Methods for Convex and General Quadratic Programming*

Philip E. Gill† Elizabeth Wong†

July 13, 2014

Abstract

Computational methods are considered for finding a point that satisfies the second-order necessary conditions for a general (possibly nonconvex) quadratic program (QP). The first part of the paper considers the formulation and analysis of an active-set method for a generic QP with both equality and inequality constraints. The method uses a search direction that is the solution of an equality-constrained subproblem involving a "working set" of linearly independent constraints. The method is a reformulation of a method for general QP first proposed by Fletcher, and modified subsequently by Gould. The reformulation facilitates a simpler analysis and has the benefit that the algorithm reduces to a variant of the simplex method when the QP is a linear program. The search direction is computed from a KKT system formed from the QP Hessian and the gradients of the working-set constraints. It is shown that, under certain circumstances, the solution of this KKT system may be updated using a simple recurrence relation, thereby giving a significant reduction in the number of KKT systems that need to be solved.

The second part of the paper focuses on the solution of QP problems with constraints in so-called standard form. We describe how the constituent KKT systems are solved, and discuss how an initial basis is defined. Numerical results are presented for all QPs in the CUTEst test collection.

Key words. Large-scale quadratic programming, active-set methods, convex and nonconvex quadratic programming, KKT systems, Schur-complement method, variable-reduction method.

*Research supported in part by National Science Foundation grants DMS-0915220 and DMS-1318480, and by Department of Energy grant DE-SC0002349.

†Department of Mathematics, University of California, San Diego, La Jolla, CA 92093-0112 (pgill@ucsd.edu, elwong@ucsd.edu).

1. Introduction

A *quadratic program* (QP) involves the minimization or maximization of a quadratic objective function subject to linear equality and inequality constraints on the variables. QPs arise in many areas, including economics, applied science and engineering. Important applications include portfolio analysis, support vector machines, structural analysis and optimal control. Quadratic programming also forms a principal computational component of many sequential quadratic programming methods for nonlinear programming (for a recent survey, see Gill and Wong [34]). Interior methods and active-set methods are two alternative approaches to handling the inequality constraints of a QP. In this paper we focus on active-set methods, which have the property that they are able to capitalize on a good estimate of the solution. In particular, if a sequence of related QPs must be solved, then the solution of one problem may be used to "warm start" the next, which can significantly reduce the amount of computation time. This feature makes active-set quadratic programming methods particularly effective in the final stages of sequential quadratic programming method.

In the first part of the paper (comprising Sections 2 and 3), we consider the formulation and analysis of an active-set method for a generic QP of the form

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & \varphi(x) = c^Tx + \frac{1}{2}x^THx \\ \text{subject to} & Ax = b, \quad Dx \geq f, \end{array} \tag{1.1}$$

where A, b, c, D, f and H are constant, H is symmetric, A is $m \times n$, and D is $m_D \times n$. (In order to simplify the notation, it is assumed that the inequalities involve only lower bounds. However, the method to be described can be generalized to treat all forms of linear constraints.) No assumptions are made about H (other than symmetry), which implies that the objective function $\varphi(x)$ need not be convex. In the nonconvex case, however, convergence will be to a point satisfying the second-order necessary conditions for optimality, which may or may not be a local minimizer (for more details, see Section 2.1). The method under consideration defines a primal-dual search pair associated with the solution of an equality-constrained subproblem involving a "working set" of linearly independent constraints. Unlike existing quadratic programming methods, the working set may include constraints that need not be active at the current iterate. In this context, we reformulate a method for a general QP that was first proposed by Fletcher [20], and modified subsequently by Gould [37]. In this reformulation, the primal-dual search directions satisfy a KKT system of equations formed from the Hessian H and the gradients of the constraints in the working set. The working set is specified by an active-set strategy that controls the inertia (i.e., the number of positive, negative and zero eigenvalues) of the KKT matrix. It is shown in Section 3 that this inertia-controlling strategy guarantees that each set of KKT equations is well-defined and nonsingular. In addition, it is shown that, under certain circumstances, the solution of this KKT system may be updated using a simple recurrence relation, thereby giving a significant reduction in the number of KKT systems that need to be solved. (For conventional inertia-controlling methods that use a working set of active constraints, see, e.g., Gill and Murray [25], and Gill, Murray, Saunders and Wright [31, 32].)

Not all active-set methods for a general QP are inertia controlling—see, for example, the methods of Bunch and Kaufman [7], Friedlander and Leyffer [22], and the quadratic programming methods in the `GALAHAD` software package of Gould, Orban, and Toint [40, 41, 38]. A number of alternative methods have been proposed for strictly convex quadratic programming with a modest number of constraints and variables, see, e.g., Goldfarb and Idnani [35], Gill et al. [24], and Powell [52]. A variable-reduction method for a large-scale convex QP is proposed by Gill, Murray and Saunders [27]. Bartlett and Biegler [3] propose a fixed-factorization method for large-scale strictly convex problems (see Section 5.2).

Sections 4–7 form the second part of the paper, which focuses on a method for QPs with constraints written in standard form. In this case, the inequality constraints of the generic form (1.1) are nonnegativity constraints $x \geq 0$. It is shown that if $H = 0$ (so that the problem has a linear objective), then the method is equivalent to a variant of the primal simplex method in which the π-values and reduced costs are updated at each iteration. Section 5 describes two approaches for solving the KKT systems. The first approach is the well-known *variable-reduction method*, which is suitable for problems for which the number of active constraints is comparable to the number of variables (i.e., for problems with a small number of degrees of freedom). The variable-reduction method uses a Cholesky factorization of the reduced Hessian and a sparse LU factorization of a basis matrix. The second approach, which we call the *block-LU method*, uses a sparse factorization of a fixed indefinite KKT matrix in conjunction with the factorization of a smaller dense matrix that is updated at each iteration (see also, Gill et al. [28] and Huynh [45]). The use of a fixed factorization allows a "black-box" sparse equation solver to be used repeatedly. This feature makes the block-LU method ideally suited to problems with structure that can be exploited by using specialized factorization. Moreover, improvements in efficiency derived from exploiting new parallel and vector computer architectures are immediately applicable via state-of-the-art linear equation solvers. Section 6 describes how an appropriate initial basis is found when the problem is not strictly convex. Finally, in Section 7 we describe the main features of the Fortran 2008 package `SQIC` (Sparse Quadratic programming using Inertia Control), which is a particular implementation of the method for standard form QPs described in Section 4. Numerical results are given for all the linear and quadratic programs in the CUTEst test collection (see [39]).

Notation. The gradient of the objective φ evaluated at x, $c + Hx$, is denoted by the vector $g(x)$, or g if it is clear where the evaluation occurs. The vector d_i^T refers to the i-th row of the constraint matrix D, so that the i-th inequality constraint is $d_i^T x \geq f_i$. The i-th component of a vector labeled with a subscript will be denoted by $[\,\cdot\,]_i$, e.g., $[v_N]_i$ is the i-th component of the vector v_N. Similarly, a subvector of components with indices in the index set $\mathcal{S}$ is denoted by $(\,\cdot\,)_{\mathcal{S}}$, e.g., $(v_N)_{\mathcal{S}}$ is the vector with components $[v_N]_i$ for $i \in \mathcal{S}$. The symbol I is used to denote an identity matrix with dimension determined by the context. The j-th column of I is denoted by e_j. Unless explicitly indicated otherwise, $\|\cdot\|$ denotes the vector two-norm or its induced matrix norm. The inertia of a real symmetric matrix A, denoted by $\text{In}(A)$, is the integer triple (a_+, a_-, a_0) giving the number of positive, negative and zero eigenvalues of A. Given vectors a and b with the same dimension, the vector with i-th component $a_i b_i$ is denoted by $a \cdot b$. Given a symmetric matrix K of the form $\left(\begin{smallmatrix} M & N^T \\ N & G \end{smallmatrix}\right)$, with M nonsingular, the matrix $G - NM^{-1}N^T$, the Schur complement of M in K, will be denoted by K/M. When the definitions of the relevant matrices are clear we will refer to "the" Schur complement.

2. Background

In this section, we review the optimality conditions for the generic QP (1.1), and describe a framework for the formulation of feasible-point active-set QP methods. Throughout, it is assumed that the matrix A has full row-rank m. This condition is easily satisfied for the class of active-set methods considered in this paper. Given an arbitrary matrix G, equality constraints $Gu = b$ are equivalent to the full rank constraints $Gu + v = b$, if we impose $v = 0$. In this formulation, the v-variables are artificial variables that are fixed at zero.

2.1. Optimality conditions

The necessary and sufficient conditions for a local solution of the QP (1.1) involve the existence of vectors z and π of Lagrange multipliers associated with the constraints $Dx \geq f$ and $Ax = b$, respectively. The conditions are summarized by the following result, which is stated without proof (see, e.g., Borwein [6], Contesse [8] and Majthay [47]).

Result 2.1. (QP optimality conditions) *The point x is a local minimizer of the quadratic program (1.1) if and only if*

(a) *$Ax = b$, $Dx \geq f$, and there exists at least one pair of vectors π and z such that $g(x) = A^T\pi + D^Tz$, with $z \geq 0$, and $z \cdot (Dx - f) = 0$;*

(b) *$p^THp \geq 0$ for all nonzero p satisfying $g(x)^Tp = 0$, $Ap = 0$, and $d_i^Tp \geq 0$ for every i such that $d_i^Tx = f_i$.* ∎

We follow the convention of refering to any x that satisfies condition (a) as a first-order KKT point.

If H has at least one negative eigenvalue and (x, π, z) satisfies condition (a) with an index i such that $z_i = 0$ and $d_i^Tx = f_i$, then x is known as a dead point. Verifying condition (b) at a dead point requires finding the global minimizer of an indefinite quadratic form over a cone, which is an NP-hard problem (see, e.g., Cottle, Habetler and Lemke [9], Murty and Kabadi [48], Pardalos and Schnitger [50], and Pardalos and Vavasis [51]). This implies that the optimality of a candidate solution of a general quadratic program can be verified only if more restrictive (but computationally tractable) sufficient conditions are satisfied. A dead point is a point at which the sufficient conditions are not satisfied, but certain necessary conditions for optimality hold. Replacing part (b) of Result 2.1 with the condition that $p^THp \geq 0$ for all nonzero p satisfying $Ap = 0$, and $d_i^Tp = 0$ for each i such that $d_i^Tx = f_i$, leads to computationally tractable necessary conditions for optimality.

Additionally, suitable sufficient conditions for optimality are given by replacing the necessary condition by the condition that $p^THp \geq 0$ for all p such that $Ap = 0$, and $d_i^Tp = 0$ for every $i \in \mathcal{A}_+(x)$, where $\mathcal{A}_+(x)$ is the index set $\mathcal{A}_+(x) = \{i : d_i^Tx = f_i \text{ and } z_i > 0\}$.

These conditions may be expressed in terms of the constraints that are satisfied with equality at x. Let x be any point satisfying the equality constraints $Ax = b$. (The assumption that A has rank m implies that there must exist at least one such x.) An inequality constraint is active at x if it is satisfied with equality. The indices associated with the active constraints comprise the active set, denoted by $\mathcal{A}(x)$. An active-constraint matrix $A_{\mathfrak{a}}(x)$ is a matrix with rows consisting of the rows of A and the gradients of the active constraints. By convention, the rows of A are listed first, giving the active-constraint matrix

$$A_{\mathfrak{a}}(x) = \begin{pmatrix} A \\ D_{\mathfrak{a}}(x) \end{pmatrix},$$

where $D_{\mathfrak{a}}(x)$ comprises the rows of D with indices in $\mathcal{A}(x)$. Note that the active-constraint matrix includes A in addition to the gradients of the active constraints. The argument x is generally omitted if it is clear where $D_{\mathfrak{a}}$ is defined.

With this definition of the active set, we give necessary conditions for the QP.

Result 2.2. (Necessary conditions in active-set form) *Let the columns of the matrix $Z_{\mathfrak{a}}$ form a basis for the null space of $A_{\mathfrak{a}}$. The point x is a local minimizer of the QP (1.1) only if*

(a) *x is a first-order KKT point, i.e.,* (i) $Ax = b$, $Dx \geq f$; (ii) $g(x)$ *lies in* range$(A_{\mathfrak{a}}^T)$, *or equivalently, there exist vectors* π *and* $z_{\mathfrak{a}}$ *such that* $g(x) = A^T\pi + D_{\mathfrak{a}}^T z_{\mathfrak{a}}$; *and* (iii) $z_{\mathfrak{a}} \geq 0$,

(b) *the reduced Hessian* $Z_{\mathfrak{a}}^T H Z_{\mathfrak{a}}$ *is positive semidefinite.* ∎

Typically, software for general quadratic programming will terminate the iterations at a dead point. Nevertheless, it is possible to define procedures that check for optimality at a dead point, even though the chance of success in a reasonable amount of computation time will depend on the size of the problem (see Forsgren, Gill and Murray [21]).

2.2. Active-set methods

The method to be considered is a two-phase active-set method. In the first phase (the feasibility phase or phase 1), the objective is ignored while a feasible point is found for the constraints $Ax = b$ and $Dx \geq f$. In the second phase (the optimality phase or phase 2), the objective is minimized while feasibility is maintained. Given a feasible x_0, active-set methods compute a sequence of feasible iterates $\{x_k\}$ such that $x_{k+1} = x_k + \alpha_k p_k$ and $\varphi(x_{k+1}) \leq \varphi(x_k)$, where p_k is a nonzero search direction and α_k is a nonnegative step length. Active-set methods are motivated by the main result of Farkas' Lemma, which states that a feasible x must either satisfy the first-order optimality conditions or be the starting point of a feasible descent direction, i.e., a direction p such that

$$A_{\mathfrak{a}}\, p \geq 0 \quad \text{and} \quad g(x)^T p < 0. \tag{2.1}$$

The method considered in this paper approximates the active set by a working set $\mathcal{W}$ of row indices of D. The working set has the form $\mathcal{W} = \{\nu_1, \nu_2, \ldots, \nu_{m_w}\}$, where m_w is the number of indices in $\mathcal{W}$. Analogous to the active-constraint matrix $A_{\mathfrak{a}}$, the $(m+m_w)\times n$ working-set matrix A_w contains the gradients of the equality constraints and inequality constraints in $\mathcal{W}$. The structure of the working-set matrix is similar to that of the active-set matrix, i.e.,

$$A_w = \begin{pmatrix} A \\ D_w \end{pmatrix},$$

where D_w is a matrix formed from the m_w rows of D with indices in $\mathcal{W}$. The vector f_w denotes the components of f with indices in $\mathcal{W}$.

There are two important distinctions between the definitions of $\mathcal{A}$ and $\mathcal{W}$.

(i) The indices of $\mathcal{W}$ define a subset of the rows of D that are linearly independent of the rows of A, i.e., the working-set matrix A_w has full row rank. It follows that m_w must satisfy $0 \leq m_w \leq \min\{n - m, m_D\}$.

(ii) The active set $\mathcal{A}$ is uniquely defined at any feasible x, whereas there may be many choices for $\mathcal{W}$. The set $\mathcal{W}$ is determined by the properties of a particular active-set method.

Conventional active-set methods define the working set as a subset of the active set (see, e.g., Gill, Murray and Wright [33], and Nocedal and Wright [49]). In this paper we relax this requirement—in particular, a working-set constraint need not be strictly satisfied at x. (More generally, a working-set constraint need not be feasible at x, although this property is not used here).

Given a working set $\mathcal{W}$ and an associated working-set matrix A_w at x, we introduce the notions of stationarity and optimality with respect to a working set. We emphasize that the definitions below do not require that the working-set constraints are active (or even feasible) at x.

Definition 2.1. (Subspace stationary point) *Let $\mathcal{W}$ be a working set defined at an x such that $Ax = b$. Then x is a* subspace stationary point with respect to $\mathcal{W}$ (or, equivalently, with respect to A_w) *if $g \in \text{range}(A_w^T)$, i.e., there exists a vector y such that $g = A_w^T y$. Equivalently, x is a subspace stationary point with respect to the working set $\mathcal{W}$ if the reduced gradient $Z_w^T g$ is zero, where the columns of Z_w form a basis for the null space of A_w.* ∎

At a subspace stationary point, the components of y are the Lagrange multipliers associated with a QP with equality constraints $Ax = b$ and $D_w x = f_w$. To be consistent with the optimality conditions of Result 2.2, we denote the first m components of y as π (the multipliers associated with $Ax = b$) and the last m_w components of y as z_w (the multipliers associated with the constraints in $\mathcal{W}$). With this notation, the identity $g(x) = A_w^T y = A^T\pi + D_w^T z_w$ holds at a subspace stationary point.

To classify subspace stationary points based on curvature information, we define the terms *second-order-consistent working set* and *subspace minimizer*.

Definition 2.2. (Second-order-consistent working set) *Let $\mathcal{W}$ be a working set associated with an x such that $Ax = b$, and let the columns of Z_w form a basis for the null space of A_w. The working set $\mathcal{W}$ is* second-order-consistent *if the reduced Hessian $Z_w^T H Z_w$ is positive definite.* ∎

The inertia of the reduced Hessian is related to the inertia of the $(n+m+m_w)\times(n+m+m_w)$ KKT matrix $K = \begin{pmatrix} H & A_w^T \\ A_w & \end{pmatrix}$ through the identity $\text{In}(K) = \text{In}(Z_w^T H Z_w) + (m + m_w, m + m_w, 0)$ (see Gould [36]). It follows that an equivalent characterization of a second-order-consistent working set is that K has inertia $(n, m + m_w, 0)$. A KKT matrix K associated with a second-order-consistent working set is said to have "correct inertia". It is always possible to impose sufficiently many temporary constraints that will covert a given working set into a second-order consistent working set. For example, a temporary vertex formed by fixing variables at their current values will always provide a KKT matrix with correct inertia (see Section 6 for more details).

Definition 2.3. (Subspace minimizer) *If x is a subspace stationary point with respect to a second-order-consistent basis $\mathcal{W}$, then x is known as a* subspace minimizer with respect to $\mathcal{W}$. *If every constraint in the working set is active, then x is called a* standard subspace minimizer*; otherwise x is called a* nonstandard subspace minimizer. ∎

3. A Method for the Generic Quadratic Program

In this section we formulate and analyze an active-set method based on controlling the inertia of the KKT matrix. Inertia-controlling methods were first proposed by Fletcher [20] and are based on the simple rule that a constraint is removed from the working set only at a *subspace minimizer*. We show that with an appropriate choice of initial point, this rule ensures that every iterate is a subspace minimizer for the associated working set. This allows for the reliable and efficient calculation of the search directions.

The method starts at a subspace minimizer x with $g(x) = A_w^T y = A^T\pi + D_w^T z_w$ and a KKT matrix with correct inertia. If x is standard and $z_w \geq 0$, then x is optimal for the QP. Otherwise, there exists an index $\nu_s \in \mathcal{W}$ such that $[z_w]_s < 0$. To proceed, we define

a descent direction that is feasible for the equality constraints and the constraints in the working set. Analogous to (2.1), p is defined so that

$$A_w p = e_{m+s} \quad \text{and} \quad g(x)^T p < 0.$$

We call any vector satisfying this condition a *nonbinding direction* because any nonzero step along it will increase the residual of the ν_s-th inequality constraint (and hence make it inactive or nonbinding). Here we define p as the solution of the equality-constrained subproblem

$$\underset{p}{\text{minimize}} \;\; \varphi(x+p) \quad \text{subject to} \quad A_w p = e_{m+s}. \tag{3.1}$$

The optimality conditions for this subproblem imply the existence of a vector q such that $g(x+p) = A_w^T(y+q)$; i.e., q is the step to the multipliers associated with the optimal solution $x+p$. This condition, along with the feasibility condition, implies that p and q satisfy the equations

$$\begin{pmatrix} H & A_w^T \\ A_w & \end{pmatrix} \begin{pmatrix} p \\ -q \end{pmatrix} = \begin{pmatrix} -(g(x) - A_w^T y) \\ e_{m+s} \end{pmatrix}. \tag{3.2}$$

The primal and dual directions have a number of important properties that are summarized in the next result.

Result 3.1. (Properties of the search direction) *Let x be a subspace minimizer such that $g = A_w^T y = A^T\pi + D_w^T z_w$, with $[z_w]_s < 0$. Then the vectors p and q satisfying the equations*

$$\begin{pmatrix} H & A_w^T \\ A_w & \end{pmatrix} \begin{pmatrix} p \\ -q \end{pmatrix} = \begin{pmatrix} -(g(x) - A_w^T y) \\ e_{m+s} \end{pmatrix} = \begin{pmatrix} 0 \\ e_{m+s} \end{pmatrix} \tag{3.3}$$

constitute the unique primal and dual solutions of the equality constrained problem defined by minimizing $\varphi(x+p)$ subject to $A_w p = e_{m+s}$. Moreover, p and q satisfy the identities

$$g^T p = y_{m+s} = [z_w]_s \quad \textit{and} \quad p^T H p = q_{m+s} = [q_w]_s, \tag{3.4}$$

where q_w denotes the vector consisting of the last m_w components of q.

Proof. The assumption that x is a subspace minimizer implies that the subproblem has a unique bounded minimizer. The optimality of p and q follows from the equations in (3.2), which represent the feasibility and optimality conditions for the minimization of $\varphi(x+p)$ on the set $\{p : A_w p = e_{m+s}\}$. The equation $g = A_w^T y$ and the definition of p from (3.3) give

$$g^T p = p^T(A_w^T y) = y^T A_w p = y^T e_{m+s} = y_{m+s} = [z_w]_s$$

Similarly, $p^T H p = p^T(A_w^T q) = e_{m+s}^T q = q_{m+s} = [q_w]_s$. ∎

Once p and q are known, a nonnegative step α is computed so that $x + \alpha p$ is feasible and $\varphi(x+\alpha p) \le \varphi(x)$. If $p^T H p > 0$, the step that minimizes $\varphi(x+\alpha p)$ as a function of α is given by $\alpha_* = -g^T p / p^T H p$. The identities (3.4) give

$$\alpha_* = -g^T p / p^T H p = -[z_w]_s / [q_w]_s.$$

As $[z_w]_s < 0$, if $[q_w]_s = p^T H p > 0$, the optimal step α_* is positive. Otherwise $[q_w]_s = p^T H p \le 0$ and φ has no bounded minimizer along p and $\alpha_* = +\infty$.

If $x + \alpha_* p$ is unbounded or infeasible, then α must be limited by α_F, the *maximum feasible step* from x along p. The feasible step is defined as $\alpha_F = \gamma_r$, where

$$\gamma_r = \min \gamma_i, \quad \text{with} \quad \gamma_i = \begin{cases} \dfrac{d_i^T x - f_i}{-d_i^T p} & \text{if } d_i^T p < 0; \\ +\infty & \text{otherwise.} \end{cases}$$

The step α is then $\min\{\alpha_*, \alpha_F\}$. If $\alpha = +\infty$, the QP has no bounded solution and the algorithm terminates. In the discussion below, we assume that α is a bounded step.

The primal and dual directions p and q defined by (3.3) have the property that $x + \alpha p$ remains a subspace minimizer with respect to A_w *for any step* α. This follows from the definitions (3.3), which imply that

$$g(x + \alpha p) = g(x) + \alpha Hp = A_w^T y + \alpha A_w^T q = A_w^T (y + \alpha q), \tag{3.5}$$

so that the gradient at $x + \alpha p$ is a linear combination of the columns of A_w^T. The step $x + \alpha p$ does not change the KKT matrix K associated with the subspace minimizer x, which implies that $x + \alpha p$ is also a subspace minimizer with respect to A_w. This means that $x + \alpha p$ may be interpreted as the solution of a problem in which the working-set constraint $d_{\nu_s}^T x \geq f_{\nu_s}$ is *shifted* to pass through $x + \alpha p$. The component $[y + \alpha q]_{m+s} = [z_w + \alpha q_w]_s$ is the Lagrange multiplier associated with the shifted version of $d_{\nu_s}^T x \geq f_{\nu_s}$. This property is known as the *parallel subspace property* of quadratic programming. It shows that if x is stationary with respect to a nonbinding constraint, then it remains so for all subsequent iterates for which that constraint remains in the working set. (The parallel subspace property forms the principal basis of a number of other active-set methods, including the parametric QP methods of Best [4] and `qpOASES` [18, 19].)

Once α has been defined, the new iterate is $\bar{x} = x + \alpha p$. The composition of the new working set and multipliers depends on the definition of α.

Case 1: $\alpha = \alpha_*$ In this case, the step $\alpha = \alpha_* = -[z_w]_s / [q_w]_s$ minimizes $\varphi(x + \alpha p)$ with respect to α, giving the s-th element of $z_w + \alpha q_w$ as

$$[z_w + \alpha q_w]_s = [z_w]_s + \alpha_* [q_w]_s = 0,$$

which implies that the Lagrange multiplier associated with the shifted constraint is *zero* at $\bar{x}$. The nature of the stationarity may be determined using the next result.

Result 3.2. (Constraint deletion) *Let x be a subspace minimizer with respect to $\mathcal{W}$. Assume that $[z_w]_s < 0$. Let $\bar{x}$ denote the point $x + \alpha p$, where p is defined by (3.3) and $\alpha = \alpha_*$ is bounded. Then $\bar{x}$ is a subspace minimizer with respect to $\bar{\mathcal{W}} = \mathcal{W} - \{\nu_s\}$.*

Proof. Let K and $\bar{K}$ denote the matrices

$$K = \begin{pmatrix} H & A_w^T \\ A_w & \end{pmatrix} \quad \text{and} \quad \bar{K} = \begin{pmatrix} H & \bar{A}_w^T \\ \bar{A}_w & \end{pmatrix},$$

where A_w and $\bar{A}_w$ are the working-set matrices associated with $\mathcal{W}$ and $\bar{\mathcal{W}}$. It suffices to show that $\bar{K}$ has the correct inertia, i.e., $\text{In}(\bar{K}) = (n, m + m_w - 1, 0)$.

Consider the matrix M such that

$$M \triangleq \begin{pmatrix} K & e_{m+n+s} \\ e_{m+n+s}^T & \end{pmatrix}.$$

By assumption, x is a subspace minimizer with $\text{In}(K) = (n, m + m_w, 0)$. In particular, K is nonsingular and the Schur complement of K in M exists with

$$M/K = -e_{n+m+s}^T K^{-1} e_{n+m+s} = -e_{n+m+s}^T \begin{pmatrix} p \\ -q \end{pmatrix} = [q_w]_s.$$

It follows that

$$\text{In}(M) = \text{In}(M/K) + \text{In}(K) = \text{In}([q_w]_s) + (n, m + m_w, 0). \tag{3.6}$$

Now consider a symmetrically permuted version of M:

$$\widetilde{M} = \begin{pmatrix} 0 & 1 & & \\ 1 & 0 & d_{\nu_s}^T & \\ & d_{\nu_s} & H & \bar{A}_w^T \\ & & \bar{A}_w & \end{pmatrix}.$$

Inertia is unchanged by symmetric permutations, so $\text{In}(M) = \text{In}(\widetilde{M})$. The 2×2 block in the upper-left corner of $\widetilde{M}$, denoted by E, has eigenvalues ± 1, so that

$$\text{In}(E) = (1, 1, 0) \quad \text{with} \quad E^{-1} = E.$$

The Schur complement of E in $\widetilde{M}$ is

$$\widetilde{M}/E = \bar{K} - \begin{pmatrix} 0 & d_{\nu_s} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ d_{\nu_s}^T & 0 \end{pmatrix} = \bar{K},$$

which implies that $\text{In}(\widetilde{M}) = \text{In}(\widetilde{M}/E) + \text{In}(E) = \text{In}(\bar{K}) + (1, 1, 0)$. Combining this with (3.6) yields

$$\begin{aligned} \text{In}(\bar{K}) &= \text{In}([q_w]_s) + (n, m + m_w, 0) - (1, 1, 0) \\ &= \text{In}([q_w]_s) + (n - 1, m + m_w - 1, 0). \end{aligned}$$

As $\alpha = \alpha_*$, the scalar $[q_w]_s$ must be positive. It follows that

$$\text{In}(\bar{K}) = (1, 0, 0) + (n - 1, m + m_w - 1, 0) = (n, m + m_w - 1, 0)$$

and the subspace stationary point $\bar{x}$ is a (standard) subspace minimizer with respect to the new working set $\bar{\mathcal{W}} = \mathcal{W} - \{\nu_s\}$. ∎

Case 2: $\alpha = \alpha_F$ In this case, α is the step to the blocking constraint $d_r^T x \geq f_r$, which is eligible to be added to the working set at $x + \alpha p$. However, the definition of the new working set depends on whether or not the blocking constraint is dependent on the constraints already in $\mathcal{W}$. If d_r is linearly independent of the columns of A_w^T, then the index r is added to the working set. Otherwise, we show in Result 3.5 below that a suitable working set is defined by exchanging rows d_{ν_s} and d_r in A_w. The following result provides a computable test for the independence of d_r and the columns of A_w^T.

Result 3.3. (Test for constraint dependency) *Let x be a subspace minimizer with respect to A_w. Assume that $d_r^T x \geq f_r$ is a blocking constraint at $\bar{x} = x + \alpha p$, where p satisfies (3.3). Define vectors u and v such that*

$$\begin{pmatrix} H & A_w^T \\ A_w & \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} d_r \\ 0 \end{pmatrix}, \tag{3.7}$$

then

(a) *d_r and the columns of A_w^T are linearly independent if and only if $u \neq 0$;*

(b) *$v_{m+s} = d_r^T p < 0$, and $u^T d_r \geq 0$ with $u^T d_r > 0$ if $u \neq 0$.*

Proof. For part (a), equations (3.7) give $Hu + A_w^T v = d_r$ and $A_w u = 0$. If $u = 0$ then $A_w^T v = d_r$, and d_r must be dependent on the columns of A_w^T. Conversely, if $A_w^T v = d_r$, then the definition of u gives $u^T A_w^T v = u^T d_r = 0$, which implies that $u^T H u = u^T(Hu + A_w^T v) = u^T d_r = 0$. By assumption, x is a subspace minimizer with respect to A_w, which is equivalent to the assumption that H is positive definite for all u such that $A_w u = 0$. Hence $u^T H u = 0$ can hold only if u is zero.

For part (b), we use equations (3.3) and (3.7) to show that

$$v_{m+s} = e_{m+s}^T v = p^T A_w^T v = p^T(d_r - Hu) = p^T d_r - q^T A_w u = d_r^T p < 0,$$

where the final inequality follows from the fact that $d_r^T p$ must be negative if $d_r^T x \geq f_r$ is a blocking constraint. If $u \neq 0$, equations (3.7) imply $Hu + A_w^T v = d_r$ and $A_w u = 0$. Multiplying the first equation by u^T and applying the second equation gives $u^T H u = u^T d_r$. As $u \in \text{null}(A_w)$ and x is a subspace minimizer, it must hold that $u^T H u = u^T d_r > 0$, as required. ∎

The next result provides expressions for the updated multipliers.

Result 3.4. (Multiplier updates) *Assume that x is a subspace minimizer with respect to A_w. Assume that $d_r^T x \geq f_r$ is a blocking constraint at the next iterate $\bar{x} = x + \alpha p$, where the direction p satisfies (3.3). Let u and v satisfy (3.7).*

(a) *If d_r and the columns of A_w^T are linearly independent, then the vector $\bar{y}$ formed by appending a zero component to the vector $y + \alpha q$ satisfies $g(\bar{x}) = \bar{A}_w^T \bar{y}$, where $\bar{A}_w$ denotes the matrix A_w with row d_r^T added in the last position.*

(b) *If d_r and the columns of A_w^T are linearly dependent, then the vector $\bar{y}$ such that*

$$\bar{y} = y + \alpha q - \sigma v, \quad \text{with} \quad \sigma = [y + \alpha q]_{m+s}/v_{m+s}, \tag{3.8}$$

satisfies $g(\bar{x}) = A_w^T \bar{y} + \sigma d_r$ with $\bar{y}_{m+s} = 0$ and $\sigma > 0$.

Proof. For part (a), the parallel subspace property (3.5) implies that $g(x + \alpha p) = g(\bar{x}) = A_w^T(y + \alpha q)$. As d_r and the columns of A_w^T are linearly independent, we may add the index r to $\mathcal{W}$ and define the new working-set matrix $\bar{A}_w^T = \begin{pmatrix} A_w^T & d_r \end{pmatrix}$. This allows us to write $g(\bar{x}) = \bar{A}_w^T \bar{y}$, with $\bar{y}$ given by $y + \alpha q$ with an appended zero component.

Now assume that A_w^T and d_r are linearly dependent. From Result 3.3 it must hold that $u = 0$ and there exists a unique v such that $d_r = A_w^T v$. For any value of σ, the parallel subspace property (3.5) gives

$$g(\bar{x}) = A_w^T(y + \alpha q) = A_w^T(y + \alpha q - \sigma v) + \sigma d_r.$$

If we choose $\sigma = [y + \alpha q]_{m+s}/v_{m+s}$ and define the vector $\bar{y} = y + \alpha q - \sigma v$, then

$$g(\bar{x}) = A_w^T \bar{y} + \sigma d_r, \quad \text{with} \quad \bar{y}_{m+s} = [y + \alpha q - \sigma v]_{m+s} = 0.$$

It follows that $g(\bar{x})$ is a linear combination of d_r and every column of A_w^T except d_s.

In order to show that $\sigma = [y + \alpha q]_{m+s}/v_{m+s}$ is positive, we consider the linear function $y_{m+s}(\alpha) = [y + \alpha q]_{m+s}$, which satisfies $y_{m+s}(0) = y_{m+s} < 0$. If $q_{m+s} = p^T H p > 0$, then $\alpha_* < \infty$ and $y_{m+s}(\alpha)$ is an increasing linear function of positive α with $y_{m+s}(\alpha_*) = 0$. This implies that $y_{m+s}(\alpha) < 0$ for any $\alpha < \alpha_*$ and $y_{m+s}(\alpha_k) < 0$. If $q_{m+s} \leq 0$, then $y_{m+s}(\alpha)$ is a nonincreasing linear function of α so that $y_{m+s}(\alpha) < 0$ for any positive α. Thus, $[y + \alpha q]_{m+s} < 0$ for any $\alpha < \alpha_*$, and $\sigma = [y + \alpha q]_{m+s}/v_{m+s} > 0$ from part (b) of Result 3.3. ∎

Result 3.5. *Let x be a subspace minimizer with respect to the working set $\mathcal{W}$. Assume that $d_r^T x \geq f_r$ is a blocking constraint at $\bar{x} = x + \alpha p$, where p is defined by (3.3).*

(a) *If d_r is linearly independent of the columns of A_w^T, then $\bar{x}$ is a subspace minimizer with respect to the working set $\bar{\mathcal{W}} = \mathcal{W} + \{r\}$.*

(b) *If d_r is linearly dependent on the columns of A_w^T, then $\bar{x}$ is a subspace minimizer with respect to the working set $\bar{\mathcal{W}} = \mathcal{W} + \{r\} - \{\nu_s\}$.*

Proof. Parts (a) and (b) of Result 3.4 imply that $\bar{x}$ is a subspace stationary point with respect to $\bar{\mathcal{W}}$. It remains to show that in each case, the new working sets are second-order-consistent.

For part (a), the new KKT matrix for the new working set $\bar{\mathcal{W}} = \mathcal{W} + \{r\}$ must have inertia $(n, m+m_w+1, 0)$. Assume that d_r and the columns of A_w^T are linearly independent, so that the vector u of (3.7) is nonzero. Let K and $\bar{K}$ denote the KKT matrices associated with the working sets $\mathcal{W}$ and $\bar{\mathcal{W}}$, i.e.,

$$K = \begin{pmatrix} H & A_w^T \\ A_w & \end{pmatrix} \quad \text{and} \quad \bar{K} = \begin{pmatrix} H & \bar{A}_w^T \\ \bar{A}_w & \end{pmatrix},$$

where $\bar{A}_w$ is the matrix A_w with the row d_r^T added in the last position.

By assumption, x is a subspace minimizer and $\text{In}(K) = (n, m+m_w, 0)$. It follows that K is nonsingular and the Schur complement of K in $\bar{K}$ exists with

$$\bar{K}/K = -\begin{pmatrix} d_r \\ 0 \end{pmatrix}^T K^{-1} \begin{pmatrix} d_r \\ 0 \end{pmatrix} = -\begin{pmatrix} d_r^T & 0 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = -d_r^T u < 0,$$

where the last inequality follows from part (b) of Result 3.3. Then,

$$\begin{aligned} \text{In}(\bar{K}) = \text{In}(\bar{K}/K) + \text{In}(K) &= \text{In}(-u^T d_r) + (n, m+m_w, 0) \\ &= (0,1,0) + (n, m+m_w, 0) = (n, m+m_w+1, 0). \end{aligned}$$

For part (b), assume that d_r and the columns of A_w^T are linearly dependent and that $\bar{\mathcal{W}} = \mathcal{W} + \{r\} - \{\nu_s\}$. By Result 3.4 and equation (3.7), it must hold that $u = 0$ and $A_w^T v = d_r$. Let A_w and $\bar{A}_w$ be the working-set matrices associated with $\mathcal{W}$ and $\bar{\mathcal{W}}$. The change in the working set replaces row s of D_w by d_r^T, so that

$$\begin{aligned} \bar{A}_w = A_w + e_{m+s}(d_r^T - d_s^T) &= A_w + e_{m+s}(v^T A_w - e_{m+s}^T A_w) \\ &= \big(I_w + e_{m+s}(v - e_{m+s})^T\big) A_w \\ &= M A_w, \end{aligned}$$

where $M = I_w + e_{m+s}(v - e_{m+s})^T$. The matrix M has $m+m_w-1$ unit eigenvalues and one eigenvalue equal to v_{m+s}. From part (b) of Result 3.3, it holds that $v_{m+s} < 0$ and hence M is nonsingular. The new KKT matrix for $\bar{\mathcal{W}}$ can be written as

$$\begin{pmatrix} H & \bar{A}_w^T \\ \bar{A}_w & \end{pmatrix} = \begin{pmatrix} I_n & \\ & M \end{pmatrix} \begin{pmatrix} H & A_w^T \\ A_w & \end{pmatrix} \begin{pmatrix} I_n & \\ & M^T \end{pmatrix}.$$

By Sylvester's Law of Inertia, the old and new KKT matrices have the same inertia, which implies that $\bar{x}$ is a subspace minimizer with respect to $\bar{\mathcal{W}}$. ∎

The first part of this result shows that $\bar{x}$ is a subspace minimizer both before and after an independent constraint is added to the working set. This is crucial because it means

that the directions p and q for the next iteration satisfy the KKT equations (3.3) with $\bar{A}_w$ in place of A_w. The second part shows that the working-set constraints can be linearly dependent only at a *standard* subspace minimizer associated with a working set that does not include constraint ν_s. This implies that it is appropriate to remove ν_s from the working set. The constraint $d_{\nu_s}^T x \geq f_{\nu_s}$ plays a significant (and explicit) role in the definition of the search direction and is called the *nonbinding working-set constraint*. The method generates sets of consecutive iterates that begin and end with a standard subspace minimizer. The nonbinding working-set constraint $d_{\nu_s}^T x \geq f_{\nu_s}$ identified at the first point of the sequence is deleted from the working set at the last point (either by deletion or replacement).

Each iteration requires the solution of two KKT systems:

$$\text{Full System 1:} \qquad \begin{pmatrix} H & A_w^T \\ A_w & 0 \end{pmatrix} \begin{pmatrix} p \\ -q \end{pmatrix} = \begin{pmatrix} 0 \\ e_{m+s} \end{pmatrix} \tag{3.9a}$$

$$\text{Full System 2:} \qquad \begin{pmatrix} H & A_w^T \\ A_w & 0 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} d_r \\ 0 \end{pmatrix}. \tag{3.9b}$$

However, for those iterations for which the number of constraints in the working set increases, it is possible to *update* the vectors p and q, making it unnecessary to solve (3.9a).

Result 3.6. *Let x be a subspace minimizer with respect to A_w. Assume the vectors p, q, u and v are defined by (3.9). Let d_r be the gradient of a blocking constraint at $\bar{x} = x + \alpha p$ such that d_r is independent of the columns of A_w^T. If $\rho = -d_r^T p / d_r^T u$, then the vectors*

$$\bar{p} = p + \rho u \quad \text{and} \quad \bar{q} = \begin{pmatrix} q - \rho v \\ \rho \end{pmatrix}$$

are well-defined and satisfy

$$\begin{pmatrix} H & \bar{A}_w^T \\ \bar{A}_w & \end{pmatrix} \begin{pmatrix} \bar{p} \\ -\bar{q} \end{pmatrix} = \begin{pmatrix} 0 \\ e_{m+s} \end{pmatrix}, \quad \text{where} \quad \bar{A}_w = \begin{pmatrix} A_w \\ d_r^T \end{pmatrix}. \tag{3.10}$$

Proof. Result 3.3 implies that u is nonzero and that $u^T d_r > 0$ so that ρ is well defined (and strictly positive).

For any scalar ρ, (3.9a) and (3.9b) imply that

$$\begin{pmatrix} H & A_w^T & d_r \\ A_w & & \\ d_r^T & & \end{pmatrix} \begin{pmatrix} p + \rho u \\ -(q - \rho v) \\ -\rho \end{pmatrix} = \begin{pmatrix} 0 \\ e_{m+s} \\ d_r^T p + \rho d_r^T u \end{pmatrix}.$$

If ρ is chosen so that $d_r^T p + \rho d_r^T u = 0$, the last component of the right-hand side is zero, and $\bar{p}$ and $\bar{q}$ satisfy (3.10) as required. ∎

With a suitable nondegeneracy assumption, the algorithm terminates in a finite number of iterations. As the number of constraints is finite, the sequence $\{x_k\}$ must contain a subsequence $\{x_{ik}\}$ of standard subspace minimizers with respect to their working sets $\{\mathcal{W}_{ik}\}$. If the Lagrange multipliers are nonnegative at any of these points, the algorithm terminates with the desired solution. Otherwise, at least one multiplier must be strictly negative, and hence the nondegeneracy assumption implies that $\alpha_F > 0$ at x_{ik}. Thus, $\varphi(x_{ik}) > \varphi(x_{ik} + \alpha_{ik} p_{ik})$, since at each iteration, the direction is defined as a descent direction with $g^T p < 0$. The subsequence $\{x_{ik}\}$ must be finite because the number of subspace minimizers is finite and the strict decrease in $\varphi(x)$ guarantees that no element of $\{x_{ik}\}$ is repeated. The finiteness of the subsequence implies that the number of intermediate iterates must

also be finite. This follows because a constraint is added to the working set (possibly with a zero step) for every intermediate iteration. Eventually, either a nonzero step will be taken, giving a strict decrease in φ, or enough constraints will be added to define a vertex (a trivial subspace minimizer).

4. Quadratic Programs in Standard Form

The inequality constraints of a QP in standard form consist of only simple upper and lower bounds on the variables. Without loss of generality, we consider methods for the standard-form QP

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ \ \varphi(x) = c^Tx + \tfrac{1}{2}x^THx \quad \text{subject to} \quad Ax = b, \quad x \geq 0. \tag{4.1}$$

This is an example of a mixed-constraint problem (1.1) with $D = I_n$ and $f = 0$. In this case, the working-set matrix D_w consists of rows of the identity matrix, and each working-set index i is associated with a variable x_i that is implicitly fixed at its current value. In this situation, as is customary for constraints in standard form, we refer to the working set as the *nonbasic set* $\mathcal{N}$, and denote its elements as $\{\nu_1, \nu_2, \ldots, \nu_{n_N}\}$ with $n_N = m_w$. The complementary set $\mathcal{B}$ of $n_B = n - n_N$ indices that are not in the working set is known as the *basic set*. The elements of the basic set are denoted by $\{\beta_1, \beta_2, \ldots, \beta_{n_B}\}$.

If P_N denotes the matrix of unit columns $\{\, e_i \,\}$ with $i \in \mathcal{N}$, then the working-set matrix A_w may be written as:

$$A_w = \begin{pmatrix} A \\ P_N^T \end{pmatrix}.$$

Similarly, if P_B is the matrix with unit columns $\{\, e_i \,\}$ with $i \in \mathcal{B}$, then $P = \begin{pmatrix} P_B & P_N \end{pmatrix}$ is a permutation matrix that permutes the columns of A_w as

$$A_w \begin{pmatrix} P_B & P_N \end{pmatrix} = A_wP = \begin{pmatrix} A \\ P_N^T \end{pmatrix} P = \begin{pmatrix} AP \\ P_N^TP \end{pmatrix} = \begin{pmatrix} A_B & A_N \\ & I_{n_N} \end{pmatrix},$$

where A_B and A_N are matrices with columns $\{\, a_{\beta_j} \,\}$ and $\{\, a_{\nu_j} \,\}$ respectively. If y is any n-vector, y_B (the *basic components of* y) denotes the n_B-vector whose j-th component is component β_j of y, and y_N (the *nonbasic components of* y) denotes the n_N-vector whose j-th component is component ν_j of y. We use the same convention for matrices, with the exception of I_B and I_N, which are reserved for the identity matrices of order n_B and n_N, respectively. With this notation, the effect of P on the Hessian and working-set matrix may be written as

$$P^THP = \begin{pmatrix} H_B & H_D \\ H_D^T & H_N \end{pmatrix}, \quad \text{and} \quad A_wP = \begin{pmatrix} A_B & A_N \\ & I_N \end{pmatrix}. \tag{4.2}$$

As in the generic mixed-constraint formulation, A_w must have full row-rank. This is equivalent to requiring that A_B has full row-rank since $\operatorname{rank}(A_w) = n_N + \operatorname{rank}(A_B)$.

For constraints in standard form, we say that x is a subspace minimizer with respect to the basic set $\mathcal{B}$ (or, equivalently, with respect to A_B). Similarly, a second-order-consistent working set is redefined as a *second-order-consistent basis*.

Result 4.1. (Subspace minimizer for standard form) *Let x be a feasible point with basic set $\mathcal{B}$. Let the columns of Z_B form a basis for the null space of A_B.*

(a) *If x is a subspace stationary point with respect to A_w, then there exists a vector π such that $g_B = A_B^T\pi$, or equivalently, $Z_B^Tg_B = 0$.*

(b) *If $\mathcal{B}$ is a second-order-consistent basis, then $Z_B^T H_B Z_B$ is positive definite. Equivalently, the KKT matrix $K_B = \begin{pmatrix} H_B & A_B^T \\ A_B & \end{pmatrix}$ has inertia $(n_B, m, 0)$.* ∎

As in linear programming, the components of the vector $z = g(x) - A^T\pi$ are called the *reduced costs*. For constraints in standard form, the multipliers z_w associated inequality constraints in the working set are denoted by z_N. The components of z_N are the nonbasic components of the reduced-cost vector, i.e.,

$$z_N = (g(x) - A^T\pi)_{\mathcal{N}} = g_N - A_N^T\pi.$$

At a subspace stationary point, it holds that $g_B - A_B^T\pi = 0$, which implies that the basic components of the reduced costs z_B are zero.

The fundamental property of constraints in standard form is that the mixed-constraint method may be formulated so that the number of variables associated with the equality-constrained QP subproblem is reduced from n to n_B. By applying the permutation matrix P to the KKT equations (3.9a), we have

$$\left(\begin{array}{cc|cc} H_B & H_D & A_B^T & \\ H_D^T & H_N & A_N^T & I_N \\ \hline A_B & A_N & & \\ & I_N & & \end{array}\right)\begin{pmatrix} p_B \\ p_N \\ -q_\pi \\ -q_N \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ e_s \end{pmatrix}, \quad \text{where } p = P\begin{pmatrix} p_B \\ p_N \end{pmatrix} \text{ and } q = \begin{pmatrix} q_\pi \\ q_N \end{pmatrix}.$$

These equations imply that $p_N = e_s$ and p_B and q_π satisfy the reduced KKT system

$$\begin{pmatrix} H_B & A_B^T \\ A_B & 0 \end{pmatrix}\begin{pmatrix} p_B \\ -q_\pi \end{pmatrix} = \begin{pmatrix} -H_D p_N \\ -A_N p_N \end{pmatrix} = -\begin{pmatrix} (h_{\nu_s})_{\mathcal{B}} \\ a_{\nu_s} \end{pmatrix}. \tag{4.3}$$

In practice, p_N is defined implicitly and only the components of p_B and q_π are computed explicitly. Once p_B and q_π are known, the increment q_N for multipliers z_N associated with the constraints $p_N = e_s$ is given by $q_N = (Hp - A^T q_\pi)_{\mathcal{N}}$.

Similarly, the solution of the second KKT system (3.9b) can be computed from the KKT equation

$$\begin{pmatrix} H_B & A_B^T \\ A_B & \end{pmatrix}\begin{pmatrix} u_B \\ v_\pi \end{pmatrix} = \begin{pmatrix} e_r \\ 0 \end{pmatrix}, \tag{4.4}$$

with $u_N = 0$ and $v_N = -(Hu + A^T v_\pi)_{\mathcal{N}}$, where $u = P\begin{pmatrix} u_B \\ u_N \end{pmatrix}$ and $v = \begin{pmatrix} v_\pi \\ v_N \end{pmatrix}$.

The KKT equations (4.3) and (4.4) allow the mixed constraint algorithm to be formulated in terms of the basic variables only, which implies that the algorithm is driven by variables entering or leaving the basic set rather than constraints entering or leaving the working set. With this interpretation, changes to the KKT matrix are based on column-changes to A_B instead of row-changes to D_w.

For completeness we summarize Results 3.2–3.5 in terms of the quantities associated with constraints in standard form (an explicit proof of each result is given by Wong [57]).

Result 4.2. *Let x be a subspace minimizer with respect to the basic set $\mathcal{B}$, with $[z_N]_s < 0$. Let $\bar{x}$ be the point such that $\bar{x}_N = x_N + \alpha e_s$ and $\bar{x}_B = x_B + \alpha p_B$, where p_B is defined as in (4.3).*

(1) *The step to the minimizer of $\varphi(x + \alpha p)$ is $\alpha_* = -z_{\nu_s}/[q_N]_s$. If α_* is bounded and $\alpha = \alpha_*$, then $\bar{x}$ is a subspace minimizer with respect to the basic set $\bar{\mathcal{B}} = \mathcal{B} + \{\nu_s\}$.*

(2) *The largest feasible step is defined using the minimum ratio test:*

$$\alpha_{\scriptscriptstyle F} = \min \gamma_i, \quad \textit{where} \quad \gamma_i = \begin{cases} \dfrac{[x_{\scriptscriptstyle B}]_i}{-[p_{\scriptscriptstyle B}]_i} & \textit{if } [p_{\scriptscriptstyle B}]_i < 0, \\ +\infty & \textit{otherwise.} \end{cases}$$

Suppose $\alpha = \alpha_{\scriptscriptstyle F}$ *and* $[x_{\scriptscriptstyle B} + \alpha p_{\scriptscriptstyle B}]_{\beta_r} = 0$ *and let* $u_{\scriptscriptstyle B}$ *and* v_π *be defined by* (4.4).

(a) e_r *and the columns of* $A_{\scriptscriptstyle B}^T$ *are linearly independent if and only if* $u_{\scriptscriptstyle B} \neq 0$.

(b) $[v_{\scriptscriptstyle N}]_s = [p_{\scriptscriptstyle B}]_r < 0$ *and* $[u_{\scriptscriptstyle B}]_r \geq 0$, *with* $[u_{\scriptscriptstyle B}]_r > 0$ *if* $u_{\scriptscriptstyle B} \neq 0$.

(c) *If* e_r *and the columns of* $A_{\scriptscriptstyle B}^T$ *are linearly independent, then* $\bar{x}$ *is a subspace minimizer with respect to* $\bar{\mathcal{B}} = \mathcal{B} - \{\beta_r\}$. *Moreover,* $g_{\bar{\scriptscriptstyle B}}(\bar{x}) = A_{\scriptscriptstyle \bar{B}}^T \bar{\pi}$ *and* $g_{\bar{\scriptscriptstyle N}}(\bar{x}) = A_{\scriptscriptstyle \bar{N}}^T \bar{\pi} + \bar{z}_{\scriptscriptstyle N}$, *where* $\bar{\pi} = \pi + \alpha q_\pi$ *and* $\bar{z}_{\scriptscriptstyle N}$ *is formed by appending a zero component to the vector* $z_{\scriptscriptstyle N} + \alpha q_{\scriptscriptstyle N}$.

(d) *If* e_r *and the columns of* $A_{\scriptscriptstyle B}^T$ *are linearly dependent, define* $\sigma = [z_{\scriptscriptstyle N} + \alpha q_{\scriptscriptstyle N}]_s / [v_{\scriptscriptstyle N}]_s$. *Then* $\bar{x}$ *is a subspace minimizer with respect to* $\bar{\mathcal{B}} = \mathcal{B} - \{\beta_r\} + \{\nu_s\}$ *with* $g_{\bar{\scriptscriptstyle B}}(\bar{x}) = A_{\scriptscriptstyle \bar{B}}^T \bar{\pi}$ *and* $g_{\bar{\scriptscriptstyle N}}(\bar{x}) = A_{\scriptscriptstyle \bar{N}}^T \bar{\pi} + \bar{z}_{\scriptscriptstyle N}$, *where* $\bar{\pi} = \pi + \alpha q_\pi - \sigma v_\pi$ *with* $\sigma > 0$, *and* $\bar{z}_{\scriptscriptstyle N}$ *is formed by appending* σ *to* $z_{\scriptscriptstyle N} + \alpha q_{\scriptscriptstyle N} - \sigma v_{\scriptscriptstyle N}$. ∎

As in the generic mixed-constraint method, the direction $p_{\scriptscriptstyle B}$ and multiplier q_π may be updated in the linearly independent case.

Result 4.3. *Let* x *be a subspace minimizer with respect to* $\mathcal{B}$. *Assume the vectors* $p_{\scriptscriptstyle B}$, q_π, $u_{\scriptscriptstyle B}$ *and* v_π *are defined by* (4.3) *and* (4.4). *Let* β_r *be the index of a linearly independent blocking variable at* $\bar{x}$, *where* $\bar{x}_{\scriptscriptstyle N} = x_{\scriptscriptstyle N} + \alpha e_s$ *and* $\bar{x}_{\scriptscriptstyle B} = x_{\scriptscriptstyle B} + \alpha p_{\scriptscriptstyle B}$. *Let* $\rho = -[p_{\scriptscriptstyle B}]_r / [u_{\scriptscriptstyle B}]_r$, *and consider the vectors* $\bar{p}_{\scriptscriptstyle B}$ *and* $\bar{q}_\pi$, *where* $\bar{p}_{\scriptscriptstyle B}$ *is the vector* $p_{\scriptscriptstyle B} + \rho u_{\scriptscriptstyle B}$ *with the* r*-th component omitted, and* $\bar{q}_\pi = q_\pi - \rho v_\pi$. *Then* $\bar{p}_{\scriptscriptstyle B}$ *and* $\bar{q}_\pi$ *are well-defined and satisfy the KKT equations for the basic set* $\mathcal{B} - \{\beta_r\}$. ∎

Linear programming. If the problem is a linear program (i.e., $H = 0$), then the basic set $\mathcal{B}$ must be chosen so that $A_{\scriptscriptstyle B}$ is nonsingular (i.e., it is square with rank m). In this case, we show that Algorithm 1 simplifies to a variant of the primal simplex method in which the π-values and reduced costs are updated by a simple recurrence relation.

When $H = 0$, the equations (4.3) reduce to $A_{\scriptscriptstyle B} p_{\scriptscriptstyle B} = -a_{\nu_s}$ and $A_{\scriptscriptstyle B}^T q_\pi = 0$, with $p_{\scriptscriptstyle N} = e_s$ and $q_{\scriptscriptstyle N} = -A_{\scriptscriptstyle N}^T q_\pi$. As $A_{\scriptscriptstyle B}$ is nonsingular, both q_π and $q_{\scriptscriptstyle N}$ are zero, and the directions $p_{\scriptscriptstyle B}$ and $p_{\scriptscriptstyle N}$ are equivalent to those defined by the simplex method. For the singularity test (4.4), the basic and nonbasic components of u satisfy $A_{\scriptscriptstyle B} u_{\scriptscriptstyle B} = 0$ and $u_{\scriptscriptstyle N} = 0$. Similarly, $v_{\scriptscriptstyle N} = -A_{\scriptscriptstyle N}^T v_\pi$, where $A_{\scriptscriptstyle B}^T v_\pi = e_r$, As $A_{\scriptscriptstyle B}$ is nonsingular, $u_{\scriptscriptstyle B} = 0$ and the linearly dependent case always applies. This implies that the r-th basic and the s-th nonbasic variables are always swapped, as in the primal simplex method.

As q is zero, the updates to the multiplier vectors π and $z_{\scriptscriptstyle N}$ defined by part 2(d) of Result 4.2 depend only on the vectors v_π and $v_{\scriptscriptstyle N}$, and the scalar $\sigma = [z_{\scriptscriptstyle N}]_s / [p_{\scriptscriptstyle B}]_r$. The resulting updates to the multipliers are:

$$\pi \leftarrow \pi - \sigma v_\pi, \quad \text{and} \quad z_{\scriptscriptstyle N} \leftarrow \begin{pmatrix} z_{\scriptscriptstyle N} - \sigma v_{\scriptscriptstyle N} \\ \sigma \end{pmatrix},$$

which are the established multiplier updates associated with the simplex method (see Gill [23] and Tomlin [56]). It follows that the simplex method is a method for which every subspace minimizer is standard.

Summary and discussion. Algorithm 1 summarizes the method for general QPs in standard form. (The relation in part 2(b) of Result 4.2 is used to simplify the computation of $[v_N]_s$.) Given an arbitrary feasible point x_0, and a second-order-consistent basis $\mathcal{B}_0$, Algorithm 1 generates a sequence of primal-dual iterates $\{(x_k, y_k)\}$ and associated basic sets $\mathcal{B}_k$ such that

$$\begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \alpha_k \begin{pmatrix} p_k \\ q_k \end{pmatrix},$$

where p_k and q_k are either computed directly by solving (4.3), or are updated from previous values using the solution of (4.4).

The algorithm starts by attempting to minimize the objective with respect to the basic variables in $\mathcal{B}_0$. If the minimizer is infeasible, the quadratic objective is minimized over a sequence of nested basic sets until enough blocking variables are fixed on their bounds to define a subspace minimizer (e.g., at a vertex, which is trivially a subspace minimizer). Once the first subspace minimizer is found, the iterates occur in groups of iterates that start and finish at a standard subspace minimizer. Each group starts with the identification of a nonbasic variable x_{ν_s} with a negative reduced cost z_{ν_s}. In the group of subsequent iterations, the reduced cost z_{ν_s} is driven to zero. During each of these *intermediate* iterations, the nonbasic variable x_{ν_s} is allowed to move away from its bound, and a blocking basic variable may be made nonbasic to maintain feasibility. Once z_{ν_s} reaches zero, the associated nonbasic variable x_{ν_s} is moved into the basic set. Figure 1 depicts a sequence of intermediate iterations starting at a subspace minimizer with respect to $\mathcal{B}_0$. The figure illustrates the two ways in which the algorithm arrives at a point with a zero value of z_{ν_s} (i.e., at a subspace minimizer). In case (A), x_{j+1} is the result of an unconstrained step along p_j. In case (B), the removal of the blocking variable from the basic set would give a rank-deficient basis and the blocking index must be swapped with the nonbasic index ν_s (see part (d) of Result 4.2).

For each intermediate iteration, the definition of the optimal step α_* involves the curvature $[q_N]_s = p^T H p$, which represents the rate of change of the reduced cost z_{ν_s} in the direction p. This curvature increases monotonically over the sequence of intermediate iterates, which implies that the curvature becomes "less negative" as blocking basic variables are made nonbasic. For a convex QP, it holds that $p^T H p \geq 0$, which implies that only the first direction associated with a group of consecutive iterates can be a direction of zero curvature. Figure 2 depicts three examples of the behavior of the nonbinding multiplier $z_{\nu_s}(\sigma)$ as x varies along the piecewise linear path $x(\sigma)$ joining the sequence of intermediate iterates. The nonbinding multiplier $z_{\nu_s}(\sigma)$ is a continuous, piecewise linear function, with a discontinuous derivative at any point where a blocking variable is made nonbasic. The value of $z_{\nu_s}(0)$ is z_{ν_s}, the (negative) reduced cost at the first standard subspace minimizer. The slope of each segment is given by the value of the curvature $\theta_j = p_j^T H p_j$ along the direction of each segment of the path $x(\sigma)$. As the iterations proceed, the nonbinding multiplier is driven to zero, and the intermediate iterations terminate at the point where $z_{\nu_s}(\sigma) = 0$. As a variable moves from basic to nonbasic along the piecewise linear path, the slope of the z-segment becomes more positive. In the left-most figure, the curvature starts at a positive value, which always holds for a strictly convex problem, and is typical for a convex problem with a nonzero H. In the right-most figure, the curvature starts at zero, which is possible for a convex problem with a singular H, and is always the case for a linear program. If the problem is unbounded, then $z_{\nu_s}(\sigma)$ remains at the fixed negative value $z_{\nu_s}(0)$ for all $\sigma \geq 0$. In the lower figure, the initial curvature is negative, and p is a direction of negative curvature. This situation may occur for a nonconvex problem. In this case $z_{\nu_s}(\sigma)$ may remain negative for a number of intermediate iterations. If the problem is unbounded, then $z_{\nu_s}(\sigma)$ is unbounded below for increasing σ.

$$
\begin{array}{lccccccccc}
 & x_0 & & \longrightarrow & & \cdots & x_{j-1} & \longrightarrow & x_j & \longrightarrow & x_{j+1} \\
\textbf{(A)} & \mathcal{B}_0 & \overset{\textit{move, delete}}{\longrightarrow} & & & \cdots & \mathcal{B}_{j-1} & \overset{\textit{move, delete}}{\longrightarrow} & \mathcal{B}_j & \overset{\textit{move and add } \nu_s}{\longrightarrow} & \mathcal{B}_{j+1} \\
\textbf{(B)} & \mathcal{B}_0 & \overset{\textit{move, delete}}{\longrightarrow} & & & \cdots & \mathcal{B}_{j-1} & \overset{\textit{move, delete}}{\longrightarrow} & \mathcal{B}_j & \overset{\textit{move and swap}}{\longrightarrow} & \mathcal{B}_{j+1}
\end{array}
$$

Figure 1: This figure illustrates the structure of a typical sequence of iterations that follow the identification of a nonoptimal reduced cost. Each sequence consists of $j+2$ iterates that begin and end at the standard subspace minimizers x_0 and x_{j+1}. The j ($j \geq 0$) intermediate iterates are nonstandard subspace minimizers. In (A), x_{j+1} is reached by taking an unconstrained step along p_j. In (B), the removal of the blocking variable from the basic set would give a rank-deficient basis and the index of the blocking variable is swapped with the index of the nonbinding nonbasic variable. The point x_{j+1} is the first standard minimizer for the next sequence.

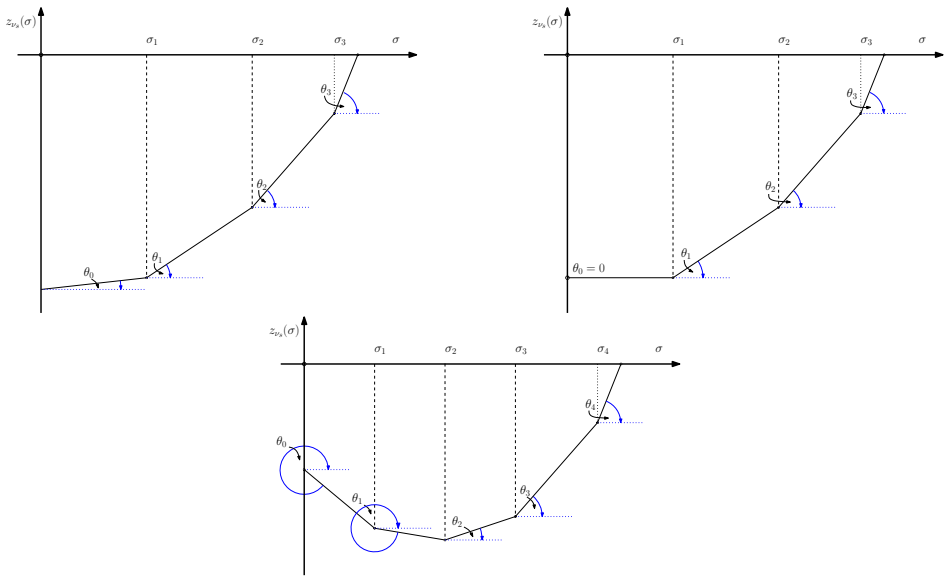

Figure 2: Three examples of the behavior of the nonbinding multiplier $z_{\nu_s}(\sigma)$ as x varies along the piecewise linear path $x(\sigma)$ joining the sequence of intermediate iterates. The function $z_{\nu_s}(\sigma)$ is piecewise linear with $z_{\nu_s}(0) < 0$, and slopes $\theta_j = p_j^T H p_j$ that increase monotonically as blocking variables are made nonbasic. As the iterations proceed, the nonbinding multiplier is driven to zero, and the intermediate iterations terminate at the point where $z_{\nu_s}(\sigma) = 0$. The left-most figure depicts a convex problem for which the curvature starts at a positive value. The right-most figure depicts a convex problem for which the curvature starts at zero. The lower figure depicts a nonconvex problem for which the curvature starts at a negative value.

Algorithm 1 Method for a general QP in standard form.

Find x_0 such that $Ax_0 = b$ and $x_0 \geq 0$;
$[x, \pi, \mathcal{B}, \mathcal{N}] = \texttt{subspaceMin}(x_0)$; [find a subspace minimizer]
$g = c + Hx$; $z = g - A^T\pi$;
$\nu_s = \text{argmin}_i\{z_i\}$; [identify the least-optimal multiplier]
while $z_{\nu_s} < 0$ **do** [drive z_{ν_s} to zero]
  Solve $\begin{pmatrix} H_B & A_B^T \\ A_B & \end{pmatrix} \begin{pmatrix} p_B \\ -q_\pi \end{pmatrix} = -\begin{pmatrix} (h_{\nu_s})_B \\ a_{\nu_s} \end{pmatrix}$; $p_N = e_s$;
  repeat
    $p = P\begin{pmatrix} p_B \\ p_N \end{pmatrix}$; $q_N = (Hp - A^Tq_\pi)_{\mathcal{N}}$;
    $\alpha_F = \texttt{minRatioTest}(x_B, p_B)$; [compute the largest step to a blocking variable]
    if $[q_N]_s > 0$ **then**
      $\alpha_* = -z_{\nu_s}/[q_N]_s$;
    else
      $\alpha_* = +\infty$; [compute the optimal step]
    end if
    $\alpha = \min\{\alpha_*, \alpha_F\}$;
    if $\alpha = +\infty$ **then**
      stop; [unbounded solution]
    end if
    $x \leftarrow x + \alpha p$; $g \leftarrow g + \alpha Hp$;
    $\pi \leftarrow \pi + \alpha q_\pi$; $z = g - A^T\pi$;
    if $\alpha_F < \alpha_*$ **then**
      Find the index r of a blocking variable;
      Solve $\begin{pmatrix} H_B & A_B^T \\ A_B & \end{pmatrix} \begin{pmatrix} u_B \\ v_\pi \end{pmatrix} = \begin{pmatrix} e_r \\ 0 \end{pmatrix}$;
      if $u_B = 0$ **then**
        $\sigma = z_{\nu_s}/[p_B]_r$; $\pi \leftarrow \pi - \sigma v_\pi$;
        $z = g - A^T\pi$; [implies $z_{\nu_s} = 0$]
      else
        $\rho = -[p_B]_r/[u_B]_r$;
        $p_B \leftarrow p_B + \rho u_B$; $q_\pi \leftarrow q_\pi - \rho v_\pi$;
      end if
      $\mathcal{B} \leftarrow \mathcal{B} - \{\beta_r\}$; $\mathcal{N} \leftarrow \mathcal{N} + \{\beta_r\}$; [make the blocking variable β_r nonbasic]
    end if
  until $z_{\nu_s} = 0$;
  $\mathcal{B} \leftarrow \mathcal{B} + \{\nu_s\}$; $\mathcal{N} \leftarrow \mathcal{N} - \{\nu_s\}$; [make variable ν_s basic]
  $\nu_s = \text{argmin}_i\{z_i\}$;
  $k \leftarrow k + 1$;
end while

5. Solving the KKT Systems

At each iteration of the primal methods discussed in Sections 4, it is necessary to solve one or two systems of the form

$$\begin{pmatrix} H_B & A_B^T \\ A_B & \end{pmatrix} \begin{pmatrix} y \\ w \end{pmatrix} = \begin{pmatrix} h \\ f \end{pmatrix}, \tag{5.1}$$

where h and f are given by right-hand sides of the equations (4.3) or (4.4). Two alternative approaches for solving (5.1) are described. The first involves the symmetric transformation of the KKT system into three smaller systems, one of which involves the explicit reduced Hessian matrix. The second approach uses a symmetric indefinite factorization of a fixed KKT matrix in conjunction with the factorization of a smaller matrix that is updated at each iteration.

5.1. Variable reduction

The variable-reduction method involves transforming the equations (5.1) to block-triangular form using the nonsingular block-diagonal matrix $\operatorname{diag}(Q, I_m)$. Consider a column permutation P such that

$$AP = \begin{pmatrix} B & S & N \end{pmatrix}, \tag{5.2}$$

with B an $m \times m$ nonsingular matrix and S an $m \times n_S$ matrix with $n_S = n_B - m$. The matrix P is a version of the permutation $P = \begin{pmatrix} P_B & P_N \end{pmatrix}$ of (4.2) that also arranges the columns of A_B in the form $A_B = \begin{pmatrix} B & S \end{pmatrix}$. The n_S variables associated with S are called the *superbasic* variables. Given P, consider the nonsingular $n \times n$ matrix Q such that

$$Q = P \begin{pmatrix} -B^{-1}S & I_m & 0 \\ I_{n_S} & 0 & 0 \\ 0 & 0 & I_N \end{pmatrix}.$$

The columns of Q may be partitioned so that $Q = \begin{pmatrix} Z & Y & W \end{pmatrix}$, where

$$Z = P \begin{pmatrix} -B^{-1}S \\ I_{n_S} \\ 0 \end{pmatrix}, \quad Y = P \begin{pmatrix} I_m \\ 0 \\ 0 \end{pmatrix} \quad \text{and} \quad W = P \begin{pmatrix} 0 \\ 0 \\ I_N \end{pmatrix}.$$

The columns of the $n \times n_S$ matrix Z form a basis for the null space of A_w, with

$$A_w Q = \begin{pmatrix} A \\ P_N^T \end{pmatrix} Q = \begin{pmatrix} 0 & B & N \\ 0 & 0 & I_N \end{pmatrix}.$$

Suppose that we wish to solve a generic KKT system

$$\begin{pmatrix} H & A^T & P_N \\ A & & \\ P_N^T & & \end{pmatrix} \begin{pmatrix} y \\ w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} h \\ f_1 \\ f_2 \end{pmatrix}.$$

Then the vector y may be computed as $y = Y y_Y + Z y_Z + W y_W$, where y_Y, y_Z, y_W and w are defined using the equations

$$\begin{pmatrix} Z^THZ & Z^THY & Z^THW & & \\ Y^THZ & Y^THY & Y^THW & B^T & \\ W^THZ & W^THY & W^THW & N^T & I_N \\ & & B & N & \\ & & & I_N & \end{pmatrix} \begin{pmatrix} y_Z \\ y_Y \\ y_W \\ w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} h_Z \\ h_Y \\ h_W \\ f_1 \\ f_2 \end{pmatrix}, \tag{5.3}$$

with $h_Z = Z^T h$, $h_Y = Y^T h$, and $h_W = W^T h$. This leads to

$$\begin{aligned} y_W &= f_2, \\ By_Y &= f_1 - Nf_2, \qquad & y_R &= Yy_Y + Wy_W, \\ Z^T H Z y_Z &= Z^T(h - Hy_R), & y_T &= Zy_Z, & y &= y_R + y_T, \\ B^T w_1 &= Y^T(h - Hy), & w_2 &= W^T(h - Hy) - N^T w_1. \end{aligned}$$

The equations simplify considerably for the KKT systems (3.9a) and (3.9b). In the case of (3.9a), the equations are:

$$\begin{aligned} Bp_Y &= -a_{\nu_s}, \qquad & p_R &= P \begin{pmatrix} p_Y \\ 0 \\ e_s \end{pmatrix}, \\ Z^T H Z p_Z &= -Z^T H p_R, & p_T &= Zp_Z, & p &= p_R + p_T, \\ B^T q_\pi &= (Hp)_{\mathcal{B}}, & q_z &= (Hp - A^T q_\pi)_{\mathcal{N}}. \end{aligned} \tag{5.4}$$

Similarly for (3.9b), it holds that $u_Y = 0$, $u_R = 0$, and

$$\begin{aligned} Z^T H Z u_Z &= Z^T e_{\beta_r}, & u &= Zu_Z, \\ B^T v_\pi &= (e_{\beta_r} - Hu)_{\mathcal{B}}, & v_z &= -(Hu + A^T v_\pi)_{\mathcal{N}}. \end{aligned} \tag{5.5}$$

These equations allow us to specialize Part 2(a) of Result 4.2, which gives the conditions for the linear independence of the rows of the new A_B.

Result 5.1. *Let x be a subspace minimizer with respect to the basic set $\mathcal{B}$. Assume that p and q are defined by (4.3), and that x_{β_r} is the variable selected to be nonbasic at the next iterate. Let the vectors u_B and v_π be defined by (4.4).*

(a) *If x_{β_r} is superbasic, then e_r and the rows of A_B are linearly independent (i.e., the matrix obtained by removing the rth column of A_B has rank m).*

(b) *If x_{β_r} is not superbasic, then e_r is linearly independent of the rows of A_B if and only if $S^T z \neq 0$, where z is the solution of $B^T z = e_r$.*

Proof. From (5.5), $u = Zu_Z$, which implies that u_B is nonzero if and only if u_Z is nonzero. Similarly, the nonsingularity of $Z^T H Z$ implies that u_Z is nonzero if and only if $Z^T e_{\beta_r}$ is nonzero. Now

$$Z^T e_{\beta_r} = \begin{pmatrix} -S^T B^{-T} & I_{n_S} & 0 \end{pmatrix} e_r.$$

If x_{β_r} is superbasic, then $r > m$ and $Z^T e_{\beta_r} = e_{r-m} \neq 0$ and u_Z is nonzero. If x_{β_r} is not superbasic, then $r \leq m$, and

$$Z^T e_{\beta_r} = -S^T B^{-T} e_r = -S^T z,$$

where z is the solution of $B^T z = e_r$. ∎

The equations (5.4) and (5.5) may be solved using a Cholesky factorization of $Z^T H Z$ and an LU factorization of B. The factors of B allow efficient calculation of matrix-vector products $Z^T v$ or Zv without the need to form the inverse of B.

5.2. Fixed-factorization updates

When A_B and H_B are large and sparse, there are many reliable and efficient sparse-matrix factorization packages for solving a symmetric indefinite system of the form (5.1). Some prominent software packages include `MA27` (Duff and Reid [16]), `HSL_MA57` (Duff [15]), `HSL_MA97` (Hogg and Scott [43]), `MUMPS` (Amestoy et al. [1]), `PARDISO` (Schenk and Gärtner [55]), and `SPOOLES` (Ashcraft and Grimes [2]). However, in a QP algorithm, a sequence of related systems must be solved in which the KKT matrix changes by a single row and column. In this situation, instead of factoring the matrix in (5.1) directly, the first K_0 may be "bordered" in a way that reflects the changes to the basic and nonbasic sets during a set of k subsequent iterations. The solution of (5.1) is then found by using a *fixed* factorization of K_0, and a factorization of a smaller matrix of (at most) order k (see Bisschop and Meeraus [5], and Gill et al. [31]). Although K_0 is symmetric, the matrix may be factored by any symmetric or unsymmetric linear solver, allowing a variety of black-box linear solvers to be incorporated into the algorithm.

Let $\mathcal{B}_0$ and $\mathcal{N}_0$ denote the initial basic and nonbasic sets that define the KKT system (5.1). There are four cases to consider:

(1) a nonbasic variable moves to the basic set and is not in $\mathcal{B}_0$,

(2) a basic variable in $\mathcal{B}_0$ becomes nonbasic,

(3) a basic variable not in $\mathcal{B}_0$ becomes nonbasic, and

(4) a nonbasic variable moves to the basic set and is in $\mathcal{B}_0$.

For case (1), let ν_s be the nonbasic variable that has become basic. The next KKT matrix can be written as

$$\left(\begin{array}{cc|c} H_B & A_B^T & (h_{\nu_s})_{\mathcal{B}_0} \\ A_B & 0 & a_{\nu_s} \\ \hline (h_{\nu_s})_{\mathcal{B}_0}^T & a_{\nu_s}^T & h_{\nu_s,\nu_s} \end{array}\right).$$

Suppose that at the next stage, another nonbasic variable ν_r becomes basic. The KKT matrix is augmented in a similar fashion, i.e.,

$$\left(\begin{array}{ccc|c} H_B & A_B^T & (h_{\nu_s})_{\mathcal{B}_0} & (h_{\nu_r})_{\mathcal{B}_0} \\ A_B & 0 & a_{\nu_s} & a_{\nu_r} \\ (h_{\nu_s})_{\mathcal{B}_0}^T & a_{\nu_s}^T & h_{\nu_s,\nu_s} & h_{\nu_s,\nu_r} \\ \hline (h_{\nu_r})_{\mathcal{B}_0}^T & a_{\nu_r}^T & h_{\nu_r,\nu_s} & h_{\nu_r,\nu_r} \end{array}\right).$$

Now consider case 2 and let $\beta_r \in \mathcal{B}_0$ become nonbasic. The change to the basic set is reflected in the new KKT matrix

$$\left(\begin{array}{cccc|c} H_B & A_B^T & (h_{\nu_s})_{\mathcal{B}_0} & (h_{\nu_r})_{\mathcal{B}_0} & e_r \\ A_B & 0 & a_{\nu_s} & a_{\nu_r} & 0 \\ (h_{\nu_s})_{\mathcal{B}_0}^T & a_{\nu_s}^T & h_{\nu_s,\nu_s} & h_{\nu_s,\nu_r} & 0 \\ (h_{\nu_r})_{\mathcal{B}_0}^T & a_{\nu_r}^T & h_{\nu_r,\nu_s} & h_{\nu_r,\nu_r} & 0 \\ \hline e_r^T & 0 & 0 & 0 & 0 \end{array}\right).$$

The unit row and column augmenting the matrix has the effect of zeroing out the components corresponding to the removed basic variable.

In case (3), the basic variable must have been added to the basic set at a previous stage as in case (1). Thus, removing it from the basic set can be done by removing the row and

column in the augmented part of the KKT matrix corresponding to its addition to the basic set. For example, if ν_s is the basic to be removed, then the new KKT matrix is given by

$$\begin{pmatrix} H_B & A_B^T & (h_{\nu_r})_{\mathcal{B}_0} & e_r \\ A_B & 0 & a_{\nu_r} & 0 \\ (h_{\nu_r})_{\mathcal{B}_0}^T & a_{\nu_r}^T & h_{\nu_r,\nu_r} & 0 \\ e_r^T & 0 & 0 & 0 \end{pmatrix}.$$

For case (4), a nonbasic variable in $\mathcal{B}_0$ implies that at some previous stage, the variable was removed from $\mathcal{B}_0$ as in case (2). The new KKT matrix can be formed by removing the unit row and column in the augmented part of the KKT matrix corresponding to the removal the variable from the basic set. In this example, the new KKT matrix becomes

$$\begin{pmatrix} H_B & A_B^T & (h_{\nu_r})_{\mathcal{B}_0} \\ A_B & 0 & a_{\nu_r} \\ (h_{\nu_r})_{\mathcal{B}_0}^T & a_{\nu_r}^T & h_{\nu_r,\nu_r} \end{pmatrix}.$$

After k iterations, the KKT system is maintained as a symmetric augmented system of the form

$$\begin{pmatrix} K & V \\ V^T & D \end{pmatrix} \begin{pmatrix} r \\ \eta \end{pmatrix} = \begin{pmatrix} b \\ f \end{pmatrix} \quad \text{with} \quad K = \begin{pmatrix} H_B & A_B^T \\ A_B & \end{pmatrix}, \tag{5.6}$$

where D is of dimension at most $2k$.

5.2.1. Schur complement and block LU methods

Although the augmented system (in general) increases in dimension by one at every iteration, the first diagonal block K of (5.6) is fixed and defined by the initial set of basic variables. The *Schur complement method* assumes that factorizations for K and the *Schur complement* $C = D - V^T K^{-1} V$ exist. Then the solution of (5.6) can be determined by solving the equations

$$Kt = b, \quad C\eta = f - V^T t, \quad Kr = b - V\eta.$$

The work required is dominated by two solves with the fixed matrix K and one solve with the Schur complement C. If the number of changes to the basic set is small enough, dense factors of C may be maintained.

The Schur complement method can be extended to a *block LU method* by storing the augmented matrix in block factors

$$\begin{pmatrix} K & V \\ V^T & D \end{pmatrix} = \begin{pmatrix} L & \\ Z^T & I \end{pmatrix} \begin{pmatrix} U & Y \\ & C \end{pmatrix}, \tag{5.7}$$

where $K = LU$, $LY = V$, $U^T Z = V$, and $C = D - Z^T Y$ is the Schur-complement matrix.

The solution of (5.6) can be computed by forming the block factors and by solving the equations

$$Lt = b, \quad C\eta = f - Z^T t, \quad Ur = t - Y\eta.$$

This method requires a solve with L and U each, one multiply with Y and Z^T, and one solve with the Schur complement C. For more details, see Gill et al. [28], Eldersveld and Saunders [17], and Huynh [45].

As the iterations of the QP algorithm proceed, the size of C increases and the work required to solve with C increases. It may be necessary to restart the process by discarding the existing factors and re-forming K based on the current set of basic variables.

5.2.2. Updating the block LU factors

Suppose the current KKT matrix is bordered by the vectors v and w, and the scalar σ

$$\left(\begin{array}{cc|c} K & V & v \\ V^T & D & w \\ \hline v^T & w^T & \sigma \end{array}\right).$$

The block LU factors Y and Z, and the Schur complement C are updated every time the system is bordered. The number of columns in matrices Y and Z and the dimension of the Schur complement increase by one. The updates y, z, c and d are defined by the equations

$$Ly = v, \qquad U^T z = v,$$
$$c = w - Z^T y = w - Y^T z, \qquad d = \sigma - z^T y,$$

so that the new block LU factors satisfy

$$\left(\begin{array}{c|cc} K & V & v \\ \hline V^T & D & w \\ v^T & w^T & \sigma \end{array}\right) = \left(\begin{array}{c|cc} L & & \\ \hline Z^T & I & \\ z^T & & 1 \end{array}\right) \left(\begin{array}{c|cc} U & Y & y \\ \hline & C & c \\ & c^T & d \end{array}\right).$$

6. Finding a Subspace Minimizer

The method described in Section 4 has the property that if the initial iterate x_0 is a subspace minimizer, then all subsequent iterates are subspace minimizers (see Result 4.2). Methods for finding an initial subspace minimizer utilize an initial estimate x_I of the solution together with matrices A_B and A_N associated with an estimate of the optimal basic and nonbasic partitions of A. These estimates are often available from the known solution of a related QP—e.g., from the solution of the previous QP subproblem in the SQP context. The initial point x_I may or may not be feasible, and the associated matrix A_B may or may not have rank m.

The definition of a second-order-consistent basis requires that the matrix A_B has rank m, and it is necessary to identify a set of linearly independent basic columns of A. One algorithm for doing this has been proposed by Gill, Murray and Saunders [26], who use a sparse LU factorization of A_B^T to identify a square nonsingular subset of the columns of A_B. If necessary, a "basis repair" scheme is used to define additional unit columns that make A_B have full rank. The nonsingular matrix B obtained as a by-product of this process may be expressed in terms of A using a column permutation P such that

$$AP = \begin{pmatrix} A_B & A_N \end{pmatrix} = \begin{pmatrix} B & S & A_N \end{pmatrix}. \tag{6.1}$$

Given x_I, a point x_0 satisfying $Ax = b$ may be computed as

$$x_0 = x_I + P \begin{pmatrix} p_Y \\ 0 \\ 0 \end{pmatrix}, \quad \text{where} \quad Bp_Y = -(Ax_I - b).$$

If the matrix

$$K_B = \begin{pmatrix} H_B & A_B^T \\ A_B & \end{pmatrix} \tag{6.2}$$

has n_B positive eigenvalues and m negative eigenvalues, then the inertia of K_B is correct and x_0 is used as the initial point for a sequence of Newton-type iterations in which $\varphi(x)$

is minimized with the nonbasic components of x fixed at their current values. Consider the equations

$$\begin{pmatrix} H_B & A_B^T \\ A_B & \end{pmatrix} \begin{pmatrix} p_B \\ -\pi \end{pmatrix} = - \begin{pmatrix} g_B \\ 0 \end{pmatrix}.$$

If p_B is zero, x is a subspace stationary point (with respect to A_B) at which K_B has correct inertia and we are done. If p_B is nonzero, two situations are possible.

If $x_B + p_B$ is infeasible, then feasibility is retained by determining the maximum nonnegative step $\alpha < 1$ such that $x_B + \alpha p_B$ is feasible. A variable on its bound at $x_B + \alpha p_B$ is then removed from the basic set and the iteration is repeated. The removal of a basic variable cannot increase the number of negative eigenvalues of K_B and a subspace minimizer must be determined in a finite number of steps.

If $x_B + p_B$ is feasible, then p_B is the step to the minimizer of $\varphi(x)$ with respect to the basic variables and it must hold that $x_B + p_B$ is a subspace minimizer.

A KKT matrix with incorrect inertia has too many negative or zero eigenvalues. In this case, an appropriate K_B may be obtained by imposing temporary constraints that are deleted during the course of subsequent iterations. For example, if $n - m$ variables are temporarily fixed at their current values, then A_B is a square nonsingular matrix and K_B necessarily has exactly m negative eigenvalues. The form of the temporary constraints depends on the method used to solve the reduced KKT equations (5.1).

6.1. Variable-reduction method

In the variable-reduction method a dense Cholesky factor of the reduced Hessian $Z^T H Z$ is updated to reflect changes in the basic set (see Section 5.1). At the initial point x_0, a partial Cholesky factorization with interchanges is used to find an upper-triangular matrix R that is the factor of the largest positive-definite leading submatrix of $Z^T H Z$. The use of interchanges tends to maximize the dimension of R. Let Z_R denote the columns of Z corresponding to R, and let Z be partitioned as $Z = \begin{pmatrix} Z_R & Z_A \end{pmatrix}$. A nonbasic set for which Z_R defines an appropriate null space can be obtained by fixing the variables corresponding to the columns of Z_A at their current values. As described above, minimization of $\varphi(x)$ then proceeds within the subspace defined by Z_R. If a variable is removed from the basic set, a row and column is removed from the reduced Hessian and an appropriate update is made to the Cholesky factor.

6.2. Fixed-factorization updates

If fixed-factorization updates to the KKT matrix are being used, the procedure for finding a second-order-consistent basis is given as follows.

1. The reduced KKT matrix (6.2) is factored as $K_B = L D L^T$, where L is unit lower-triangular and D is block diagonal with 1×1 and 2×2 blocks. If the inertia of K_B is correct, then we are done.

2. If the inertia of K_B is incorrect, the symmetric indefinite factorization

$$H_A = H_B + \rho A_B^T A_B = L_A D_A L_A^T$$

is computed for some modest positive penalty parameter ρ. As the inertia of K_B is not correct, D_A will have some negative eigenvalues for all positive ρ.

The factorization of H_A may be written in the form

$$H_A = L_A U \Lambda U^T L_A^T = V \Lambda V^T,$$

where $U\Lambda U^T$ is the spectral decomposition of D_A. The block diagonal structure of D_A implies that U is a block-diagonal orthonormal matrix. The inertia of Λ is the same as the inertia of H_A, and there exists a positive semidefinite diagonal matrix E such that $\Lambda+E$ is positive definite. If $\bar{H}_A$ is the positive-definite matrix $V(\Lambda+E)V^T$, then

$$\bar{H}_A = H_A + VEV^T = H_A + \sum_{e_{jj}>0} e_{jj} v_j v_j^T.$$

If H_A has r nonpositive eigenvalues, let V_B denote the $r \times n_B$ matrix consisting of the columns of V associated with the positive components of E. The augmented KKT matrix

$$\begin{pmatrix} H_B & A_B^T & V_B \\ A_B & 0 & 0 \\ V_B^T & 0 & 0 \end{pmatrix}$$

has exactly $m + r$ negative eigenvalues and hence has correct inertia.

The minimization of $\varphi(x)$ proceeds subject to the original constraints and the (general) temporary constraints $V_B^T x_B = 0$.

The efficiency of this scheme will depend on the number of surplus negative and zero eigenvalues in H_A. In practice, if the number of negative eigenvalues exceeds a preassigned threshold, then a temporary vertex is defined by fixing the variables associated with the columns of S in (6.1) (see the discussion of Section 7.1).

7. Numerical Results

7.1. Implementation

The package `SQIC` is a Fortran 2008 implementation of the general quadratic programming method discussed in Section 4. `SQIC` is designed to solve large-scale problems of the form

$$\underset{x}{\text{minimize}} \;\; \varphi(x) = c^T x + \tfrac{1}{2} x^T H x \quad \text{subject to} \quad l \le \begin{pmatrix} x \\ Ax \end{pmatrix} \le u,$$

where l and u are constant lower and upper bounds, c is the constant linear term of the objective, and A and H are sparse matrices of dimension $m \times n$ and $n \times n$ respectively. Internally, `SQIC` transforms this problem into standard form by introducing a vector of slack variables s. The equivalent problem is

$$\underset{x,s}{\text{minimize}} \;\; \varphi(x) = c^T x + \tfrac{1}{2} x^T H x \quad \text{subject to} \quad Ax - s = 0, \quad l \le \begin{pmatrix} x \\ s \end{pmatrix} \le u. \tag{7.1}$$

By default, a scaling of H and A is defined based on the scaling algorithm in [53] applied to the symmetric KKT matrix defined with H and A. The built-in scaling routines used by the linear solvers are turned off.

At any given iteration, `SQIC` operates in either *variable-reduction mode* or *block-matrix mode*. The mode determines which method is used to solve the KKT system. The starting mode depends on the available solvers and on the number of superbasics at the initial QP point. If the initial number of superbasics is greater than 2000, then `SQIC` starts in block-matrix mode; otherwise, it starts in variable-reduction mode. In subsequent iterations, `SQIC` will switch between variable-reduction mode and block-matrix mode as the number of superbasic variables changes. The user may override the default settings and specify that `SQIC` starts in a specific mode or uses one of the modes exclusively.

An initial feasible point and basis are found by using the phase 1 algorithm of `SQOPT` [27], which uses the simplex method to minimize the sum of the infeasibilities of the bound

constraints subject to $Ax = b$. The resulting basis defines a vertex with n_S variables temporarily fixed between their bounds. As SQIC does not require a vertex to start, these variables are freed simultaneously to create a basic set of size $m + n_S$. If the KKT matrix associated with this basic set has incorrect inertia, then the number of negative eigenvalues is greater than m and the estimated number of temporary constraints e_a is defined as the difference of these numbers. If e_a is greater than $\max\left(10, \frac{1}{2}(n_B - m)\right)$, then the n_S variables are removed from the basic set and the initial m-basis provided by SQOPT is used to define a vertex. Otherwise, the method described in Section 6.2 is used to define temporary constraints that define a second-order-consistent basis.

Table 1: SQIC tolerances and their default settings. ϵ_M is the machine precision.

Tolerance	Default Setting
Linear independence test $\epsilon_{\texttt{dep}}$	5×10^{-9}
Feasibility $\epsilon_{\texttt{fea}}$	10^{-6}
Optimality $\epsilon_{\texttt{opt}}$	10^{-6}
Iterative refinement $\epsilon_{\texttt{res}}$	$\epsilon_M^{0.8}$
Upper bound on Schur-complement condition number	10^{16}

Three linear solvers have been incorporated into SQIC to store the block-LU (or block-LDL$^{\mathrm{T}}$) factors of the KKT matrix. These are the symmetric LDL$^{\mathrm{T}}$ solver HSL_MA57 [44], and the unsymmetric LU solvers LUSOL [29] and UMFPACK [10, 11, 12, 13]. In the discussion below of the numerical results, SQIC-LUSOL, SQIC-UMFPACK and SQIC-MA57 refer to the versions of SQIC with block-matrix solver options LUSOL, UMFPACK and HSL_MA57, respectively. In variable-reduction mode, all of these versions use the LUSOL package to maintain the LU factors of the square basis matrix B (see equation (5.2)).

In block-matrix mode, the Schur complement matrix is maintained by the dense matrix factorization package LUMOD [54]. LUMOD was updated to Fortran 90 by Huynh [45] for the convex quadratic programming code QPBLU, which also utilizes a block-LU scheme. Modifications were made to the Fortran 90 version of LUMOD to incorporate it into SQIC.

The algorithm described in Section 6.2 for computing temporary constraints for a second-order-consistent basis requires a linear solver that computes an LDL$^{\mathrm{T}}$ factorization and provides access to the matrix L. Of the three solvers that were tested, only HSL_MA57 is a symmetric indefinite solver and allows access to the L matrix. For all other solvers, a temporary vertex is defined at the initial feasible point if the initial basis is not second-order consistent.

Table 1 lists the values of various tolerances used to obtain the numerical results. For example, the test for linear dependence in (4.4) is $[u_B]_r \leq \epsilon_{\texttt{dep}}\, [p_B]_r$, where $\epsilon_{\texttt{dep}}$ is a tolerance with default value $\epsilon_{\texttt{dep}} = 5 \times 10^{-9}$.

There are two situations in which the Schur complement is discarded and the KKT matrix is refactorized. The first is for structural reasons when the dimension of the Schur complement exceeds $\min\left(1000, \frac{1}{2}(n_B + m)\right)$. The second is for numerical reasons when the estimated condition number condC of the Schur complement is greater than 10^{16}, in which case the new factors are used to define a step of iterative refinement for x and π. If no refactorization is needed, but condC or condK (the estimated condition number of the matrix K of (5.7)) is greater than 10^9, then the residuals of the equations that define x and π are computed. If the norm of the residual is greater than $\epsilon_{\texttt{res}} \max(\texttt{condK}, \texttt{condC})$, then one step of iterative refinement is applied to x and π. The default value of the refinement tolerance $\epsilon_{\texttt{res}}$ is $\epsilon_M^{0.8}$, where ϵ_M is the machine precision. The estimate condK is provided by

the block solver. If no such estimate is available, then the test for refinement is based solely on `condC`.

Both `SQIC` and `SQOPT` use the EXPAND procedure of Gill et al. [30] to allow the variables (x, s) to stray outside their bounds by as much as a user-specified feasibility tolerance $\epsilon_{\texttt{fea}}$ with default value 10^{-6}. The EXPAND procedure allows some choice of constraint to be added to the working set and reduces the chance of cycling at a point where the working-set constraints are nearly linearly dependent. EXPAND first computes a maximum feasible step α_P for an expanded feasible defined by perturbing each constraint bound by the working feasibility tolerance. All constraints at a distance α ($\alpha \leq \alpha_P$) along p from the current point are then viewed as acceptable candidates for inclusion in the working set. The constraint whose normal makes the biggest angle with the search direction is added to the working set. This strategy helps keep the basis matrix A_B well conditioned. Over a period of $K = 10^3$ iterations, a "working" feasibility tolerance increases from $\frac{1}{2}\epsilon_{\texttt{fea}}$ to $\epsilon_{\texttt{fea}}$ in steps of $\frac{1}{2}\epsilon_{\texttt{fea}}/K$. At certain stages, the following "resetting procedure" is used to remove small constraint infeasibilities. First, all nonbasic variables are moved exactly onto their bounds. A count is kept of the number of non-trivial adjustments made. If the count is nonzero, the basic variables are recomputed. Finally, the working feasibility tolerance is reinitialized to $\frac{1}{2}\epsilon_{\texttt{fea}}$. If a problem requires more than K iterations, the resetting procedure is invoked and a new cycle of iterations is started. (The decision to resume phase 1 or phase 2 is based on comparing any infeasibilities with $\epsilon_{\texttt{fea}}$.) The resetting procedure is also invoked when the solver reaches an apparently optimal, infeasible, or unbounded solution, unless this situation has already occurred twice. If any non-trivial adjustments are made, iterations are continued. Although the EXPAND procedure provides no guarantee that cycling will not occur, the probability is very small (see Hall and McKinnon [42]).

By default, `SQIC` is terminated at a point (x, π, z) that approximately satisfies three conditions: (i) the reduced KKT matrix has correct inertia; (ii) the reduced gradient is zero, and (iii) the reduced costs are nonnegative. The definition of a "zero" reduced gradient and "nonnegative" reduced cost is determined by the positive tolerance $\epsilon_{\texttt{opt}}$, which has default value 10^{-6}. For a given $\epsilon_{\texttt{opt}}$, `SQIC` will terminate when

$$\max_{i\in\mathcal{B}} |z_i| \leq \epsilon_{\texttt{opt}}\|\pi\|_\infty, \quad \text{and} \quad \begin{cases} z_i \geq -\epsilon_{\texttt{opt}}\|\pi\|_\infty & \text{if } x_i \geq -\ell_i,\ i \in \mathcal{N}; \\ z_i \leq \ \ \epsilon_{\texttt{opt}}\|\pi\|_\infty & \text{if } x_i \leq \ \ u_i,\ i \in \mathcal{N}. \end{cases} \tag{7.2}$$

If the QP is convex, then (x, π, z) approximates a point at which the objective has a global minimum. In addition, if all the nonbasic reduced costs are sufficiently large, then (x, π, z) approximates the unique global minimizer. Otherwise (x, π, z) is a weak (i.e., non-unique) global minimizer. For a convex QP, a point (x, π, z) satisfying (7.2) is judged to be a weak global minimizer if there is at least one nonbasic reduced cost that satisfies $|z_j| < \epsilon_{\texttt{opt}}\|\pi\|_\infty$.

If the QP is not convex, then the situation is more complicated. If all the nonbasic reduced costs are sufficiently large then (x, π, z) is an approximate local minimizer. If some nonbasic reduced costs are approximately zero, then (x, π, z) is an approximate "dead-point", i.e., a point at which the first and second-order necessary conditions for optimality hold, but the second-order sufficient conditions do not hold. A dead-point may or may not be optimal. Moreover, the verification of optimality requires finding the global minimizer of an indefinite quadratic form over a cone, which is an NP-hard problem (see the discussion following Result 2.1 of Section 2.1).

In order to declare the QP optimal or compute a feasible descent direction at a dead-point, it may be necessary to remove a variable from the nonbasic set when the reduced KKT matrix is singular (in which case K_B does not have correct inertia). For example,

consider a problem written in the form (7.1) with

$$\varphi(x) = -x_1 x_2, \quad A = \begin{pmatrix} 1 & 1 \end{pmatrix}, \quad 0 \leq x_1, x_2 \leq +\infty, \text{ and } -\infty \leq s_1 \leq +\infty.$$

In this case, if x_1 and x_2 are nonbasic with $x_1 = x_2 = s_1 = 0$, then it is necessary to make both x_1 and x_2 basic in order to determine a feasible descent direction. An analogous situation applies in the general case, where it can be shown that a feasible descent direction may be computed at a non-optimal dead-point by simultaneously removing only *two* variables from the nonbasic set (see Contesse [8]). A procedure for computing such a direction as part of an inertia-controlling method is given by Forsgren, Gill and Murray [21]. However, the underlying computational intractability of verifying the sufficient conditions implies that there is no reasonable bound on the number of iterations that might be needed to identify a feasible descent direction (if such a direction exists). For this reason, the default strategy for `SQIC` is to terminate at a point satisfying the conditions (7.2). If there is at least one nonbasic reduced cost such that z_j such that $|z_j| \leq \epsilon_{\texttt{opt}} \|\pi\|_\infty$, then (x, π, z) is declared to be either a likely weak minimizer or a dead point, depending on whether or not negative curvature was encountered during any previous iteration. (It is not possible to guarantee that a problem is convex without the additional cost of a symmetric indefinite factorization of the full Hessian.)

The `SQIC` package includes an option to request that nonbasic variables with "zero" reduced costs be moved sequentially to the basic set if the iterations terminate at a point where the second-order sufficient conditions are not satisfied (i.e., at a weak global minimizer or a dead-point). This "phase 3" procedure continues at the point of termination until one of the following situations applies: (i) no small reduced costs remain and K_B has correct inertia; (ii) K_B becomes singular; or (iii) a feasible direction of negative curvature is identified. In the first situation, all the constraints with zero reduced costs are weakly active. In the case of (ii) the inertia-controlling strategy prohibits the removal of additional zero reduced costs, and phase 3 is terminated. In the case of (iii), `SQIC` was terminated at a nonoptimal dead-point, which implies that phase 3 can be terminated and phase 2 restarted.

At each step of phase 3, a zero reduced cost z_{ν_s} is identified, and a direction p is computed using System 1. If $p^T H p > |z_{\nu_s}|$ then the curvature is considered to be sufficiently positive. In this case, x_{ν_s} is added to the basic set, and another zero reduced cost is selected without moving from the current point. If $p^T H p \leq |z_{\nu_s}|$, then the curvature is judged to be zero and the algorithm is terminated, as dictated by the circumstances of case (ii). This point is declared as either a weak minimizer or dead-point based on whether or not negative curvature was encountered at a previous iteration. If $p^T H p < -|z_{\nu_s}|$, then the curvature is considered to be negative and the objective is unbounded below along p. In this case, either a constraint must be blocking or the problem is unbounded. (As z_{ν_s} is considered to be zero, any "sign" attributed to z_{ν_s} for the identification of the blocking variable is based on which of the upper or lower bounds on x_{ν_s} is nonbasic.) If the step to a blocking constraint is zero (i.e., the maximum feasible step α_F is zero), then phase 3 has confirmed that the final point is a dead-point and the algorithm is terminated. If $\alpha_F > 0$, then the step is taken and `SQIC` returns to phase 2.

7.2. Results

A total of 253 QPs were identified from the CUTEst [39] test set. No linear programs were tested because all of the codes under consideration revert to the simplex method when the objective is linear. The QP problems are grouped into two sets based on the final number of superbasic variables obtained by the default solver `SQIC-LUSOL`. The final number of superbasics can be slightly different when `SQIC` is used with other linear solvers. A test problem is included in the "large" set if the final number of superbasics is greater than 1000

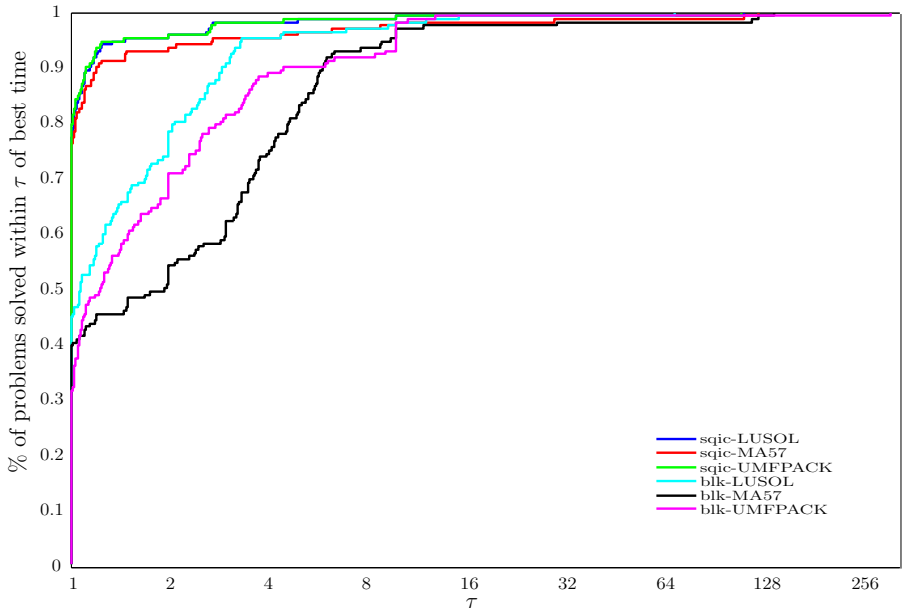

Figure 3: Performance profile of solve times for SQIC on the CUTEst QP test set with a small number of superbasics. The profiles SQIC-LUSOL, SQIC-MA57 and SQIC-UMFPACK refer to versions of SQIC with block-matrix solver options LUSOL, HSL_MA57 and UMFPACK. The profiles with prefix "blk-" correspond to runs for which SQIC was forced to use block-matrix mode regardless of the number of superbasics.

or $\frac{1}{2}(m+n)$. The remaining test problems form the "small" set. The CUTEst set contains 173 small and 80 large problems. A time limit of 5000 seconds was imposed in each case. (In practice, the 5000 second limit is not exact since the time limit is checked every twenty iterations.)

Results are presented for SQIC with its default settings using the three linear solvers HSL_MA57, UMFPACK and the included solver LUSOL, on an iMac with a 3.4GHz Intel Core i7 processor and 16GB of memory. The GNU Fortran compiler gfortran version 4.8.2 was used to compile the code with optimization flag "-O". The results are summarized using performance profiles (in $\log_2$ scale) proposed by Dolan and Moré [14]. In addition to the runs with default settings, all problems were run using so-called "forced" block-matrix mode in which the block-matrix method was used to solve every KKT system. These results are denoted by the prefix "blk-" in the performance profiles.

Only two problems failed to solve with the default settings. Problem CVXQP3 timed out with UMFPACK, and problem UBH1 encountered numerical difficulties in block-matrix mode with HSL_MA57. UBH1 was solved successfully with a setting of 10^9 for the bound on the Schur-complement matrix.

Performance profiles for problems with a "small" number of superbasics are shown in Figure 3. The performance of SQIC-LUSOL, SQIC-UMFPACK and SQIC-MA57 on this subset is similar because SQIC stayed in variable-reduction mode for almost all the iterations and did not use the block-matrix solver. It is clear from the profile that variable-reduction mode with any of the three solvers is significantly more effective than using only block-matrix mode on this set of problems. The weaker performance of SQIC in "forced" block-matrix mode can be attributed to the overhead of factoring the larger block matrix. In addition, because the final number of superbasics is small, solvers that used a non-vertex starting

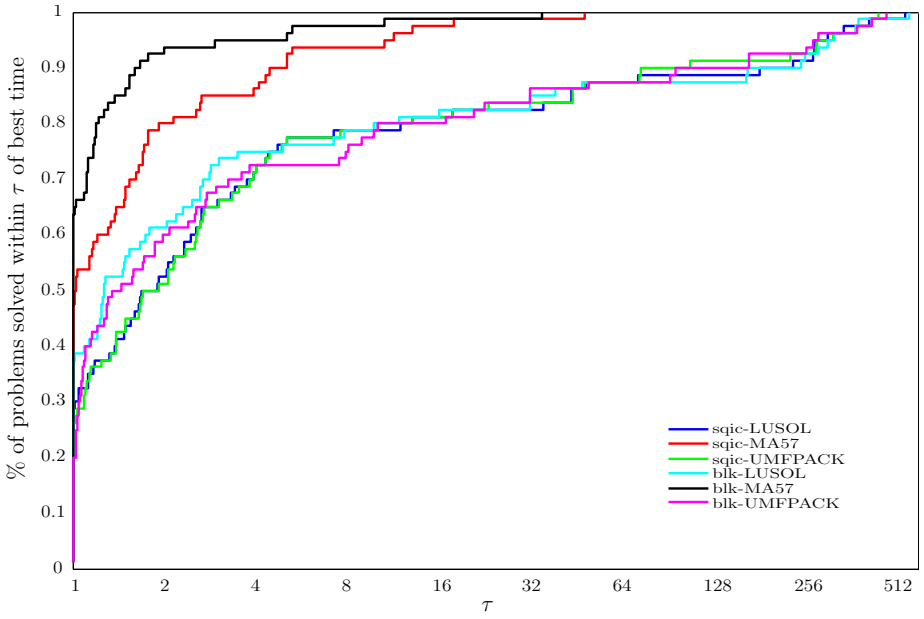

Figure 4: Performance profile of solve times for `SQIC` on the CUTEst QP test set with a large number of superbasics. The profiles `SQIC-LUSOL`, `SQIC-MA57` and `SQIC-UMFPACK` refer to versions of `SQIC` with block-matrix solver options `LUSOL`, `HSL_MA57` and `UMFPACK`. The profiles with prefix "`blk-`" correspond to runs for which `SQIC` was forced to use block-matrix mode regardless of the number of superbasics.

Table 2: Statistics associated with the procedure used to define a second-order consistent basis for `SQIC-MA57` are presented. The column "`nTmp`" gives the number of temporary constraints computed. "`Time`" is the number of seconds to compute the constraints, and "`% Time`" is the percentage of the total solve time required to identify the temporary constraints. The column "`Dens`" is the density of the matrix $H_B + \rho A_B^T A_B$ as a percentage and "`DensL`" is the density of the factor L.

Default mode											
Name	nTmp	Time	% Time	Dens	DensL	Name	nTemp	Time	% Time	Dens	DensL
BLOCKQP2	1	31.01	63.74	49.97	49.97	STNQP1	348	1.11	8.43	0.07	0.30
BLOCKQP4	1	30.84	60.56	49.97	49.97	STNQP2	769	2.37	4.05	0.12	0.91

Block-matrix mode											
Name	nTmp	Time	% Time	Dens	DensL	Name	nTmp	Time	% Time	Dens	DensL
AONNDNIL	23	102.09	75.84	9.27	16.27	HATFLDH	1	0.00	0.00	31.25	43.75
AONSDSIL	15	101.43	73.45	9.27	16.28	HS3MOD	1	0.00	0.00	66.67	66.67
A2NSDSIL	20	101.83	71.30	9.27	16.28	MARATOSB	2	0.00	0.00	66.67	66.67
A5NSDSIL	10	101.50	70.34	9.29	16.30	MPC15	1	0.11	8.58	0.52	2.16
BLOCKQP2	1	30.94	63.62	49.97	49.97	MPC4	1	0.11	7.76	0.54	2.23
BLOCKQP4	1	30.86	58.60	49.97	49.97	MPC8	1	0.10	7.19	0.53	2.18
BQPGAUSS	20	0.60	79.57	50.03	50.03	STATIC3	58	0.00	0.00	0.96	2.28
GMNCASE1	1	0.03	32.08	22.05	25.09	STNQP1	348	1.07	8.29	0.07	0.30
GOULDQP1	5	0.00	0.00	9.57	24.02	STNQP2	769	2.37	4.06	0.12	0.91

point or started with a larger number of superbasics (e.g., HSL_MA57) require more iterations to remove the extra superbasics from the basis than solvers that start at a vertex (where the number of superbasics is zero).

On problems with a "large" final number of superbasics, the performance profiles of Figure 4 indicate that SQIC is the most efficient when using HSL_MA57 as the block solver. HSL_MA57 allows SQIC to start at points with an arbitrary number of superbasic variables, giving it an advantage over the other solvers, which must start at a vertex. These solvers require many more iterations than HSL_MA57 to build up to the final number of superbasics. Figure 4 also highlights the benefit of allowing the user to start phase 2 in block-matrix mode when it is known in advance that the number of superbasics is large. The performance gap between the two modes involving HSL_MA57 is likely due to the large number of superbasics: in this test set, the number of superbasics is large enough to make variable-reduction less efficient, but not large enough to cause SQIC to switch to block-matrix mode.

Table 2 provides some statistics associated with the procedure used to define a second-order consistent basis for SQIC-MA57. For each problem that required the procedure, information is provided on the number of temporary constraints that were imposed, the density of the matrix $H_B + \rho A_B^T A_B$, and the amount of time needed to assemble the matrix for factorization. In general, the computation time is related to the size of the problem and the density of the matrix $H_B + \rho A_B^T A_B$. For many of the larger problems, in particular AONNDNIL, AONSDSIL, A2NSDSIL, and A5NSDSIL, the time needed to identify the temporary constraints is a significant percentage of the total solution time.

Table 3 lists problems that computed at least one direction of negative curvature. The table also provides statistics on the total number of subspace minimizers and the number of subspace minimizers at which a direction of negative curvature was computed.

Results are also presented that allow a comparison between SQIC and the convex QP solver SQOPT [27], which is an implementation of a reduced-Hessian, reduced-gradient active-set method. The method of SQOPT removes a variable from the nonbasic set at the *start* of a sequence of intermediate iterates and maintains the matrix factors associated with the variable-reduction method described in Section 5.1. With this method, the reduced Hessian $Z^T H Z$ is positive semidefinite with at most one zero eigenvalue. If the reduced Hessian is positive definite, a suitable direction is computed from the equations

$$Z^T H Z p_S = -Z^T g, \tag{7.3}$$

which are solved using a dense Cholesky factor of $Z^T H Z$. If the reduced Hessian is singular, the Cholesky factor is used to define p_S such that $Z^T H Z p_S = 0$ and $p_S^T Z^T g < 0$. If the number of superbasics is large, then solving (7.3) becomes expensive. By default, SQOPT switches to a conjugate-gradient method to solve for a direction, when n_S is greater than 2000. Therefore, it is to be expected that SQIC, which utilizes the block-matrix method, will provide superior performance when there are many superbasics.

Figures 5 and 6 are the performance profiles of SQIC and SQOPT on a set of 145 convex CUTEst problems with a small and large number of superbasics. The test set consists of problems that were identified as being convex in [46] and by checking the definiteness of the Hessian matrix of all the CUTEst problems in MATLAB. Of the 145 convex problems, 70 are in the "small" set and 75 in the "large" set. As expected, Figure 5 shows that SQOPT is the best solver for convex problems with a small number of superbasics. For the "large" convex problem set, SQIC is superior to SQOPT for all solvers. In particular, SQIC-MA57 shows marked improvement over SQOPT, demonstrating the superiority of the block-matrix approach in this context.

Overall, of the 253 problems that were solved by SQIC-MA57, 41 terminated at a dead point and 55 terminated at a weak minimizer. Table 4 illustrates the result of running SQIC-MA57 both with and without the "phase 3" option enabled. With the additional phase, 10 of the

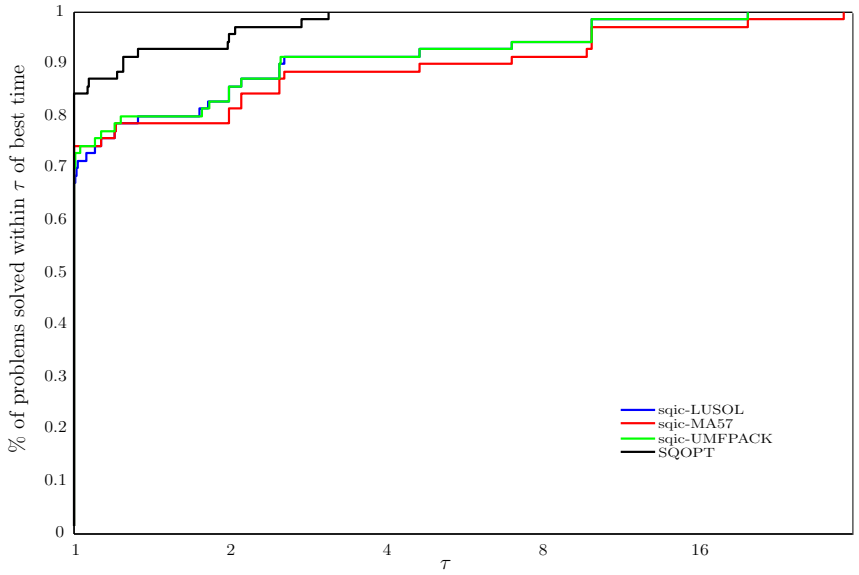

Figure 5: Performance profile of solve times for `SQIC` and `SQOPT` on convex CUTEst problems with a small number of superbasics. The profiles `SQIC-LUSOL`, `SQIC-MA57` and `SQIC-UMFPACK` refer to versions of `SQIC` with block-matrix solver options `LUSOL`, `HSL_MA57` and `UMFPACK`.

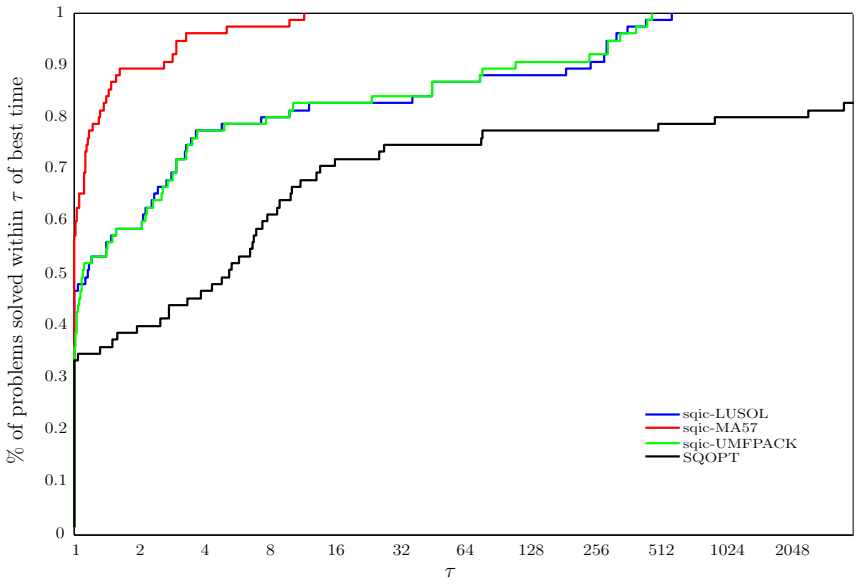

Figure 6: Performance profile of solve times for `SQIC` and `SQOPT` on convex CUTEst problems with a large number of superbasics. The profiles `SQIC-LUSOL`, `SQIC-MA57` and `SQIC-UMFPACK` refer to versions of `SQIC` with block-matrix solver options `LUSOL`, `HSL_MA57` and `UMFPACK`.

Table 3: A list of problems that had directions of negative curvature. `#smin` is the number of subspace minimizers where a direction of negative curvature was computed. `Total` is the total number of subspace minimizers. A column with two entries separated by a "/" indicates a problem for which the information differed depending on the linear solver. The first entry is information for the LU solver (`LUSOL` or `UMFPACK`); the second is for the LDL^T solver `HSL_MA57`.

Name	#smin	Total	Name	#smin	Total	Name	#smin	Total
AONNDNDL	22	1963	BLOCKQP4	1/0	7512/505	MPC8	15	510
AONNDNIL	105	248	BLOCKQP5	4999	5013	MPC9	15	525
AONNDNSL	129	1524	GOULDQP1	6	12	NCVXBQP1	9591	10009
AONNSNSL	76	1835	LEUVEN2	2	178	NCVXBQP2	8184	11137
AONSDSIL	21	86	LEUVEN3	338	988	NCVXBQP3	4243	10808
AONSSSSL	5	182	LEUVEN4	345	1291	NCVXQP1	630	631
A2NNDNDL	76	2551	LEUVEN5	338	988	NCVXQP2	729	852
A2NNDNSL	153	2600	LEUVEN6	205	478	NCVXQP3	252	693
A2NNSNSL	10	313	MARATOSB	1	1	NCVXQP4	748	749
A2NSDSIL	156	2003	MPC10	13	507	NCVXQP5	639	691
A2NSDSSL	3	2007	MPC11	18	338	NCVXQP6	331	540
A2NSSSSL	2	515	MPC12	12	589	NCVXQP7	351	352
A5NNDNDL	375	4803	MPC13	19	496	NCVXQP8	457	463
A5NNDNSL	7	2672	MPC14	15	442	NCVXQP9	158	463
A5NNSNSL	230	2849	MPC15	15	410	PORTSNQP	1	260
A5NSDSIL	256	2338	MPC16	20	386	QPNBAND	25000	50001
A5NSDSSL	60	5691	MPC2	6	449	QPNBOEI1	14	313
A5NSSSSL	1369	1987	MPC3	13	445	QPNBOEI2	3	90
BIGGSC4	2	7	MPC4	8	497	QPNSTAIR	1	96
BLOCKQP1	4999	5011	MPC5	16	417	STATIC3	2	39
BLOCKQP2	1/0	7512/4	MPC6	17	479	STNQP1	513/126	6423/127
BLOCKQP3	4999	5011	MPC7	11	434	STNQP2	822/250	4099/292

41 dead points and 30 of the 55 weak minimizers terminated at an optimal point, i.e., at a point satisfying the second-order sufficient conditions for optimality. In all but two cases, phase 3 verified that the weak minimizer or dead point was optimal, i.e., the additional phase-3 iterations added superbasic variables "in place" until the phase was terminated. Nonconvex problems `AONNDNIL` and `BIGGSC4` moved from a dead point to the locally optimal solution in phase 3.

Acknowledgments

We extend sincere thanks to Michael Saunders, Iain Duff and Nick Gould for their assistance during the development of `SQIC` and the computational tests of Section 7. We are also grateful to the three referees for constructive comments that resulted in significant improvements in the manuscript.

References

[1] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM J. Matrix Anal. Appl.*, 23(1):15–41 (electronic), 2001. 21

[2] C. Ashcraft and R. Grimes. SPOOLES: an object-oriented sparse matrix library. In *Proceedings of the Ninth SIAM Conference on Parallel Processing for Scientific Computing 1999 (San Antonio, TX)*, page 10, Philadelphia, PA, 1999. SIAM. 21

[3] R. A. Bartlett and L. T. Biegler. QPSchur: a dual, active-set, Schur-complement method for large-scale and structured convex quadratic programming. *Optim. Eng.*, 7(1):5–32, 2006. 2

[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: Festschrift for Klaus Ritter*, pages 57–76. Physica, Heidelberg, 1996. 8

[5] J. Bisschop and A. Meeraus. Matrix augmentation and partitioning in the updating of the basis inverse. *Math. Program.*, 13:241–254, 1977. 21

[6] J. M. Borwein. Necessary and sufficient conditions for quadratic minimality. *Numer. Funct. Anal. and Optimiz.*, 5:127–140, 1982. 4

[7] J. R. Bunch and L. Kaufman. A computational method for the indefinite quadratic programming problem. *Linear Algebra Appl.*, 34:341–370, 1980. 2

[8] L. B. Contesse. Une caractérisation complète des minima locaux en programmation quadratique. *Numer. Math.*, 34:315–332, 1980. 4, 28

[9] R. W. Cottle, G. J. Habetler, and C. E. Lemke. On classes of copositive matrices. *Linear Algebra Appl.*, 3:295–310, 1970. 4

[10] T. A. Davis. Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. *ACM Trans. Math. Software*, 30(2):196–199, 2004. 26

[11] T. A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Software*, 30(2):167–195, 2004. 26

[12] T. A. Davis and I. S. Duff. An unsymmetric-pattern multifrontal method for sparse *LU* factorization. *SIAM J. Matrix Anal. Appl.*, 18(1):140–158, 1997. 26

[13] T. A. Davis and I. S. Duff. A combined unifrontal/multifrontal method for unsymmetric sparse matrices. *ACM Trans. Math. Software*, 25(1):1–20, 1999. 26

[14] E. D. Dolan and J. J. Moré. Benchmarking optimization software with COPS. Technical Memorandum ANL/MCS-TM-246, Argonne National Laboratory, Argonne, IL, 2000. 29

[15] I. S. Duff. MA57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM Trans. Math. Software*, 30(2):118–144, 2004. 21

[16] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Trans. Math. Software*, 9:302–325, 1983. 21

[17] S. K. Eldersveld and M. A. Saunders. A block-LU update for large-scale linear programming. *SIAM J. Matrix Anal. Appl.*, 13:191–201, 1992. 22

[18] H. J. Ferreau. *An Online Active Set Strategy for Fast Solution of Parametric Quadratic Programs with Applications to Predictive Engine Control*. PhD thesis, University of Heidelberg, 2006. 8

[19] H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *Internat. J. Robust Nonlinear Control*, 18(8):816–830, 2008. 8

[20] R. Fletcher. A general quadratic programming algorithm. *J. Inst. Math. Applics.*, 7:76–91, 1971. 2, 6

[21] A. Forsgren, P. E. Gill, and W. Murray. On the identification of local minimizers in inertia-controlling methods for quadratic programming. *SIAM J. Matrix Anal. Appl.*, 12:730–746, 1991. 5, 28

[22] M. P. Friedlander and S. Leyffer. Global and finite termination of a two-phase augmented Lagrangian filter method for general quadratic programs. *SIAM J. Sci. Comput.*, 30(4):1706–1729, 2008. 2

[23] P. E. Gill. Recent developments in numerically stable methods for linear programming. *IMA Bulletin*, 10:180–186, 1973. 15

[24] P. E. Gill, N. I. M. Gould, W. Murray, M. A. Saunders, and M. H. Wright. A weighted Gram-Schmidt method for convex quadratic programming. *Math. Program.*, 30:176–195, 1984. 2

[25] P. E. Gill and W. Murray. Numerically stable methods for quadratic programming. *Math. Program.*, 14:349–372, 1978. 2

[26] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Rev.*, 47:99–131, 2005. 23

[27] P. E. Gill, W. Murray, and M. A. Saunders. User's guide for SQOPT Version 7: Software for large-scale linear and quadratic programming. Numerical Analysis Report 06-1, Department of Mathematics, University of California, San Diego, La Jolla, CA, 2006. 2, 25, 31

[28] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Sparse matrix methods in optimization. *SIAM J. Sci. Statist. Comput.*, 5(3):562–589, 1984. 3, 22

[29] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Maintaining *LU* factors of a general sparse matrix. *Linear Algebra Appl.*, 88/89:239–270, 1987. 26

[30] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. A practical anti-cycling procedure for linearly constrained optimization. *Math. Program.*, 45:437–474, 1989. 27

[31] 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 Numerical Computation*, pages 113–138. Oxford University Press, 1990. 2, 21

[32] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. Inertia-controlling methods for general quadratic programming. *SIAM Rev.*, 33(1):1–36, 1991. 2

[33] P. E. Gill, W. Murray, and M. H. Wright. *Numerical Linear Algebra and Optimization, volume 1*. Addison-Wesley Publishing Company, Redwood City, 1991. 5

[34] P. E. Gill and E. Wong. Sequential quadratic programming methods. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, volume 154 of *The IMA Volumes in Mathematics and its Applications*, pages 147–224. Springer New York, 2012. 10.1007/978-1-4614-1927-3_6. 2

[35] D. Goldfarb and A. Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Math. Programming*, 27(1):1–33, 1983. 2

[36] N. I. M. Gould. On practical conditions for the existence and uniqueness of solutions to the general equality quadratic programming problem. *Math. Program.*, 32:90–99, 1985. 6

[37] N. I. M. Gould. An algorithm for large-scale quadratic programming. *IMA J. Numer. Anal.*, 11(3):299–324, 1991. 2

[38] N. I. M. Gould, D. Orban, and P. L. Toint. `GALAHAD`, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. *ACM Trans. Math. Software*, 29(4):353–372, Dec 2003. 2

[39] N. I. M. Gould, D. Orban, and P. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads. Technical report, Rutherford Appleton Laboratory, Chilton, England, 2013. 3, 28

[40] N. I. M. Gould and P. L. Toint. An iterative working-set method for large-scale nonconvex quadratic programming. *Appl. Numer. Math.*, 43(1-2):109–128, 2002. 19th Dundee Biennial Conference on Numerical Analysis (2001). 2

[41] N. I. M. Gould and P. L. Toint. Numerical methods for large-scale non-convex quadratic programming. In *Trends in industrial and applied mathematics (Amritsar, 2001)*, volume 72 of *Appl. Optim.*, pages 149–179. Kluwer Acad. Publ., Dordrecht, 2002. 2

[42] J. A. J. Hall and K. I. M. McKinnon. The simplest examples where the simplex method cycles and conditions where EXPAND fails to prevent cycling. Technical Report MS 96-010, Department of Mathematics and Statistics, University of Edinburgh, 1996. 27

[43] J. D. Hogg and J. A. Scott. HSL_MA97: a bit-compatible multifrontal code for sparse symmetric systems. Technical report, Rutherford Appleton Laboratory, Oxon, UK, 2011. 21

[44] HSL. A collection of Fortran codes for large-scale scientific computation. http://www.hsl.rl.ac.uk, 2013. 26

[45] H. M. Huynh. *A Large-Scale Quadratic Programming Solver Based on Block-LU Updates of the KKT System*. PhD thesis, Program in Scientific Computing and Computational Mathematics, Stanford University, Stanford, CA, 2008. 3, 22, 26

[46] C. M. Maes. *A Regularized Active-Set Method for Sparse Convex Quadratic Programming*. PhD thesis, Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA, August 2010. 31

[47] A. Majthay. Optimality conditions for quadratic programming. *Math. Programming*, 1:359–365, 1971. 4

[48] K. G. Murty and S. N. Kabadi. Some NP-complete problems in quadratic and nonlinear programming. *Math. Program.*, 39:117–129, 1987. 4

[49] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999. 5

[50] P. M. Pardalos and G. Schnitger. Checking local optimality in constrained quadratic programming is NP-hard. *Oper. Res. Lett.*, 7(1):33–35, 1988. 4

[51] P. M. Pardalos and S. A. Vavasis. Quadratic programming with one negative eigenvalue is NP-hard. *J. Global Optim.*, 1(1):15–22, 1991. 4

[52] M. J. D. Powell. On the quadratic programming algorithm of Goldfarb and Idnani. *Math. Programming Stud.*, 25:46–61, 1985. 2

[53] D. Ruiz. A scaling algorithm to equilibrate both row and column norms in matrices. Report RAL-TR-2001-034, Rutherford Appleton Laboratory, Oxon, UK, September 2001. 25

[54] M. A. Saunders. LUMOD: Updating a dense square factorization LC = U. http://www.stanford.edu/group/SOL/software/lumod.html. 26

[55] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. In *Computational science—ICCS 2002, Part II (Amsterdam)*, volume 2330 of *Lecture Notes in Comput. Sci.*, pages 355–363. Springer, Berlin, 2002. 21

[56] J. A. Tomlin. On pricing and backward transformation in linear programming. *Math. Programming*, 6:42–47, 1974. 15

[57] E. Wong. *Active-Set Methods for Quadratic Programming*. PhD thesis, Department of Mathematics, University of California San Diego, La Jolla, CA, 2011. 14

Table 4: Problems that originally ended on a weak minimizer or dead point using SQIC with HSL_MA57 are re-run with Phase 3. Results for the problems that ended optimally with Phase 3 are listed.

	Weak minimizer without Phase 3				Optimal point with Phase 3			
Name	Objective	# Itn	nS	Time	Objective	# Itn	nS	Time
A2ENDNDL	0.0000E+00	6805	47	15.04	0.0000E+00	7743	985	18.03
A2ENINDL	0.0000E+00	6703	57	15.62	0.0000E+00	7613	967	18.43
A2ESDNDL	9.4684E-25	6329	74	15.98	9.4684E-25	7240	985	18.84
A2ESINDL	0.0000E+00	6690	44	14.93	0.0000E+00	7613	967	17.69
A2NSDSDL	4.8243E-11	40497	5	149.13	2.5794E-11	40911	419	150.56
A5ENDNDL	0.0000E+00	5903	230	14.13	0.0000E+00	8134	2461	24.80
A5ENINDL	0.0000E+00	5914	222	13.82	0.0000E+00	8221	2529	24.83
A5ESDNDL	0.0000E+00	5674	238	14.07	0.0000E+00	7897	2461	24.67
A5ESINDL	0.0000E+00	5755	197	13.33	0.0000E+00	8087	2529	24.54
A5NSDSDL	1.2278E-11	38515	29	156.69	-6.7193E-11	39636	1150	161.10
ALLINQP	-5.4813E+03	16957	9820	91.37	-5.4813E+03	36597	29460	274.29
AUG3DCQP	6.1560E+04	22216	17665	120.56	6.1560E+04	22264	17713	121.00
CHENHARK	-2.0000E+00	2017	2984	10.46	-2.0000E+00	2033	3000	10.49
GOULDQP3	2.3796E-05	5814	4988	20.83	2.3796E-05	5856	5030	21.13
GRIDNETC	1.6187E+02	1391	2578	4.89	1.6187E+02	1392	2579	4.99
HATFLDH	-2.4500E+01	4	0	0.00	-2.4500E+01	5	1	0.00
LEUVEN1	-1.5243E+07	1515	14	0.35	-1.5243E+07	1614	113	0.36
LISWET10	9.8965E+00	34	18	0.02	9.8965E+00	75	59	0.02
LISWET11	9.9054E+00	49	29	0.02	9.9054E+00	60	40	0.02
LISWET12	3.4752E+02	24	5	0.01	3.4752E+02	28	9	0.01
LISWET8	1.4313E+02	28	16	0.02	1.4313E+02	149	137	0.04
LISWET9	3.9292E+02	18	7	0.01	3.9292E+02	35	24	0.02
ODNAMUR	9.2366E+03	3729	5512	192.82	9.2366E+03	4504	6287	211.11
PENTDI	-7.5000E-01	3	2	0.02	-7.5000E-01	2499	2498	4.33
POWELL20	6.5120E+09	2500	1	5.89	6.5120E+09	2502	3	5.92
PRIMAL3	-1.3576E-01	102	648	0.36	-1.3576E-01	103	649	0.36
QPCBOEI1	1.1504E+07	700	113	0.03	1.1504E+07	703	116	0.03
QPCSTAIR	6.2044E+06	311	21	0.02	6.2044E+06	359	69	0.02
RDW2D52F	8.6159E-03	71	37	0.00	8.6159E-03	72	38	0.00
SOSQP2	-1.2487E+03	4777	1251	9.99	-1.2487E+03	4778	1252	9.93
	Dead point without Phase 3				Optimal point with Phase 3			
Name	Objective	# Itn	nS	Time	Objective	# Itn	nS	Time
AONNDNIL	6.0072E+01	12049	55	32.64	5.8632E+01	12056	54	32.67
A5NNDNDL	1.0364E-08	55587	198	231.96	1.0101E-08	55646	251	233.31
BIGGSC4	-2.4375E+01	11	1	0.00	-2.4500E+01	12	1	0.00
MPC16	-1.5034E+07	1081	16	0.21	-1.5034E+07	1202	137	0.23
MPC4	-1.5033E+07	1357	21	0.28	-1.5033E+07	1468	132	0.30
MPC6	-1.5034E+07	1245	18	0.26	-1.5034E+07	1355	128	0.28
NCVXQP2	-5.7759E+07	991	0	0.06	-5.7759E+07	992	0	0.06
QPNBOEI1	6.7367E+06	683	92	0.03	6.7367E+06	685	94	0.03
QPNBOEI2	1.3683E+06	229	27	0.01	1.3683E+06	236	34	0.01
QPNSTAIR	5.1460E+06	349	20	0.02	5.1460E+06	390	59	0.02