refinement procedure for linear systems. One premise of the present work is that a black-box QP solver can be used in this framework, and that this QP solver is responsible for handling the update of the active set. It appears difficult or impossible to design a globally convergent variant of the proposed algorithm without explicitly managing the active set. Nevertheless, the numerical results show that the method is robust to moderate changes of the QP data. In a practical setting, one could attempt to solve a QP with the proposed method, and, if cycling or divergence is observed, the QP could be solved with a regular active-set QP solver. This new QP solution may then be used as the new reference QP.

Acknowledgments

The authors would like to thank Frank E. Curtis for sharing his Matlab implementation `p-sqp` of the algorithm presented in [7]. We are also grateful to Jorge Nocedal and Andreas Potschka, whose comments on the manuscript helped to improve the exposition of the material.

References

[1] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2005.

[2] J. Albersmeyer. *Adjoint based algorithms and numerical methods for sensitivity generation and optimization of large scale dynamic systems.* PhD thesis, Ruprecht–Karls–Universität Heidelberg, 2010.

[3] R.A. Bartlett and L.T. Biegler. QPSchur: A dual, active set, Schur complement method for large-scale and structured convex quadratic programming algorithm. *Optimization and Engineering*, 7:5–32, 2006.

[4] M.J. Best. An algorithm for the solution of the parametric quadratic programming problem. In H. Fischer, B. Riedmüller, and S. Schäffler, editors, *Applied Mathematics and Parallel Computing*, pages 57–76. Physica-Verlag, Heidelberg, 1996.

[5] H.G. Bock, M. Diehl, E.A. Kostina, and J.P. Schlöder. Constrained Optimal Feedback Control for DAE. In L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors, *Real-Time PDE-Constrained Optimization*, chapter 1, pages 3–24. SIAM, 2007.

[6] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the 9th IFAC World Congress*, pages 242–247, Budapest, 1984. Pergamon Press.

[7] R.H. Byrd, F.E. Curtis, and J. Nocedal. Infeasibility detection and SQP methods for nonlinear optimization. *SIAM Journal on Optimization*, 20(5):2281–2299, 2008.

[8] F.E. Curtis, T.C. Johnson, D.P. Robinson, and A. Wächter. An Inexact Sequential Quadratic Optimization Algorithm for Large-Scale Nonlinear Optimization. Technical Report 13T-001, COR@L Laboratory, Department of ISE, Lehigh University, 2013.

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

[10] H.J. Ferreau, H.G. Bock, and M. Diehl. An online active set strategy for fast parametric quadratic programming in MPC applications. In *Proceedings of the IFAC Workshop on Nonlinear Model Predictive Control for Fast Systems, Grenoble*, 2006.

[11] H.J. Ferreau, C. Kirches, A. Potschka, H.G. Bock, and M. Diehl. qpOASES: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 2013. (submitted).

[12] A.V. Fiacco and G.P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques.* John Wiley, New York, USA, 1968. Reprinted by SIAM Publications, 1990.

[13] R. Fletcher. *Practical Methods of Optimization.* John Wiley and Sons, New York, USA, second edition, 1987.

[14] R. Fletcher. Stable reduced Hessian updates for indefinite quadratic programming. *Mathematical Programming*, 87(2):251–264, 2000.

[15] D. Fong and M. Saunders. LSMR: An iterative algorithm for sparse least-squares problems. *SIAM Journal on Scientific Computing*, 33(5):2950–2971, 2011.

[16] R.W. Freund. Preconditioning of symmetric, but highly indefinite linear systems. In A. Sydow, editor, *Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics*, pages 551–556, Berlin, 1997. Wissenschaft & Technik.

[17] P.E. Gill, W. Murray, M.A. Saunders, and M.H. Wright. A Schur-complement method for sparse quadratic programming. In M. G. Cox and S. J. Hammarling, editors, *Reliable Scientific Computation*, pages 113–138. Oxford University Press, Oxford, 1990.

[18] N.I.M. Gould and Ph.L. Toint. An iterative working-set method for large-scale nonconvex quadratic programming. *Applied Numerical Mathematics*, 43(1-2):109–128, 2002.

[19] C. Kirches, L. Wirsching, H.G. Bock, and J.P. Schlöder. Efficient direct multiple shooting for nonlinear model predictive control on long horizons. *Journal of Process Control*, 22(3):540–550, 2012.

[20] D.B. Leineweber, I. Bauer, A.A.S. Schäfer, H.G. Bock, and J.P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization (Parts I and II). *Computers & Chemical Engineering*, 27:157–174, 2003.

[21] J.T. Linderoth and M.W.P. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11(2):173–187, 1999.

[22] C.C. Paige and M.A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software*, 8(1):43–71, 1982.

[23] Y. Saad and M.H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–860, 1986.

A Detailed Description of Algorithm (2) for SQMR

Algorithm 3 Solving QP (1) using hot-starts for QPs (20) and (31), accelerated by SQMR

```
1: Given: Initial iterates $d_0 \geq \ell$ and $\lambda_0$.
2: Initialize: $\mathcal{A}_{-1} = \{l \mid d_0^{(l)} = \ell^{(l)}\}$ and ref_flag $\leftarrow$ false.
3: for $i = 0, 1, 2, \ldots$ do
4:     Solve QP (20) to obtain optimal solution $p_i$ with optimal multipliers $p_i^\lambda$.
5:     Determine the active set $\mathcal{A}_i = \{\, l \mid d_i^{(l)} + p_i^{(l)} = \ell^{(l)}\}$.
6:     if $\mathcal{A}_i = \mathcal{A}_{i-1}$ and ref_flag = true then
7:         Set ref_flag $\leftarrow$ false.
8:         Fix active set $\mathcal{A} \leftarrow \mathcal{A}_i$ and $\mathcal{F} \leftarrow \mathcal{A}^C$.
9:         Initialize SQMR with $\begin{pmatrix} r_0 \\ s_0 \end{pmatrix} = \begin{pmatrix} -\left(g + Wd_i + A^T\lambda_i\right)^{\mathcal{F}} \\ c + Ad_i \end{pmatrix}$, $t = \begin{pmatrix} p_i^{\mathcal{F}} \\ p_i^\lambda \end{pmatrix}$, $q_0 = t$, $\tau_0 = \|t\|$, $\theta_0 = 0$,
           $\rho_0 = \begin{pmatrix} r_0 \\ s_0 \end{pmatrix}^T q_0$, $\begin{pmatrix} \bar{p}_0^{\mathcal{F}} \\ \bar{p}_0^\lambda \end{pmatrix} = 0$, and $\begin{pmatrix} \bar{d}_0^{\mathcal{F}} \\ \bar{\lambda}_0 \end{pmatrix} = 0$.
10:        for $j = 0, 1, 2, \ldots$ do
11:            Compute $\tilde{t} = \mathcal{M}q_j$, $\sigma_j = q_j^T\tilde{t}$, $\alpha_j = \rho_j/\sigma_j$, $\begin{pmatrix} r_{j+1} \\ s_{j+1} \end{pmatrix} = \begin{pmatrix} r_j \\ s_j \end{pmatrix} - \alpha_j\tilde{t}$.
12:            Apply preconditioner by solving QP (31) to obtain $z_{j+1}$ and $z_{j+1}^\lambda$.
13:            if $z_{j+1}^{\mathcal{A}} \neq 0$ in optimal solution of QP (31) then
14:                Update outer iterate $(d_{i+1}^{\mathcal{F}}, d_{i+1}^{\mathcal{A}}, \lambda_{i+1}) \leftarrow (\bar{d}_{j+1}^{\mathcal{F}}, \ell^{\mathcal{A}}, \bar{\lambda}_{j+1})$; break.
15:            end if
16:            Compute: $t = \begin{pmatrix} z_{j+1}^{\mathcal{F}} \\ z_{j+1}^\lambda \end{pmatrix}$, $\theta_{j+1} = \|t\|_2/\tau_j$, $\gamma_{j+1} = 1/\sqrt{1+\theta_{j+1}^2}$, and $\tau_{j+1} = \tau_j\theta_{j+1}\gamma_{j+1}$.
17:            Compute: $\begin{pmatrix} \bar{p}_{j+1}^{\mathcal{F}} \\ \bar{p}_{j+1}^\lambda \end{pmatrix} = \gamma_{j+1}^2\theta_j^2 \begin{pmatrix} \bar{p}_j^{\mathcal{F}} \\ \bar{p}_j^\lambda \end{pmatrix} + \gamma_{j+1}^2\alpha_j q_j$.
18:            Update: $\begin{pmatrix} \bar{d}_{j+1}^{\mathcal{F}} \\ \bar{\lambda}_{j+1} \end{pmatrix} = \begin{pmatrix} \bar{d}_j^{\mathcal{F}} \\ \bar{\lambda}_j \end{pmatrix} + \begin{pmatrix} \bar{p}_{j+1}^{\mathcal{F}} \\ \bar{p}_{j+1}^\lambda \end{pmatrix}$ and $\bar{d}_{j+1}^{\mathcal{A}} = \ell^{\mathcal{A}}$.
19:            if $\bar{d}_{j+1}^{\mathcal{F}} \ngeq \ell^{\mathcal{F}}$ then
20:                Update outer iterate $(d_{i+1}^{\mathcal{F}}, d_{i+1}^{\mathcal{A}}, \lambda_{i+1}) \leftarrow (\bar{d}_{j+1}^{\mathcal{F}}, \ell^{\mathcal{A}}, \bar{\lambda}_{j+1})$; break.
21:            end if
22:            if $(\bar{d}_{j+1}^{\mathcal{F}}, \bar{\lambda}_{j+1})$ solves (28) then
23:                Return optimal solution $(d_*, \bar{\lambda}_{j+1})$ with $d_*^{\mathcal{F}} = \bar{d}_{j+1}^{\mathcal{F}}$ and $d_*^{\mathcal{A}} = \ell^{\mathcal{A}}$.
24:            end if
25:            Compute: $\rho_{j+1} = \begin{pmatrix} r_{j+1} \\ s_{j+1} \end{pmatrix}^T t$, $\beta_{j+1} = \rho_{j+1}/\rho_j$, $q_{j+1} = t + \beta_{j+1}q_j$.
26:        end for
27:    else
28:        Update $d_{i+1} = d_i + p_i$ and $\lambda_{i+1} = \lambda_i + p_i^\lambda$.
29:        Set ref_flag $\leftarrow$ true.
30:    end if
31:    if $(d_{i+1}, \lambda_{i+1})$ solves (1) then
32:        Return optimal solution $(d_{i+1}, \lambda_{i+1})$.
33:    end if
34: end for
```

B Details of Numerical Experiment for Random QCQPs

For a QCQP of a specified size and perturbation, the following table lists these quantities: average number of SQP iterations; the number of successfully solved instances, average CPU time (in seconds), and `qpOASES` pivots for the `qpOASES` experiments; the number of successfully solved QPs, average CPU time (in seconds), `iQP` iterations, `qpOASES` pivots, and matrix-vector products encountered in total during the `iQP` experiments.

Problem			**SQP**	**qpOASES**			**iQP**				
n	m	σ	it	✓	CPUs	#piv	✓	CPUs	it	#piv	#mvp
50	10	0.01	8.2	10	0.2	11.9	10	0.4	44.9	56.6	75.3
50	10	0.05	9.5	10	0.3	20.5	10	0.6	84.8	109.4	150.4
50	10	0.1	10.0	10	0.3	26.2	10	0.8	119.6	158.7	215.0
50	10	0.2	10.5	10	0.3	35.5	7	1.1	180.1	275.3	319.1
50	25	0.01	7.7	10	0.2	9.3	10	0.4	37.4	46.1	61.3
50	25	0.05	8.4	10	0.2	15.1	10	0.5	65.8	81.6	115.5
50	25	0.1	9.2	10	0.2	20.3	10	0.6	99.0	124.0	179.0
50	25	0.2	9.2	10	0.2	29.1	10	1.0	162.5	234.9	293.8
50	40	0.01	8.0	10	0.2	9.5	10	0.4	39.3	48.0	64.6
50	40	0.05	8.5	10	0.2	19.0	10	0.5	71.7	93.8	126.2
50	40	0.1	8.6	10	0.3	29.5	10	0.7	108.2	153.3	195.3
50	40	0.2	9.1	10	0.3	44.3	2	1.1	200.5	314.5	368.0
50	75	0.01	7.8	10	0.3	12.0	10	0.4	43.5	55.9	73.2
50	75	0.05	8.0	10	0.3	32.4	10	0.6	80.7	127.8	142.9
50	75	0.1	9.1	10	0.4	50.9	8	0.9	150.5	267.2	268.1
50	75	0.2	9.4	7	0.4	76.0	0	–	–	–	–
200	40	0.01	9.0	10	1.6	15.5	10	0.7	42.7	58.1	69.3
200	40	0.05	9.7	10	1.6	37.5	10	0.9	75.1	118.1	130.5
200	40	0.1	10.2	10	1.6	54.3	10	1.0	111.0	183.5	197.8
200	40	0.2	10.6	10	1.6	82.4	10	1.4	180.8	354.4	324.8
200	100	0.01	9.0	10	2.1	15.8	10	0.8	43.2	58.6	70.2
200	100	0.05	10.0	10	2.3	38.0	9	1.1	78.9	122.2	137.8
200	100	0.1	10.0	10	2.3	57.8	10	1.3	115.6	197.0	206.7
200	100	0.2	10.0	10	2.3	88.6	10	2.2	210.3	458.1	378.0
200	160	0.01	9.0	10	3.4	16.5	10	1.2	42.9	59.5	69.5
200	160	0.05	10.0	10	3.8	51.7	10	1.7	83.0	139.7	145.8
200	160	0.1	10.0	10	3.9	83.8	10	1.7	124.6	244.5	224.3
200	160	0.2	10.0	10	3.8	121.9	6	2.8	259.8	684.8	457.7
200	300	0.01	9.0	10	14.4	37.8	10	1.7	46.8	86.2	76.8
200	300	0.05	10.0	10	16.5	119.1	10	2.5	102.3	256.5	179.8
200	300	0.1	10.0	10	16.8	182.4	10	3.4	168.6	492.3	300.2
200	300	0.2	10.9	10	18.7	255.2	0	–	–	–	–
500	100	0.01	9.1	10	47.5	21.3	10	3.1	43.7	64.6	71.1
500	100	0.05	10.0	10	49.2	59.5	10	4.1	74.5	139.8	128.5
500	100	0.1	11.0	10	54.6	94.4	10	5.0	109.4	229.5	192.1
500	100	0.2	11.1	10	54.1	156.4	10	6.5	185.5	493.3	329.0
500	250	0.01	10.0	10	59.9	32.6	10	6.7	44.2	77.8	70.0
500	250	0.05	10.2	10	60.0	96.0	10	8.1	80.3	190.1	138.0

Continued on next page

Problem			**SQP**	**qpOASES**			**iQP**				
n	m	σ	it	✓	CPUs	#piv	✓	CPUs	it	#piv	#mvp
500	250	0.1	11.0	10	68.8	146.7	10	9.2	122.1	317.6	216.7
500	250	0.2	11.0	10	70.4	203.8	10	12.6	219.0	664.5	391.0
500	400	0.01	10.0	10	96.2	45.0	10	10.4	45.6	91.2	72.4
500	400	0.05	11.0	10	109.5	160.6	10	13.1	90.2	272.2	156.8
500	400	0.1	11.0	10	112.9	233.0	10	15.1	136.6	456.6	242.0
500	400	0.2	11.0	10	114.0	307.8	10	22.3	270.7	1044.8	478.9
500	750	0.01	10.0	10	349.2	129.8	10	20.8	52.2	186.7	84.8
500	750	0.05	11.0	10	393.3	422.2	10	32.5	112.8	612.7	195.0
500	750	0.1	11.0	10	399.8	557.8	10	43.7	171.0	976.1	302.6
500	750	0.2	11.0	10	404.9	687.1	0	–	–	–	–
1000	200	0.01	10.0	3	2099.5	34.7	3	21.8	43.7	79.0	68.3
1000	200	0.05	11.0	3	2456.9	124.3	3	27.0	80.7	214.0	139.7
1000	200	0.1	11.0	3	2942.1	191.0	3	30.4	106.0	329.3	185.7
1000	200	0.2	12.0	3	3176.1	296.0	3	43.7	187.3	671.3	333.0
1000	500	0.01	10.0	3	1448.3	66.3	3	49.6	45.3	113.7	71.7
1000	500	0.05	11.0	3	1991.9	230.0	3	62.7	81.7	343.7	138.7
1000	500	0.1	11.6	3	2431.3	328.0	2	73.9	122.0	518.0	215.0
1000	500	0.2	12.6	3	2774.5	434.7	2	98.8	228.0	1029.0	407.0
1000	800	0.01	10.0	3	1313.7	102.3	3	75.8	46.3	149.7	73.3
1000	800	0.05	11.2	3	1801.5	357.3	3	110.8	89.7	495.7	152.7
1000	800	0.1	12.0	3	2069.0	500.3	3	133.4	145.0	792.0	254.3
1000	800	0.2	12.6	2	2447.8	660.5	3	179.4	275.0	1666.0	473.0
1000	1500	0.01	11.0	3	3637.3	307.0	2	181.7	56.5	380.5	90.0
1000	1500	0.05	12.0	3	4082.9	866.0	3	284.2	115.3	1132.3	200.0
1000	1500	0.1	12.3	3	4459.8	1119.0	3	347.6	179.0	1655.0	317.7
1000	1500	0.2	12.2	3	4547.8	1331.3	1	556.4	334.0	3225.0	578.0