16.4 ACTIVE-SET METHODS FOR CONVEX QP

We now describe active-set methods, which are generally the most effective methods for small- to medium-scale problems. We start by discussing the convex case, in which the matrix G in (16.1a) is positive semidefinite. Since the feasible region defined by (16.1b), (16.1c) is a convex set, any local solution of the QP is a global minimizer. The case in which G is an indefinite matrix raises complications in the algorithms; our discussion of this case appears in the next section.

Recall our definition (16.25) above of the active set $\mathcal{A}(x)$ at the optimal point x^*. We will call it an *optimal* active set.

If $\mathcal{A}(x^*)$ were known in advance, we could find the solution by applying one of the techniques for equality-constrained QP of Section 16.2 to the problem

$$\min_x q(x) = \tfrac{1}{2}x^T Gx + x^T d \quad \text{subject to} \quad a_i^T x = b_i, \quad i \in \mathcal{A}(x^*).$$

Of course, we usually don't have prior knowledge of $\mathcal{A}(x^*)$, and as we will now see, determination of this set is the main challenge facing algorithms for inequality-constrained QP.

We have already encountered an active-set approach for linear programming in Chapter 13, namely the simplex method. An active-set method starts by making a guess of the optimal active set, and if this guess turns out to be incorrect, it repeatedly uses gradient and Lagrange multiplier information to drop one index from the current estimate of $\mathcal{A}(x^*)$ and add a new index. Active-set methods for QP differ from the simplex method in that the iterates may not move from one vertex of the feasible region to another. Some iterates (and, indeed, the solution of the problem) may lie at other points on the boundary or interior of the feasible region.

Active-set methods for QP come in three varieties, known as *primal*, *dual*, and *primal–dual*. We restrict our discussion to primal methods, which generate iterates that remain feasible with respect to the primal problem (16.1) while steadily decreasing the primal objective function $q(\cdot)$.

Primal active-set methods usually start by computing a feasible initial iterate x_0, and then ensure that all subsequent iterates remain feasible. They find a step from one iterate to the next by solving a quadratic subproblem in which a subset of the constraints in (16.1b), (16.1c) is imposed as equalities. This subset is referred to as the *working set* and is denoted at the kth iterate x_k by $\mathcal{W}_k$. It consists of all the equality constraints $i \in \mathcal{E}$ (see 16.1b) together with some—but not necessarily all—of the active inequality constraints. An important requirement we impose on $\mathcal{W}_k$ is that the gradients a_i of the constraints in the working set be linearly independent, even when the full set of active constraints at that point has linearly dependent gradients. Later, we discuss how this condition can be enforced without compromising convergence of the algorithm to a solution.

Given an iterate x_k and the working set $\mathcal{W}_k$, we first check whether x_k minimizes the quadratic q in the subspace defined by the working set. If not, we compute a step p by solving an equality-constrained QP subproblem in which the constraints corresponding to the working set $\mathcal{W}_k$ are regarded as equalities and all other constraints are temporarily disregarded. To express this subproblem in terms of the step p, we define

$$p = x - x_k, \qquad g_k = Gx_k + d,$$

and by substituting for x into the objective function (16.1a), we find that

$$q(x) = q(x_k + p) = \tfrac{1}{2} p^T G p + g_k^T p + c,$$

where $c = \frac{1}{2} x_k^T G x_k + d^T x_k$ is a constant term. Since we can drop c from the objective without changing the solution of the problem, we can write the QP subproblem to be solved at the kth iteration as follows:

$$\min_p \ \tfrac{1}{2} p^T G p + g_k^T p \tag{16.27a}$$

$$\text{subject to } a_i^T p = 0 \text{ for all } i \in \mathcal{W}_k. \tag{16.27b}$$

We denote the solution of this subproblem by p_k. Note that for each $i \in \mathcal{W}_k$, the term $a_i^T x$ does not change as we move along p_k, since we have $a_i^T (x_k + p_k) = a_i^T x_k = b_i$. It follows that since the constraints in $\mathcal{W}_k$ were satisfied at x_k, they are also satisfied at $x_k + \alpha p_k$, for any value of α. When G is positive definite, the solution of (16.27b) can be computed by any of the techniques described in Section 16.2.

Suppose for the moment that the optimal p_k from (16.27) is nonzero. We need to decide how far to move along this direction. If $x_k + p_k$ is feasible with respect to all the constraints, we set $x_{k+1} = x_k + p_k$. Otherwise, we set

$$x_{k+1} = x_k + \alpha_k p_k, \tag{16.28}$$

where the step-length parameter α_k is chosen to be the largest value in the range $[0, 1)$ for which all constraints are satisfied. We can derive an explicit definition of α_k by considering what happens to the constraints $i \notin \mathcal{W}_k$, since the constraints $i \in \mathcal{W}_k$ will certainly be satisfied regardless of the choice of α_k. If $a_i^T p_k \geq 0$ for some $i \notin \mathcal{W}_k$, then for all $\alpha_k \geq 0$ we have $a_i^T (x_k + \alpha_k p_k) \geq a_i^T x_k \geq b_i$. Hence, this constraint will be satisfied for all nonnegative choices of the step-length parameter. Whenever $a_i^T p_k < 0$ for some $i \notin \mathcal{W}_k$, however, we have that $a_i^T (x_k + \alpha_k p_k) \geq b_i$ only if

$$\alpha_k \leq \frac{b_i - a_i^T x_k}{a_i^T p_k}.$$

Since we want α_k to be as large as possible in $[0, 1]$ subject to retaining feasibility, we have the following definition:

$$\alpha_k \stackrel{\text{def}}{=} \min \left(1, \min_{i \notin \mathcal{W}_k,\, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k} \right). \tag{16.29}$$

We call the constraints i for which the minimum in (16.29) is achieved the *blocking constraints*. (If $\alpha_k = 1$ and no new constraints are active at $x_k + \alpha_k p_k$, then there are no blocking constraints on this iteration.) Note that it is quite possible for α_k to be zero, since we could have $a_i^T p_k < 0$ for some constraint i that is active at x_k but not a member of the current working set $\mathcal{W}_k$.

If $\alpha_k < 1$, that is, the step along p_k was blocked by some constraint not in $\mathcal{W}_k$, a new working set $\mathcal{W}_{k+1}$ is constructed by adding one of the blocking constraints to $\mathcal{W}_k$.

We continue to iterate in this manner, adding constraints to the working set until we reach a point $\hat{x}$ that minimizes the quadratic objective function over its current working set $\hat{\mathcal{W}}$. It is easy to recognize such a point because the subproblem (16.27) has solution $p = 0$. Since $p = 0$ satisfies the optimality conditions (16.5) for (16.27), we have that

$$\sum_{i \in \hat{\mathcal{W}}} a_i \hat{\lambda}_i = g = G\hat{x} + d, \tag{16.30}$$

for some Lagrange multipliers $\hat{\lambda}_i$, $i \in \hat{\mathcal{W}}$. It follows that $\hat{x}$ and $\hat{\lambda}$ satisfy the first KKT condition (16.26a), if we define the multipliers corresponding to the inequality constraints that are not in the working set to be zero. Because of the control imposed on the step-length, $\hat{x}$ is also feasible with respect to all the constraints, so the second and third KKT conditions (16.26b) and (16.26c) are satisfied by $\hat{x}$.

We now examine the signs of the multipliers corresponding to the inequality constraints in the working set, that is, the indices $i \in \hat{\mathcal{W}} \cap \mathcal{I}$. If these multipliers are all nonnegative, the fourth KKT condition (16.26d) is also satisfied, so we conclude that $\hat{x}$ is a KKT point for the original problem (16.1). In fact, since G is positive semidefinite, we can show that $\hat{x}$ is a local minimizer. When G is positive definite, $\hat{x}$ is a strict local minimizer.

If, on the other hand, one of the multipliers $\hat{\lambda}_j$, $j \in \hat{\mathcal{W}} \cap \mathcal{I}$, is negative, the condition (16.26d) is not satisfied, and the objective function $q(\cdot)$ may be decreased by dropping this constraint, as shown in Section 12.2. We then remove an index j corresponding to one of the negative multipliers from the working set and solve a new subproblem (16.27) for the new step. We show in the following theorem that this strategy produces a direction p at the next iteration that is feasible with respect to the dropped constraint. We continue to assume that the vectors in the working set are linearly independent, and we defer a discussion of how this can be achieved to the next section when the algorithm has been fully stated.

Theorem 16.4.

Suppose that the point $\hat{x}$ satisfies first-order conditions for the equality-constrained subproblem with working set $\hat{\mathcal{W}}$; that is, equation (16.30) is satisfied along with $a_i^T \hat{x} = b_i$ for all $i \in \hat{\mathcal{W}}$. Suppose, too, that the constraint gradients a_i, $i \in \hat{\mathcal{W}}$, are linearly independent, and that there is an index $j \in \hat{\mathcal{W}}$ such that $\hat{\lambda}_j < 0$. Finally, let p be the solution obtained by dropping the constraint j and solving the following subproblem:

$$\min_p \ \tfrac{1}{2} p^T G p + (G\hat{x} + d)^T p, \tag{16.31a}$$

$$\text{subject to } a_i^T p = 0, \text{ for all } i \in \hat{\mathcal{W}} \text{ with } i \neq j. \tag{16.31b}$$

(This is the subproblem to be solved at the next iteration of the algorithm.) Then p is a feasible direction for constraint j, that is, $a_j^T p \geq 0$. Moreover, if p satisfies second-order sufficient conditions for (16.31), then we have that $a_j^T p > 0$, and p is a descent direction for $q(\cdot)$.

PROOF. Since p solves (16.31), we have from the results of Section 16.1 that there are multipliers $\tilde{\lambda}_i$, for all $i \in \hat{\mathcal{W}}$ with $i \neq j$, such that

$$\sum_{i \in \hat{\mathcal{W}},\, i \neq j} \tilde{\lambda}_i a_i = Gp + (G\hat{x} + d). \tag{16.32}$$

In addition, we have by second-order necessary conditions that if Z is a null-space basis vector for the matrix

$$[a_i]^T_{i \in \hat{\mathcal{W}},\, i \neq j},$$

then $Z^T G Z$ is positive semidefinite. Clearly, p has the form $p = Zp_z$ for some vector p_z, so it follows that $p^T G p \geq 0$.

We have made the assumption that $\hat{x}$ and $\hat{\mathcal{W}}$ satisfy the relation (16.30). By subtracting (16.30) from (16.32), we obtain

$$\sum_{i \in \mathcal{W},\, i \neq j} (\tilde{\lambda}_i - \hat{\lambda}_i) a_i - \hat{\lambda}_j a_j = Gp. \tag{16.33}$$

By taking inner products of both sides with p and using the fact that $a_i^T p = 0$ for all $i \in \hat{\mathcal{W}}$ with $i \neq j$, we have that

$$-\hat{\lambda}_j a_j^T p = p^T G p. \tag{16.34}$$

Since $p^T G p \geq 0$ and $\hat{\lambda}_j < 0$ by assumption, it follows immediately that $a_j^T p \geq 0$.

If the second-order sufficient conditions are satisfied, we have that $Z^T G Z$ defined above is positive definite. From (16.34) we can have $a_j^T p = 0$ only if $p^T G p = p_z^T Z^T G Z p_z =$

0, which happens only if $p_z = 0$ and $p = 0$. But if $p = 0$, then by substituting into (16.33) and using linear independence of a_i for $i \in \hat{\mathcal{W}}$, we must have that $\hat{\lambda}_j = 0$, which contradicts our choice of j. We conclude that $p^T G p > 0$ in (16.34), and therefore $a_j^T p > 0$ whenever p satisfies the second-order sufficient conditions for (16.31). □

While any index j for which $\hat{\lambda}_j < 0$ usually will give directions along which the algorithm can make progress, the most negative multiplier is often chosen in practice (and in the algorithm specified below). This choice is motivated by the sensitivity analysis given in Chapter 12, which shows that the rate of decrease in the objective function when one constraint is removed is proportional to the magnitude of the Lagrange multiplier for that constraint.

We conclude with a result that shows that whenever p_k obtained from (16.27) is nonzero and satisfies second-order sufficient optimality conditions for the current working set, then it is a direction of strict descent for $q(\cdot)$.

Theorem 16.5.

Suppose that the solution p_k of (16.27) is nonzero and satisfies the second-order sufficient conditions for optimality for that problem. Then the function $q(\cdot)$ is strictly decreasing along the direction p_k.

PROOF. Since p_k satisfies the second-order conditions, that is, $Z^T G Z$ is positive definite for the matrix Z whose columns are a basis of the null space of the constraints (16.27b), we have by applying Theorem 16.2 to (16.27) that p_k is the unique global solution of (16.27). Since $p = 0$ is also a feasible point for (16.27), its objective value in (16.27a) must be larger than that of p_k, so we have

$$\tfrac{1}{2} p_k^T G p_k + g_k^T p_k < 0.$$

Since $p_k^T G p_k \geq 0$ by convexity, this inequality implies that $g_k^T p_k < 0$. Therefore, we have

$$q(x_k + \alpha_k p_k) = q(x_k) + \alpha g_k^T p_k + \tfrac{1}{2}\alpha^2 p_k^T G p_k < q(x_k),$$

for all $\alpha > 0$ sufficiently small. □

A corollary of this result is that when G is positive definite—the *strictly* convex case—the second-order sufficient conditions are satisfied for *all* feasible subproblems of the form (16.27), so that we obtain a strict decrease in $q(\cdot)$ whenever $p_k \neq 0$. This fact is significant in a later section, when we discuss finite termination of the algorithm.

SPECIFICATION OF THE ACTIVE-SET METHOD FOR CONVEX QP

Having given a complete description of the active-set algorithm for convex QP, it is time for the following formal specification:

Algorithm 16.1 (Active-Set Method for Convex QP).

Compute a feasible starting point x_0;
Set $\mathcal{W}_0$ to be a subset of the active constraints at x_0;
for $k = 0, 1, 2, \ldots$
  Solve (16.27) to find p_k;
  if $p_k = 0$
    Compute Lagrange multipliers $\hat{\lambda}_i$ that satisfy (16.30),
      set $\hat{\mathcal{W}} = \mathcal{W}_k$;
    if $\hat{\lambda}_i \geq 0$ for all $i \in \mathcal{W}_k \cap \mathcal{I}$;
      STOP with solution $x^* = x_k$;
    else
      Set $j = \arg\min_{j \in \mathcal{W}_k \cap \mathcal{I}} \hat{\lambda}_j$;
      $x_{k+1} = x_k$; $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \backslash \{j\}$;
  else (* $p_k \neq 0$ *)
    Compute α_k from (16.29);
    $x_{k+1} \leftarrow x_k + \alpha_k p_k$;
    if there are blocking constraints
      Obtain $\mathcal{W}_{k+1}$ by adding one of the blocking
        constraints to $\mathcal{W}_{k+1}$;
    else
      $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k$;
end (for)

Various techniques can be used to determine an initial feasible point. One such is to use the "Phase I" approach described in Chapter 13. Though no significant modifications are needed to generalize this method from linear programming to quadratic programming, we describe a variant here that allows the user to supply an initial estimate $\tilde{x}$ of the vector x. This estimate need not be feasible, but prior knowledge of the QP may be used to select a value of $\tilde{x}$ that is "not too infeasible," which will reduce the work needed to perform the Phase I step.

Given $\tilde{x}$, we define the following feasibility linear program:

$$\begin{aligned} \min_{(x,z)} \quad & e^T z \\ \text{subject to } a_i^T x + \gamma_i z_i &= b_i, \qquad i \in \mathcal{E}, \\ a_i^T x + \gamma_i z_i &\geq b_i \qquad i \in \mathcal{I}, \\ z &\geq 0, \end{aligned}$$

where $e = (1, \ldots, 1)^T$, $\gamma_i = -\text{sign}(a_i^T \tilde{x} - b_i)$ for $i \in \mathcal{E}$, while $\gamma_i = 1$ for $i \in \mathcal{I}$. A feasible initial point for this problem is then

$$x = \tilde{x}, \qquad z_i = |a_i^T \tilde{x} - b_i| \ \ (i \in \mathcal{E}), \qquad z_i = \max(b_i - a_i^T \tilde{x}, 0) \ \ (i \in \mathcal{I}).$$

It is easy to verify that if $\tilde{x}$ is feasible for the original problem (16.1), then $(\tilde{x}, 0)$ is optimal for the feasibility subproblem. In general, if the original problem has feasible points, then the optimal objective value in the subproblem is zero, and any solution of the subproblem yields a feasible point for the original problem. The initial working set $\mathcal{W}_0$ for Algorithm16.1 can be found by taking a linearly independent subset of the active constraints at the x component of the solution of the feasibility problem.

An alternative approach is the so-called "big M" method, which does away with the "Phase I" and instead includes a measure of infeasibility in the objective that is guaranteed to be zero at the solution. That is, we introduce a scalar artificial variable t into (16.1) to measure the constraint violation, and solve the problem

$$\min_{(x,t)} \quad \tfrac{1}{2} x^T G x + x^T d + M t, \tag{16.35a}$$

$$\text{subject to } t \geq (a_i^T x - b_i), \quad i \in \mathcal{E}, \tag{16.35b}$$

$$t \geq -(a_i^T x - b_i), \quad i \in \mathcal{E}, \tag{16.35c}$$

$$t \geq b_i - a_i^T x, \quad i \in \mathcal{I}, \tag{16.35d}$$

$$t \geq 0, \tag{16.35e}$$

for some large positive value of M. It can be shown by applying the theory of exact penalty functions (see Chapter 15) that whenever there exist feasible points for the original problem (16.1), then for all M sufficiently large, the solution of (16.35) will have $t = 0$, with an x component that is a solution for (16.1).

Our strategy is to use some heuristic to choose a value of M and solve (16.35) by the usual means. If the solution we obtain has a positive value of t, we increase M and try again. Note that a feasible point is easy to obtain for the subproblem (16.35): We set $x = \tilde{x}$ (where, as before, $\tilde{x}$ is the user-supplied initial guess) and choose t large enough that all the constraints in (16.35) are satisfied.

This approach is related to the Sℓ_1QP method described in Chapter 18; the main difference is that the "big M" method is based on the infinity norm rather than on the ℓ_1 norm. (See (12.7) in Chapter 12.)

AN EXAMPLE

In this section we use subscripts on the vector x to denote its components, while superscripts denote the iteration index. For example, x^4 denotes the fourth iterate of x, while x_1 denotes the first component.

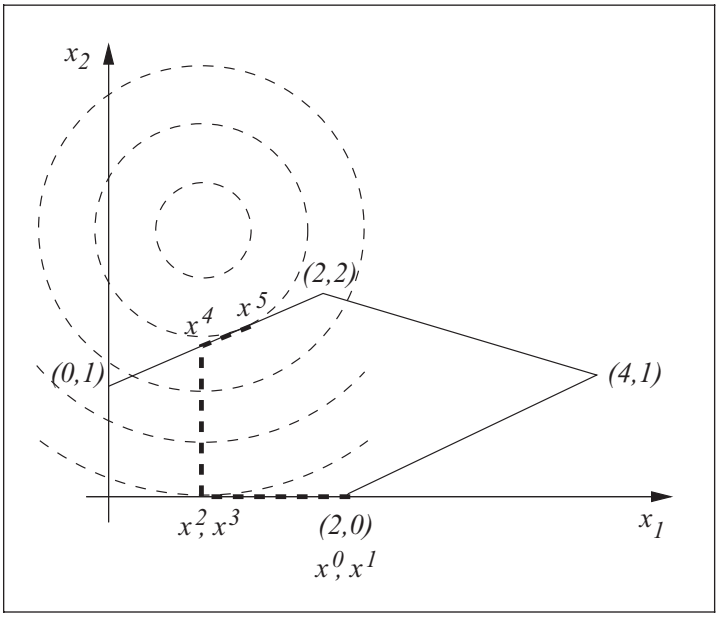

Figure 16.3 Iterates of the active-set method.

❑ EXAMPLE 16.3

We apply Algorithm 16.1 to the following simple 2-dimensional problem, which is illustrated in Figure 16.3.

$$\min_x \; q(x) = (x_1 - 1)^2 + (x_2 - 2.5)^2 \tag{16.36a}$$

$$\text{subject to} \quad x_1 - 2x_2 + 2 \geq 0, \tag{16.36b}$$

$$-x_1 - 2x_2 + 6 \geq 0, \tag{16.36c}$$

$$-x_1 + 2x_2 + 2 \geq 0, \tag{16.36d}$$

$$x_1 \geq 0, \tag{16.36e}$$

$$x_2 \geq 0. \tag{16.36f}$$

We label the constraints, in order, with the indices 1 through 5. For this problem it is easy to determine a feasible initial point; suppose that we choose $x^0 = (2, 0)$. Constraints 3 and 5 are active at this point, and we set $\mathcal{W}_0 = \{3, 5\}$. (Note that we could just as validly have chosen $\mathcal{W}_0 = \{5\}$ or $\mathcal{W}_0 = \{3\}$ or even $\mathcal{W} = \emptyset$; each would lead the algorithm to perform quite differently.)

Since x^0 lies on a vertex of the feasible region, it is obviously a minimizer of the objective function q with respect to the working set $\mathcal{W}_0$; that is, the solution of (16.27) with $k = 0$ is

$p = 0$. We can then use (16.30) to find the multipliers $\hat{\lambda}_3$ and $\hat{\lambda}_5$ associated with the active constraints. Substitution of the data from our problem into (16.30) yields

$$\begin{bmatrix} -1 \\ 2 \end{bmatrix} \hat{\lambda}_3 + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \hat{\lambda}_5 = \begin{bmatrix} 2 \\ -5 \end{bmatrix},$$

which has the solution $(\hat{\lambda}_3, \hat{\lambda}_5) = (-2, -1)$.

We now remove constraint 3 from the working set, since it has the most negative multiplier, and set $\mathcal{W}_1 = \{5\}$. We begin iteration 1 by finding the solution of (16.27) for $k = 1$, which is $p^1 = (-1, 0)^T$. The step-length formula (16.29) yields $\alpha_1 = 1$, and the new iterate is $x^2 = (1, 0)$.

There are no blocking constraints, so that $\mathcal{W}_2 = \mathcal{W}_1 = \{5\}$, and we find at the start of iteration 2 that the solution of (16.27) is again $p^2 = 0$. From (16.30) we deduce that the Lagrange multiplier for the lone working constraint is $\hat{\lambda}_5 = -5$, so we drop 5 from the working set to obtain $\mathcal{W}_3 = \emptyset$.

Iteration 3 starts by solving the unconstrained problem, to obtain the solution $p^3 = (0, 2.5)$. The formula (16.29) yields a step length of $\alpha_3 = 0.6$ and a new iterate $x^4 = (1, 1.5)$. There is a single blocking constraint—constraint 1—so we obtain $\mathcal{W}_4 = \{1\}$. The solution of (16.27) for $k = 4$ is then $p^4 = (0.4, 0.2)$, and the new step length is 1. There are no blocking constraints on this step, so the next working set is unchanged: $\mathcal{W}_5 = \{1\}$. The new iterate is $x^5 = (1.4, 1.7)$.

Finally, we solve (16.27) for $k = 5$ to obtain a solution $p^5 = 0$. The formula (16.30) yields a multiplier $\hat{\lambda}_1 = 1.25$, so we have found the solution. We set $x^* = (1.4, 1.7)$ and terminate.

□

FURTHER REMARKS ON THE ACTIVE-SET METHOD

We noted above that there is flexibility in the choice of the initial working set, and that each initial choice leads to a different iteration sequence. When the initial active constraints have independent gradients, as above, we can include them all in $\mathcal{W}_0$. Alternatively, we can select a subset. For instance, if in the example above we have chosen $\mathcal{W}_0 = \{3\}$, the first iterate would have yielded $p^0 = (0.2, 0.1)$ and a new iterate of $x^1 = (2.2, 0.1)$. If we had chosen $\mathcal{W}_0 = \{5\}$, we would have moved immediately to the new iterate $x^1 = (1, 0)$, without first performing the operation of dropping the index 3, as is done in the example. Finally, if we had selected $\mathcal{W}_0 = \emptyset$, we would obtain $p^1 = (-1, 2.5)$, $\alpha_1 = \frac{2}{3}$, a new iterate of $x^1 = (\frac{4}{3}, \frac{5}{3})$, and a new working set of $\mathcal{W}_1 = \{1\}$. The solution x^* would have been found on the next iteration.

Even if the initial working set $\mathcal{W}_0$ coincides with the initial active set, the sets $\mathcal{W}_k$ and $\mathcal{A}(x^k)$ may differ at later iterations. For instance, when a particular step encounters more than one blocking constraint, just one of them is added to the working set, so the identification

between $\mathcal{W}_k$ and $\mathcal{A}(x^k)$ is broken. Moreover, subsequent iterates differ in general according to what choice is made.

We require that the constraint gradients in $\mathcal{W}_0$ be linearly independent, and our strategy for modifying the working set ensures that this same property holds for all subsequent working sets $\mathcal{W}_k$. When we encounter a blocking constraint on a particular step, a constraint that is not in the working set is encountered during the line search, and its constraint normal cannot be a linear combination of the normals a_i in the current working set (see Exercise 15). Hence, linear independence is maintained after the blocking constraint is added to the working set. On the other hand, deletion of an index from the working set certainly does not introduce linear dependence.

The strategy of removing the constraint corresponding to the most negative Lagrange multiplier often works well in practice, but has the disadvantage that it is susceptible to the scaling of the constraints. (By multiplying constraint i by some factor $\beta > 0$ we do not change the geometry of the optimization problem, but we introduce a scaling of $1/\beta$ to the corresponding multiplier λ_i.) Choice of the most negative multiplier is analogous to Dantzig's original pivot rule for the simplex method in linear programming (see Chapter 13), and as we saw there, more sophisticated strategies that were more resistant to scaling often gave better results. We will not discuss these advanced features here.

Finally, we note that the strategy of adding or deleting at most one constraint at each iteration of the Algorithm 16.1 places a natural lower bound on the number of iterations needed to reach optimality. Suppose, for instance, that we have a problem in which m constraints are active at the solution x^*, but that we start from a point x^0 that is strictly feasible with respect to all the inequality constraints. In this case, the algorithm will need at least m iterations to move from x^0 to x^*. Even more iterations would be required if the algorithm adds some constraint j to the working set at some iteration, only to remove it at a later step.

FINITE TERMINATION OF THE CONVEX QP ALGORITHM

It is not difficult to show that Algorithm 16.1 converges for strictly convex QPs under certain assumptions, that is, it identifies the solution x^* in a finite number of iterations. This claim is certainly true if we assume that the method always takes a nonzero step length α_k whenever the direction p_k computed from (16.27) is nonzero. Our argument proceeds as follows:

- If the solution of (16.27) is $p_k = 0$, the current point x_k is the unique global minimizer of $q(\cdot)$ for the working set $\mathcal{W}_k$; see Theorem 16.5. If it is not the solution of the original problem (16.1) (that is, at least one of the Lagrange multipliers is negative), Theorems 16.4 and 16.5 together show that the step p_{k+1} computed after a constraint is dropped will be a strict decrease direction for $q(\cdot)$. Therefore, because of our assumption $\alpha_k > 0$, we have that the value of q is lower than $q(x_k)$ at all subsequent iterations. It follows that the algorithm can never return to the working set $\mathcal{W}_k$, since

subsequent iterates have values of q that are lower than the global minimizer for this working set.

- The algorithm encounters an iterate k for which $p_k = 0$ solves (16.27) at least on every nth iteration. To show this claim, note that whenever we have an iteration for which $p_k \neq 0$, either we have $\alpha_k = 1$ (in which case we reach the minimizer of q on the current working set $\mathcal{W}_k$, so that the next iteration will yield $p_{k+1} = 0$), or else a constraint is added to the working set $\mathcal{W}_k$. If the latter situation occurs repeatedly, then after at most n iterations the working set will contain n indices, which correspond to n linearly independent vectors. The solution of (16.27) will then be $p_k = 0$, since only the zero vector will satisfy the constraints (16.27b).

- Taken together, the two statements above indicate that the algorithm finds the global minimum of q on its current working set periodically (at least once every n iterations) and that having done so, it never visits this particular working set again. It follows that since there are only a finite number of possible working sets, the algorithm cannot iterate forever. Eventually, it encounters a minimizer for a current working set that satisfies optimality conditions for (16.1), and it terminates with a solution.

The assumption that we can always take a nonzero step along a nonzero descent direction p_k calculated from (16.27) guarantees that the algorithm does not undergo *cycling*. This term refers to the situation in which a sequence of consecutive iterations results in no movement in iterate x, while the working set $\mathcal{W}_k$ undergoes deletions and additions of indices and eventually repeats itself. That is, for some integers k and $l \geq 1$, we have that $x^k = x^{k+l}$ and $\mathcal{W}_k = \mathcal{W}_{k+l}$. At some point in the cycle, a constraint is dropped (as in Theorem 16.4) but a new constraint $i \notin \mathcal{W}_k$ is encountered immediately without any movement along the computed direction p. Procedures for handling degeneracy and cycling in quadratic programming are similar to those for linear programming discussed in Chapter 13; we will not discuss them here. Most QP implementations simply ignore the possibility of cycling.

UPDATING FACTORIZATIONS

We have seen that the step computation in the active-set method given in Algorithm 16.1 requires the solution of the equality-constrained subproblem (16.27). As mentioned at the beginning of this chapter, this computation amounts to solving the KKT system (16.5). Since the working set can change by just one index at every iteration, the KKT matrix differs in at most one row and one column from the previous iteration's KKT matrix. Indeed, G remains fixed, whereas the matrix A of constraint gradients corresponding to the current working set may change through addition or deletion of a single row.

It follows from this observation that we can compute the matrix factors needed to solve (16.27) at the current iteration by updating the factors computed at the previous iteration, rather than recomputing them from scratch. These updating techniques are crucial to the efficiency of active-set methods.

We will limit our discussion to the case in which the step is computed by using the null-space method (16.16)–(16.19). Suppose that A has m linearly independent rows, and assume that the bases Y and Z are defined by means of a QR factorization of A (see Chapter 15 for details). Thus

$$A^T \Pi = Q \begin{bmatrix} R \\ 0 \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} \tag{16.37}$$

(see (15.20)), where Π is a permutation matrix; R is square, upper triangular and nonsingular; $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$ is $n \times n$ orthogonal; and Q_1 and R both have m columns and Q_2 has $n - m$ columns. As noted in Chapter 15, we can choose Z to be simply the orthonormal matrix Q_2.

Suppose that one constraint is added to the working set at the next iteration, so that the new constraint matrix is $\bar{A}^T = [A^T, a]$, where a is a column vector of length n such that $\bar{A}^T$ retains full column rank. As we now show, there is an economical way to update the Q and R factors in (16.37) to obtain new factors (and hence a new null-space basis matrix $\bar{Z}$, with $n - m - 1$ columns) for the expanded matrix $\bar{A}$. Note first that since $Q_1 Q_1^T + Q_2 Q_2^T = I$,

$$\bar{A}^T \begin{bmatrix} \Pi & 0 \\ 0 & 1 \end{bmatrix} = \left[\begin{array}{c|c} A\Pi & a \end{array}\right] = Q \left[\begin{array}{c|c} R & Q_1^T a \\ 0 & Q_2^T a \end{array}\right]. \tag{16.38}$$

We can now define an orthogonal matrix $\hat{Q}$ that transforms the vector $Q_2^T a$ to a vector in which all elements except the first are zero. That is, we have

$$\hat{Q}(Q_2^T a) = \begin{bmatrix} \gamma \\ 0 \end{bmatrix},$$

where γ is a scalar. (Since $\hat{Q}$ is orthogonal, we have $\|Q_2^T a\| = |\gamma|$.) From (16.38) we now have

$$\bar{A}^T \begin{bmatrix} \Pi & 0 \\ 0 & 1 \end{bmatrix} = Q \left[\begin{array}{c|c} R & Q_1^T a \\ \hline 0 & \hat{Q}^T \begin{bmatrix} \gamma \\ 0 \end{bmatrix} \end{array}\right] = Q \begin{bmatrix} I & 0 \\ 0 & \hat{Q}^T \end{bmatrix} \left[\begin{array}{c|c} R & Q_1^T a \\ 0 & \gamma \\ 0 & 0 \end{array}\right].$$

This factorization has the form

$$\bar{A}^T \bar{\Pi} = \bar{Q} \begin{bmatrix} \bar{R} \\ 0 \end{bmatrix},$$

where

$$\bar{\Pi} = \begin{bmatrix} \Pi & 0 \\ 0 & 1 \end{bmatrix}, \quad \bar{Q} = Q \begin{bmatrix} I & 0 \\ 0 & \hat{Q}^T \end{bmatrix} = \begin{bmatrix} Q_1 & Q_2 \hat{Q}^T \end{bmatrix}, \quad \bar{R} = \begin{bmatrix} R & Q_1^T a \\ 0 & \gamma \end{bmatrix}.$$

We can therefore choose $\bar{Z}$ to be the last $n-m-1$ columns of $Q_2\hat{Q}^T$. If we know Z explicitly and need an explicit representation of $\bar{Z}$, we need to account for the cost of obtaining $\hat{Q}$ and the cost of forming the product $Q_2\hat{Q}^T = Z\hat{Q}^T$. Because of the special structure of $\hat{Q}$, this cost is of order $n(n-m)$, compared to the cost of computing (16.37) from scratch, which is of order n^2m. The updating strategy is much less expensive, especially when the null space is small (that is, $n - m \ll n$).

An updating technique can also be designed for the case in which a row is removed from A. This operation has the effect of deleting a column from R in (16.37), which disturbs the upper triangular property of this matrix by introducing a number of nonzeros on the diagonal immediately below the main diagonal of the matrix. Upper triangularity can be restored by applying a sequence of plane rotations. These rotations introduce a number of inexpensive transformations into Q, and the updated null-space matrix is obtained by selecting the last $n-m+1$ columns from this matrix after the transformations are complete. The new null-space basis in this case will have the form

$$\bar{Z} = \begin{bmatrix} Z & \bar{z} \end{bmatrix}, \tag{16.39}$$

that is, the current matrix Z is augmented by a single column. The total cost of this operation varies with the location of the removed column in A, but is in all cases cheaper than recomputing a QR factorization from scratch. For details of these procedures, see Gill et al. [105, Section 5].

Let us now consider the reduced Hessian. Because of the special form of (16.27) we have $c = 0$ in (16.5), and the step p_Y given in (16.17) is zero. Thus from (16.18), the null-space component p_Z is the solution of

$$(Z^T G Z) p_Z = -Z^T g. \tag{16.40}$$

We can sometimes find ways of updating the factorization of the reduced Hessian Z^TGZ after Z has changed. Suppose that we have the Cholesky factorization of the current reduced Hessian, written as

$$Z^T G Z = LL^T,$$

and that at the next step Z changes as in (16.39), gaining a column after deletion of a constraint. A series of plane rotations can then be used to transform the Cholesky factor L into the new factor $\bar{L}$ for the new reduced Hessian $\bar{Z}^T G \bar{Z}$.

A variety of other simplifications are possible. For example, as discussed in Section 16.6, we can update the reduced gradient $Z^T g$ at the same time as we update Z to $\bar{Z}$.

16.5 ACTIVE-SET METHODS FOR INDEFINITE QP

We now consider the case in which the Hessian matrix G has some negative eigenvalues. Algorithm 16.1, the active-set method for convex QP, can be adapted to this indefinite case by modifying the computation of the search direction and step length in certain situations.

To explain the need for the modification, we consider the computation of a step by a null-space method, that is, $p = Zp_Z$, where p_Z is given by (16.40). If the reduced Hessian $Z^T GZ$ is positive definite, then this step p points to the minimizer of the subproblem (16.27), and the logic of the iteration need not be changed. If $Z^T GZ$ has negative eigenvalues, however, p points only to a saddle point of (16.27) and is therefore not a suitable step. Instead, we seek an alternative direction s_Z that is a direction of *negative curvature* for $Z^T GZ$. We then have that

$$q(x + \alpha Z s_Z) \to -\infty \quad \text{as } \alpha \to \infty. \tag{16.41}$$

Additionally, we can choose the sign of s_Z so that Zs_Z is a non–ascent direction for q at the current point x, that is, $\nabla q(x)^T Z s_Z \leq 0$. By moving along the direction Zs_Z, we will encounter a constraint that can then be added to the working set for the next iteration. (If we don't find such a constraint, the problem is unbounded.) If the reduced Hessian for the new working set is not positive definite, we can repeat this process until enough constraints have been added to make the reduced Hessian positive definite. A difficulty with this general approach, however, is that if we allow the reduced Hessian to have several negative eigenvalues, we need to compute its spectral factorization or symmetric indefinite factorization in order to obtain appropriate negative curvature directions, but it is difficult to make these methods efficient when the reduced Hessian changes from one working set to the next.

Inertia controlling methods are a practical class of algorithms for indefinite QP that never allow the reduced Hessian to have more than one negative eigenvalue. As in the convex case, there is a preliminary phase in which a feasible starting point x_0 is found. We place the additional demand on x_0 that it be either a vertex (in which case the reduced Hessian is the null matrix) or a constrained stationary point at which the reduced Hessian is positive definite. (We see below how these conditions can be met.) At each iteration, the algorithm will either add or remove a constraint from the working set. If a constraint is added, the reduced Hessian is of smaller dimension and must remain positive definite or be the null matrix (see the exercises). Therefore, an indefinite reduced Hessian can arise only when one of the constraints is removed from the working set, which happens only when the current

point is a minimizer with respect to the current working set. In this case, we will choose the new search direction to be a direction of negative curvature for the reduced Hessian.

There are various algorithms for indefinite QP that differ in the way that indefiniteness is detected, in the computation of the negative curvature direction, and in the handling of the working set. We now discuss an algorithm that makes use of pseudo-constraints (as proposed by Fletcher [80]) and that computes directions of negative curvature by means of the LDL^T factorization (as proposed by Gill and Murray [107]).

Suppose that the current reduced Hessian $Z^T G Z$ is positive definite and that it is factored as $Z^T G Z = LDL^T$, where L is unit lower triangular and D is diagonal with positive diagonal entries. We denote the number of elements in the current working set $\mathcal{W}$ by t. After removing a constraint from the working set, the new null-space basis can be chosen in the form $Z_+ = [Z \mid z]$ (that is, one additional column), so that the new factors have the form

$$L_+ = \begin{bmatrix} L & 0 \\ l^T & 1 \end{bmatrix}, \qquad D_+ = \begin{bmatrix} D & 0 \\ 0 & d_{n-t+1} \end{bmatrix},$$

for some vector l and element d_{n-t+1}. If we discover that d_{n-t+1} in D is negative, we know that the reduced Hessian is indefinite on the manifold defined by the new working set. We can then compute a direction s_Z of negative curvature for $Z^T G Z$ and a corresponding direction s of negative curvature for G as follows:

$$L_+^T s_Z = e_{n-t+1}, \qquad s = Z_+ s_Z.$$

We can verify that these directions have the desired properties:

$$\begin{aligned} s^T G s &= s_Z^T Z_+^T G Z_+ s_Z \\ &= s_Z^T L_+ D_+ L_+^T s_Z \\ &= e_{n-t+1}^T D_+ e_{n-t+1} \\ &= d_{n-t+1} < 0. \end{aligned}$$

Before moving along this direction s, we will define the working set in a special way. Suppose that i is the index of the constraint that is scheduled to be removed from the working set—the index whose removal causes the reduced Hessian to become indefinite and leads to the negative curvature direction s derived above. Rather then removing i explicitly from the working set, as we do in Algorithm 16.1, we leave it in the working set as a *pseudo-constraint*. By doing so, we ensure that the reduced Hessian for this working set remains positive definite. We now move along the negative curvature direction s until we encounter a constraint, and then continue to take steps in the usual manner, adding constraints until the solution of an equality-constrained subproblem is found. If at this point we can safely delete the pseudo-constraint i from the working set while retaining positive definiteness of

the reduced Hessian, then we do so. If not, we retain it in the working set until a similar opportunity arises on a later iteration.

We illustrate this strategy with a simple example.

ILLUSTRATION

Consider the following indefinite quadratic program in two variables:

$$\min \tfrac{1}{2} x^T \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} x, \tag{16.42a}$$

$$\text{subject to } x_2 \geq 0, \tag{16.42b}$$

$$x_1 + 2x_2 \geq 2, \tag{16.42c}$$

$$-5x_1 + 4x_2 \leq 10, \tag{16.42d}$$

$$x_1 \leq 3. \tag{16.42e}$$

We use superscripts to number the iterates of the algorithm, and use subscripts to denote components of a vector. We choose the initial point $x^1 = (2, 0)$ and define the working set as $\mathcal{W} = \{1, 2\}$, where we have numbered the constraints in the order they appear in (16.42); see Figure 16.4. Since x^1 is a vertex, it is the solution with respect to the

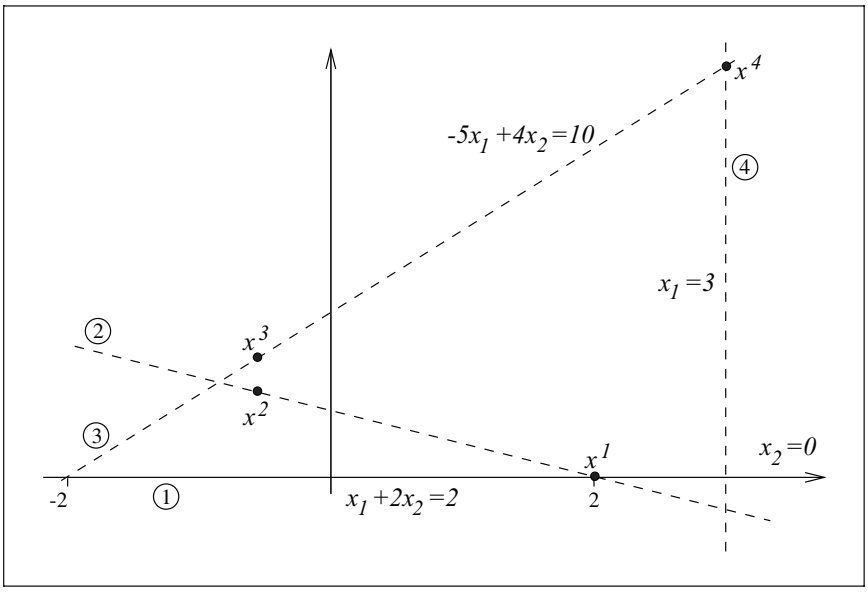

Figure 16.4 Iterates of indefinite QP algorithm.

initial working set $\mathcal{W}$. We compute Lagrange multipliers by solving the system

$$\begin{bmatrix} 0 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix},$$

giving $(\lambda_1, \lambda_2) = (-4, 2)$. We must therefore remove the first constraint from the working set.

The working set is redefined as $\mathcal{W} = \{2\}$, and we can define the null-space basis as $Z = (2, -1)^T$. Since $Z^T G Z = 3$, we have that $L = 1$ and $D = 3$, which is positive. We solve the equality-constrained subproblem for the current working set, and find that the solution is $x^2 = (-\frac{2}{3}, \frac{4}{3})^T$, and $\lambda^2 = -\frac{2}{3}$.

Since this multiplier is negative, we must remove the second constraint from the working set. We then have $\mathcal{W} = \emptyset$ and could define

$$Z = \begin{bmatrix} 2 & 1 \\ -1 & 2 \end{bmatrix},$$

where the second column has been chosen to be perpendicular to the first. To simplify the computations, however, let us define $Z = I$, so that $L = I$ and $D = G$. This D factor has a negative element in the second diagonal element; see (16.42a). We have now encountered the situation where an indefinite QP algorithm must differ from an algorithm for convex QP.

By solving the system $L^T s_z = e_{n-t} = e_2$ we find that $s_z = (0, 1)^T$ and the direction of negative curvature is given by $s = Z s_z = (0, 1)^T$. The direction s is indeed a descent direction, because we have

$$\nabla q(x^2)^T s = \begin{bmatrix} -2/3 \\ -4/3 \end{bmatrix}^T \begin{bmatrix} 0 \\ 1 \end{bmatrix} < 0.$$

(If this inequality were not true, we could simply change the sign of s.) Before moving along this direction, we include the second constraint as a pseudo-constraint, so that the working set is redefined as $\mathcal{W} = \{2\}$.

We now compute the new iterate $x^3 = x^2 + \alpha s = (-\frac{2}{3}, \frac{5}{3})$, where $\alpha = \frac{1}{3}$ is the step length to the newly encountered third constraint. We add this constraint to the working set, which is now $\mathcal{W} = \{2, 3\}$, and update the LDL^T factorization as we change the working set from $\{2\}$ to $\{2, 3\}$. Since the current iterate is a vertex, it is the solution with respect to the current working set. We also find that both L and D are null matrices.

We will now try to remove the pseudo-constraint. We attempt to solve the equality-constrained subproblem with respect to the working set $\{3\}$, but find that $Z = [4, 5]^T$ and $Z^T G Z = -9$, so that $L = 1$ and $D = -9$. In other words, removal of the pseudo-constraint would yield a reduced Hessian with negative eigenvalues, so we must retain the second

constraint as a pseudo-constraint for the time being. However, we can use the hypothetical deletion to calculate a negative-curvature search direction, to move us away from the current iterate. By solving $L^T s_z = e_{2-1} = e_1$ we find that $s_z = 1$, and so the desired direction is $s = Zs_z = (4, 5)^T$. It is easy to verify that s is a descent direction.

We now compute the new iterate $x^4 = x^3 + \alpha s = (3, 25/4)^T$, and find that the step length $\alpha = 11/12$ leads us to the fourth constraint. By adding this constraint to the working set, we obtain $\mathcal{W} = \{2, 3, 4\}$. It is now possible to *drop the pseudo-constraint* 2, since by doing so the working set would become $\{3, 4\}$ and the reduced Hessian is the null matrix, which has no negative eigenvalues. In other words, x^4 solves the subproblem with respect to the working set $\{3, 4\}$.

Continuing with the iteration, we compute Lagrange multipliers at x^4 by solving the following system:

$$\begin{bmatrix} 3 \\ -25/4 \end{bmatrix} = \begin{bmatrix} 5 & -1 \\ -4 & 0 \end{bmatrix} \begin{bmatrix} \lambda_3 \\ \lambda_4 \end{bmatrix}.$$

We obtain $(\lambda_3, \lambda_4) = (25/16, 77/16)$, and we conclude that x^4 is a local solution of the quadratic program (16.42). (In fact, it is also a global solution).

One drawback of this approach is that by including more constraints in the working set, we increase the possibility of degeneracy.

CHOICE OF STARTING POINT

We mentioned above that the initial point must be chosen so as to be a vertex, or at least a point at which the reduced-Hessian matrix is positive definite. To ensure that this property holds, we may need to introduce *artificial constraints* that are active at the initial point and are linearly independent with respect to the other constraints in the working set. For example, consider the quadratic program

$$\begin{aligned} \min \quad & -x_1 x_2 \\ \text{subject to} \; & -1 \le x_1 \le 1, \\ & -1 \le x_2 \le 1. \end{aligned}$$

If the starting point is $(0, 0)^T$, no constraints are active, and the reduced Hessian is indefinite for the working set $\mathcal{W}^0 = \emptyset$. By introducing the artificial constraints $x_1 = 0$ and $x_1 - x_2 = 0$, the initial point has the desired properties, so we can start the active-set procedure described above. Once the minimizer for the current working set has been computed, the sign of each artificial constraint normal is chosen so that its multiplier is nonpositive and the constraint may be deleted. Temporary constraints are usually deleted first, if there is a choice.

FAILURE OF THE ACTIVE-SET METHOD

When G is not positive definite, the inertia controlling algorithm just described is not guaranteed to find a local minimizer of the quadratic program. In fact, there may exist nonoptimal points at which *any* active-set method will find it difficult to proceed. These points satisfy the first-order optimality conditions (i.e., they are stationary points), and their reduced Hessian is positive definite, but at least one of the Lagrange multipliers corresponding to the inequality constraints is zero. At such a point, it is not possible to compute a direction that improves the objective function by deleting *only one* constraint at a time. We may have to delete two or more constraints at once in order to make progress.

This phenomenon is illustrated by the problem

$$\min\ -x_1x_2 \quad \text{subject to } 0 \le x_1 \le 1,\ 0 \le x_2 \le 1.$$

Suppose that the starting point is $(0, 0)$ and that both active constraints are included in the initial working set. It is easy to verify that this point is a stationary point, and it is certainly a minimizer on the given working set by virtue of being a vertex. However, if we delete either constraint from the working set, the new reduced Hessian is singular. No feasible direction of negative curvature can therefore be computed by removing only one constraint, so the active-set method will terminate at this point, which is not a local solution of the quadratic program. We can decrease the objective by moving along the direction $(1, 1)$, for instance.

Even though this type of failure is possible, it is not very common, and rounding errors tend to move the algorithm away from such difficult points. Various devices have been proposed to decrease the probability of failure, but none of them can guarantee that a solution will be found in all cases. In the example above, we could move x_1 from zero to 1 while holding x_2 fixed at zero, and this would not affect the value of the objective q. Next we could move x_2 from zero to 1 to obtain the optimal solution.

In general, the cause of this difficulty is that first-order optimality does not lead to second-order optimality for active-set methods.

DETECTING INDEFINITENESS USING THE LBL^T FACTORIZATION

We can also use the symmetric indefinite factorization algorithm (16.12) to determine whether the reduced Hessian Z^TBZ is positive definite. Since this factorization is useful in other optimization methods, such as interior-point methods and sequential quadratic programming methods, we present some of its fundamental properties.

Recall that the inertia of a symmetric matrix K is defined as the triplet formed by the number of positive, negative, and zero eigenvalues of K. We write

$$\text{inertia}(K) = (n_+, n_-, n_0).$$

Sylvester's law of inertia states that if C is a nonsingular matrix, then inertia$(C^T K C) =$ inertia(K); see, for example, [115]. The following result follows from the repeated application of Sylvester's law.

Theorem 16.6.
Let K be defined by (16.7) and suppose that A has rank m. Then

$$\text{inertia}(K) = \text{inertia}(Z^T G Z) + (m, m, 0).$$

This theorem implies that if $Z^T G Z$ is positive definite, then the inertia of the KKT matrix is $(n, m, 0)$, as stated in Lemma 16.3. Note that Theorem 16.6 also shows that K has at least m positive eigenvalues, even if G is negative definite.

Let us suppose that we compute the symmetric indefinite factorization (16.12) of the KKT matrix K. To simplify the discussion, we assume that the permutation matrix P is the identity, and write

$$K = LBL^T.$$

Sylvester's theorem implies that B and K have the same inertia. Moreover, since the 2×2 blocks in B are normally constructed so as to have one positive and one negative eigenvalue, we can compute the inertia of K by examining the blocks of B. In particular, the number of positive eigenvalues of K equals the number of 2×2 blocks plus the number of positive 1×1 blocks.

The symmetric indefinite factorization can therefore be used in inertia controlling methods for indefinite quadratic programming to control the logic of the algorithm.

16.6 THE GRADIENT--PROJECTION METHOD

In the classical active-set method described in the previous two sections, the active set and working set change slowly, usually by a single index at each iteration. As a result, this method may require many iterations to converge on large-scale problems. For instance, if the starting point x^0 has no active constraints, while 200 constraints are active at the solution, then at least 200 iterations of the active-set method will be required to reach the solution.

The gradient projection method is designed to make rapid changes to the active set. It is most efficient when the constraints are simple in form—in particular, when there are only bounds on the variables. Accordingly, we will restrict our attention to the following bound-constrained problem:

$$\min_x \; q(x) = \tfrac{1}{2} x^T G x + x^T d \tag{16.43a}$$

$$\text{subject to} l \le x \le u, \tag{16.43b}$$