

CHAPTER *16*

Quadratic Programming

An optimization problem with a quadratic objective function and linear constraints is called a quadratic program. Problems of this type are important in their own right, and they also arise as subproblems in methods for general constrained optimization, such as sequential quadratic programming (Chapter 18), augmented Lagrangian methods (Chapter 17), and interior-point methods (Chapter 19).

The general quadratic program (QP) can be stated as

$$\min_x \quad q(x) = \frac{1}{2}x^T Gx + x^T c \quad (16.1a)$$

$$\text{subject to} \quad a_i^T x = b_i, \quad i \in \mathcal{E}, \quad (16.1b)$$

$$a_i^T x \geq b_i, \quad i \in \mathcal{I}, \quad (16.1c)$$

where G is a symmetric $n \times n$ matrix, \mathcal{E} and \mathcal{I} are finite sets of indices, and c , x , and $\{a_i\}$, $i \in \mathcal{E} \cup \mathcal{I}$, are vectors in \mathbb{R}^n . Quadratic programs can always be solved (or shown to be infeasible) in a finite amount of computation, but the effort required to find a solution depends strongly on the characteristics of the objective function and the number of inequality constraints. If the Hessian matrix G is positive semidefinite, we say that (16.1) is a *convex QP*, and in this case the problem is often similar in difficulty to a linear program. (*Strictly convex QPs* are those in which G is positive definite.) *Nonconvex QPs*, in which G is an indefinite matrix, can be more challenging because they can have several stationary points and local minima.

In this chapter we focus primarily on convex quadratic programs. We start by considering an interesting application of quadratic programming.

□ **EXAMPLE 16.1** (PORTFOLIO OPTIMIZATION)

Every investor knows that there is a tradeoff between risk and return: To increase the expected return on investment, an investor must be willing to tolerate greater risks. Portfolio theory studies how to model this tradeoff given a collection of n possible investments with returns r_i , $i = 1, 2, \dots, n$. The returns r_i are usually not known in advance and are often assumed to be random variables that follow a normal distribution. We can characterize these variables by their expected value $\mu_i = E[r_i]$ and their variance $\sigma_i^2 = E[(r_i - \mu_i)^2]$. The variance measures the fluctuations of the variable r_i about its mean, so that larger values of σ_i indicate riskier investments. The returns are not in general independent, and we can define correlations between pairs of returns as follows:

$$\rho_{ij} = \frac{E[(r_i - \mu_i)(r_j - \mu_j)]}{\sigma_i \sigma_j}, \quad \text{for } i, j = 1, 2, \dots, n.$$

The correlation measures the tendency of the return on investments i and j to move in the same direction. Two investments whose returns tend to rise and fall together have a positive correlation; the nearer ρ_{ij} is to 1, the more closely the two investments track each other. Investments whose returns tend to move in opposite directions have a negative correlation.

An investor constructs a portfolio by putting a fraction x_i of the available funds into investment i , for $i = 1, 2, \dots, n$. Assuming that all available funds are invested and that short-selling is not allowed, the constraints are $\sum_{i=1}^n x_i = 1$ and $x \geq 0$. The return on the

portfolio is given by

$$R = \sum_{i=1}^n x_i r_i. \quad (16.2)$$

To measure the desirability of the portfolio, we need to obtain measures of its expected return and variance. The expected return is simply

$$E[R] = E \left[\sum_{i=1}^n x_i r_i \right] = \sum_{i=1}^n x_i E[r_i] = x^T \mu,$$

while the variance is given by

$$\text{Var}[R] = E[(R - E[R])^2] = \sum_{i=1}^n \sum_{j=1}^n x_i x_j \sigma_i \sigma_j \rho_{ij} = x^T G x,$$

where the $n \times n$ symmetric positive semidefinite matrix G defined by

$$G_{ij} = \rho_{ij} \sigma_i \sigma_j$$

is called the *covariance matrix*.

Ideally, we would like to find a portfolio for which the expected return $x^T \mu$ is large while the variance $x^T G x$ is small. In the model proposed by Markowitz [201], we combine these two aims into a single objective function with the aid of a “risk tolerance parameter” denoted by κ , and we solve the following problem to find the optimal portfolio:

$$\max x^T \mu - \kappa x^T G x, \quad \text{subject to} \quad \sum_{i=1}^n x_i = 1, \quad x \geq 0.$$

The value chosen for the nonnegative parameter κ depends on the preferences of the individual investor. Conservative investors, who place more emphasis on minimizing risk in their portfolio, would choose a large value of κ to increase the weight of the variance measure in the objective function. More daring investors, who are prepared to take on more risk in the hope of a higher expected return, would choose a smaller value of κ .

The difficulty in applying this portfolio optimization technique to real-life investing lies in defining the expected returns, variances, and correlations for the investments in question. Financial professionals often combine historical data with their own insights and expectations to produce values of these quantities. □

16.1 EQUALITY-CONSTRAINED QUADRATIC PROGRAMS

We begin our discussion of algorithms for quadratic programming by considering the case in which only equality constraints are present. Techniques for this special case are applicable also to problems with inequality constraints since, as we see later in this chapter, some algorithms for general QP require the solution of an equality-constrained QP at each iteration.

PROPERTIES OF EQUALITY-CONSTRAINED QPs

For simplicity, we write the equality constraints in matrix form and state the equality-constrained QP as follows:

$$\min_x q(x) \stackrel{\text{def}}{=} \frac{1}{2}x^T Gx + x^T c \quad (16.3a)$$

$$\text{subject to } Ax = b, \quad (16.3b)$$

where A is the $m \times n$ Jacobian of constraints (with $m \leq n$) whose rows are a_i^T , $i \in \mathcal{E}$ and b is the vector in \mathbb{R}^m whose components are b_i , $i \in \mathcal{E}$. For the present, we assume that A has full row rank (rank m) so that the constraints (16.3b) are consistent. (In Section 16.8 we discuss the case in which A is rank deficient.)

The first-order necessary conditions for x^* to be a solution of (16.3) state that there is a vector λ^* such that the following system of equations is satisfied:

$$\begin{bmatrix} G & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x^* \\ \lambda^* \end{bmatrix} = \begin{bmatrix} -c \\ b \end{bmatrix}. \quad (16.4)$$

These conditions are a consequence of the general result for first-order optimality conditions, Theorem 12.1. As in Chapter 12, we call λ^* the vector of Lagrange multipliers. The system (16.4) can be rewritten in a form that is useful for computation by expressing x^* as $x^* = x + p$, where x is some estimate of the solution and p is the desired step. By introducing this notation and rearranging the equations, we obtain

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = \begin{bmatrix} g \\ h \end{bmatrix}, \quad (16.5)$$

where

$$h = Ax - b, \quad g = c + Gx, \quad p = x^* - x. \quad (16.6)$$

The matrix in (16.5) is called the Karush–Kuhn–Tucker (KKT) matrix, and the following result gives conditions under which it is nonsingular. As in Chapter 15, we use Z to

denote the $n \times (n - m)$ matrix whose columns are a basis for the null space of A . That is, Z has full rank and satisfies $AZ = 0$.

Lemma 16.1.

Let A have full row rank, and assume that the reduced-Hessian matrix $Z^T G Z$ is positive definite. Then the KKT matrix

$$K = \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \quad (16.7)$$

is nonsingular, and hence there is a unique vector pair (x^, λ^*) satisfying (16.4).*

PROOF. Suppose there are vectors w and v such that

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = 0. \quad (16.8)$$

Since $Aw = 0$, we have from (16.8) that

$$0 = \begin{bmatrix} w \\ v \end{bmatrix}^T \begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = w^T G w.$$

Since w lies in the null space of A , it can be written as $w = Zu$ for some vector $u \in \mathbb{R}^{n-m}$. Therefore, we have

$$0 = w^T G w = u^T Z^T G Z u,$$

which by positive definiteness of $Z^T G Z$ implies that $u = 0$. Therefore, $w = 0$, and by (16.8), $A^T v = 0$. Full row rank of A then implies that $v = 0$. We conclude that equation (16.8) is satisfied only if $w = 0$ and $v = 0$, so the matrix is nonsingular, as claimed. \square

\square **EXAMPLE 16.2**

Consider the quadratic programming problem

$$\begin{aligned} \min q(x) &= 3x_1^2 + 2x_1x_2 + x_1x_3 + 2.5x_2^2 + 2x_2x_3 + 2x_3^2 - 8x_1 - 3x_2 - 3x_3, \\ \text{subject to} \quad &x_1 + x_3 = 3, \quad x_2 + x_3 = 0. \end{aligned} \quad (16.9)$$

We can write this problem in the form (16.3) by defining

$$G = \begin{bmatrix} 6 & 2 & 1 \\ 2 & 5 & 2 \\ 1 & 2 & 4 \end{bmatrix}, \quad c = \begin{bmatrix} -8 \\ -3 \\ -3 \end{bmatrix}, \quad A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 3 \\ 0 \end{bmatrix}.$$

The solution x^* and optimal Lagrange multiplier vector λ^* are given by

$$x^* = (2, -1, 1)^T, \quad \lambda^* = (3, -2)^T.$$

In this example, the matrix G is positive definite, and the null-space basis matrix can be defined as in (15.15), giving

$$Z = (-1, -1, 1)^T. \quad (16.10)$$

□

We have seen that when the conditions of Lemma 16.1 are satisfied, there is a unique vector pair (x^*, λ^*) that satisfies the first-order necessary conditions for (16.3). In fact, the second-order sufficient conditions (see Theorem 12.6) are also satisfied at (x^*, λ^*) , so x^* is a strict local minimizer of (16.3). In fact, we can use a direct argument to show that x^* is a *global* solution of (16.3).

Theorem 16.2.

Let A have full row rank and assume that the reduced-Hessian matrix $Z^T G Z$ is positive definite. Then the vector x^ satisfying (16.4) is the unique global solution of (16.3).*

PROOF. Let x be any other feasible point (satisfying $Ax = b$), and as before, let p denote the difference $x^* - x$. Since $Ax^* = Ax = b$, we have that $Ap = 0$. By substituting into the objective function (16.3a), we obtain

$$\begin{aligned} q(x) &= \frac{1}{2}(x^* - p)^T G(x^* - p) + c^T(x^* - p) \\ &= \frac{1}{2}p^T Gp - p^T Gx^* - c^T p + q(x^*). \end{aligned} \quad (16.11)$$

From (16.4) we have that $Gx^* = -c + A^T \lambda^*$, so from $Ap = 0$ we have that

$$p^T Gx^* = p^T(-c + A^T \lambda^*) = -p^T c.$$

By substituting this relation into (16.11), we obtain

$$q(x) = \frac{1}{2}p^T Gp + q(x^*).$$

Since p lies in the null space of A , we can write $p = Zu$ for some vector $u \in \mathbb{R}^{n-m}$, so that

$$q(x) = \frac{1}{2}u^T Z^T G Z u + q(x^*).$$

By positive definiteness of $Z^T G Z$, we conclude that $q(x) > q(x^*)$ except when $u = 0$, that is, when $x = x^*$. Therefore, x^* is the unique global solution of (16.3). \square

When the reduced Hessian matrix $Z^T G Z$ is positive semidefinite with zero eigenvalues, the vector x^* satisfying (16.4) is a local minimizer but not a strict local minimizer. If the reduced Hessian has negative eigenvalues, then x^* is only a stationary point, not a local minimizer.

16.2 DIRECT SOLUTION OF THE KKT SYSTEM

In this section we discuss efficient methods for solving the KKT system (16.5). The first important observation is that if $m \geq 1$, the KKT matrix is always indefinite. We define the inertia of a symmetric matrix K to be the scalar triple that indicates the numbers n_+ , n_- , and n_0 of positive, negative, and zero eigenvalues, respectively, that is,

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

The following result characterizes the inertia of the KKT matrix.

Theorem 16.3.

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

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

Therefore, if $Z^T G Z$ is positive definite, $\text{inertia}(K) = (n, m, 0)$.

The proof of this result is given in [111], for example. Note that the assumptions of this theorem are satisfied by Example 16.2. Hence, if we construct the 5×5 matrix K using the data of this example, we obtain $\text{inertia}(K) = (3, 2, 0)$.

Knowing that the KKT system is indefinite, we now describe the main direct techniques used to solve (16.5).

FACTORING THE FULL KKT SYSTEM

One option for solving (16.5) is to perform a triangular factorization on the full KKT matrix and then perform backward and forward substitution with the triangular factors. Because of indefiniteness, we cannot use the Cholesky factorization. We could use Gaussian

elimination with partial pivoting (or a sparse variant thereof) to obtain the L and U factors, but this approach has the disadvantage that it ignores the symmetry.

The most effective strategy in this case is to use a *symmetric indefinite factorization*, which we have discussed in Chapter 3 and the Appendix. For a general symmetric matrix K , this factorization has the form

$$P^T K P = L B L^T, \quad (16.12)$$

where P is a permutation matrix, L is unit lower triangular, and B is block-diagonal with either 1×1 or 2×2 blocks. The symmetric permutations defined by the matrix P are introduced for numerical stability of the computation and, in the case of large sparse K , for maintaining sparsity. The computational cost of the symmetric indefinite factorization (16.12) is typically about half the cost of sparse Gaussian elimination.

To solve (16.5), we first compute the factorization (16.12) of the coefficient matrix. We then perform the following sequence of operations to arrive at the solution:

$$\begin{aligned} & \text{solve } Lz = P^T \begin{bmatrix} g \\ h \end{bmatrix} && \text{to obtain } z; \\ & \text{solve } B\hat{z} = z && \text{to obtain } \hat{z}; \\ & \text{solve } L^T \bar{z} = \hat{z} && \text{to obtain } \bar{z}; \\ & \text{set } \begin{bmatrix} -p \\ \lambda^* \end{bmatrix} = P\bar{z}. \end{aligned}$$

Since multiplications with the permutation matrices P and P^T can be performed by simply rearranging vector components, they are inexpensive. Solution of the system $B\hat{z} = z$ entails solving a number of small 1×1 and 2×2 systems, so the number of operations is a small multiple of the system dimension ($m + n$), again inexpensive. Triangular substitutions with L and L^T are more costly. Their precise cost depends on the amount of sparsity, but is usually significantly less than the cost of performing the factorization (16.12).

This approach of factoring the full $(n + m) \times (n + m)$ KKT matrix (16.7) is quite effective on many problems. It may be expensive, however, when the heuristics for choosing the permutation matrix P are not able to maintain sparsity in the L factor, so that L becomes much more dense than the original coefficient matrix.

SCHUR-COMPLEMENT METHOD

Assuming that G is positive definite, we can multiply the first equation in (16.5) by AG^{-1} and then subtract the second equation to obtain a linear system in the vector λ^* alone:

$$(AG^{-1}A^T)\lambda^* = (AG^{-1}g - h). \quad (16.13)$$

We solve this symmetric positive definite system for λ^* and then recover p from the first equation in (16.5) by solving

$$Gp = A^T \lambda^* - g. \quad (16.14)$$

This approach requires us to perform operations with G^{-1} , as well as to compute the factorization of the $m \times m$ matrix $AG^{-1}A^T$. Therefore, it is most useful when:

- G is well conditioned and easy to invert (for instance, when G is diagonal or block-diagonal); or
- G^{-1} is known explicitly through a quasi-Newton updating formula; or
- the number of equality constraints m is small, so that the number of backsolves needed to form the matrix $AG^{-1}A^T$ is not too large.

The name ‘‘Schur-Complement method’’ derives from the fact that, by applying block Gaussian elimination to (16.7) using G as the pivot, we obtain the block upper triangular system

$$\begin{bmatrix} G & A^T \\ 0 & -AG^{-1}A^T \end{bmatrix}. \quad (16.15)$$

In linear algebra terminology, the matrix $AG^{-1}A^T$ is the Schur complement of G in the matrix K of (16.7). By applying this block elimination technique to the system (16.5), and performing a block backsolve, we obtain (16.13), (16.14).

We can use an approach like the Schur-complement method to derive an explicit inverse formula for the KKT matrix in (16.5). This formula is

$$\begin{bmatrix} G & A^T \\ A & 0 \end{bmatrix}^{-1} = \begin{bmatrix} C & E \\ E^T & F \end{bmatrix}, \quad (16.16)$$

with

$$\begin{aligned} C &= G^{-1} - G^{-1}A^T(AG^{-1}A^T)^{-1}AG^{-1}, \\ E &= G^{-1}A^T(AG^{-1}A^T)^{-1}, \\ F &= -(AG^{-1}A^T)^{-1}. \end{aligned}$$

The solution of (16.5) can be obtained by multiplying its right-hand side by this inverse matrix. If we take advantage of common expressions, and group the terms appropriately, we recover the approach (16.13), (16.14).

NULL-SPACE METHOD

The null-space method does not require nonsingularity of G and therefore has wider applicability than the Schur-complement method. It assumes only that the conditions of Lemma 16.1 hold, namely, that A has full row rank and that $Z^T G Z$ is positive definite. However, it requires knowledge of the null-space basis matrix Z . Like the Schur-complement method, it exploits the block structure in the KKT system to decouple (16.5) into two smaller systems.

Suppose that we partition the vector p in (16.5) into two components, as follows:

$$p = Yp_y + Zp_z, \quad (16.17)$$

where Z is the $n \times (n - m)$ null-space matrix, Y is any $n \times m$ matrix such that $[Y \mid Z]$ is nonsingular, p_y is an m -vector, and p_z is an $(n - m)$ -vector. The matrices Y and Z were discussed in Section 15.3, where Figure 15.4 shows that Yx_y is a particular solution of $Ax = b$, while Zx_z is a displacement along these constraints.

By substituting p into the second equation of (16.5) and recalling that $AZ = 0$, we obtain

$$(AY)p_y = -h. \quad (16.18)$$

Since A has rank m and $[Y \mid Z]$ is $n \times n$ nonsingular, the product $A[Y \mid Z] = [AY \mid 0]$ has rank m . Therefore, AY is a nonsingular $m \times m$ matrix, and p_y is well determined by the equations (16.18). Meanwhile, we can substitute (16.17) into the first equation of (16.5) to obtain

$$-GYp_y - GP_z + A^T \lambda^* = g$$

and multiply by Z^T to obtain

$$(Z^T G Z)p_z = -Z^T G Y p_y - Z^T g. \quad (16.19)$$

This system can be solved by performing a Cholesky factorization of the reduced-Hessian matrix $Z^T G Z$ to determine p_z . We therefore can compute the total step $p = Yp_y + Zp_z$. To obtain the Lagrange multiplier, we multiply the first block row in (16.5) by Y^T to obtain the linear system

$$(AY)^T \lambda^* = Y^T (g + Gp), \quad (16.20)$$

which can be solved for λ^* .

□ EXAMPLE 16.3

Consider the problem (16.9) given in Example 16.2. We can choose

$$Y = \begin{bmatrix} 2/3 & -1/3 \\ -1/3 & 2/3 \\ 1/3 & 1/3 \end{bmatrix}$$

and set Z as in (16.10). Note that $AY = I$.

Suppose we have $x = (0, 0, 0)^T$ in (16.6). Then

$$h = Ax - b = -b, \quad g = c + Gx = c = \begin{bmatrix} -8 \\ -3 \\ -3 \end{bmatrix}.$$

Simple calculation shows that

$$p_Y = \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \quad p_Z = [0],$$

so that

$$p = x^* - x = Yp_Y + Zp_Z = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}.$$

After recovering λ^* from (16.20), we conclude that

$$x^* = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}, \quad \lambda^* = \begin{bmatrix} 3 \\ -2 \end{bmatrix}.$$

□

The null-space approach can be very effective when the number of degrees of freedom $n - m$ is small. Its main limitation lies in the need for the null-space matrix Z which, as we have seen in Chapter 15, can be expensive to compute in some large problems. The matrix Z is not uniquely defined and, if it is poorly chosen, the reduced system (16.19) may become ill conditioned. If we choose Z to have orthonormal columns, as is normally done in software for small and medium-sized problems, then the conditioning of $Z^T G Z$ is at least as good as that of G itself. When A is large and sparse, however, an orthonormal Z is expensive to

compute, so for practical reasons we are often forced to use one of the less reliable choices of Z described in Chapter 15.

It is difficult to give hard and fast rules about the relative effectiveness of null-space and Schur-complement methods, because factors such as fill-in during computation of Z vary significantly even among problems of the same dimension. In general, we can recommend the Schur-complement method if G is positive definite and $AG^{-1}A^T$ can be computed relatively cheaply (because G is easy to invert or because m is small relative to n). Otherwise, the null-space method is often preferable, in particular when it is much more expensive to compute factors of G than to compute the null-space matrix Z and the factors of $Z^T G Z$.

16.3 ITERATIVE SOLUTION OF THE KKT SYSTEM

An alternative to the direct factorization techniques discussed in the previous section is to use an iterative method to solve the KKT system (16.5). Iterative methods are suitable for solving very large systems and often lend themselves well to parallelization. The conjugate gradient (CG) method is not recommended for solving the full system (16.5) as written, because it can be unstable on systems that are not positive definite. Better options are Krylov methods for general linear or symmetric indefinite systems. Candidates include the GMRES, QMR, and LSQR methods; see the Notes and References at the end of the chapter. Other iterative methods can be derived from the null-space approach by applying the conjugate gradient method to the reduced system (16.19). Methods of this type are key to the algorithms of Chapters 18 and 19, and are discussed in the remainder of this section. We assume throughout that $Z^T G Z$ is positive definite.

CG APPLIED TO THE REDUCED SYSTEM

We begin our discussion of iterative null-space methods by deriving the underlying equations in the notation of the equality-constrained QP (16.3). Expressing the solution of the quadratic program (16.3) as

$$x^* = Yx_y + Zx_z, \quad (16.21)$$

for some vectors $x_z \in \mathbb{R}^{n-m}$, $x_y \in \mathbb{R}^m$, the constraints $Ax = b$ yield

$$AYx_y = b, \quad (16.22)$$

which determines the vector x_y . In Chapter 15, various practical choices of Y are described, some of which allow (16.22) to be solved economically. Substituting (16.21) into (16.3), we see that x_z solves the unconstrained reduced problem

$$\min_{x_z} \frac{1}{2}x_z^T Z^T G Z x_z + x_z^T c_z,$$

where

$$c_z = Z^T G Y x_y + Z^T c. \quad (16.23)$$

The solution x_z satisfies the linear system

$$Z^T G Z x_z = -c_z. \quad (16.24)$$

Since $Z^T G Z$ is positive definite, we can apply the CG method to this linear system and substitute x_z into (16.21) to obtain a solution of (16.3).

As discussed in Chapter 5, preconditioning can improve the rate of convergence of the CG iteration, so we assume that a preconditioner W_{zz} is given. The preconditioned CG method (Algorithm 5.3) applied to the $(n - m)$ -dimensional reduced system (16.24) is as follows. (We denote the steps produced by the CG iteration by d_z .)

Algorithm 16.1 (Preconditioned CG for Reduced Systems).

Choose an initial point x_z ;

Compute $r_z = Z^T G Z x_z + c_z$, $g_z = W_{zz}^{-1} r_z$, and $d_z = -g_z$;

repeat

$$\alpha \leftarrow r_z^T g_z / d_z^T Z^T G Z d_z; \quad (16.25a)$$

$$x_z \leftarrow x_z + \alpha d_z; \quad (16.25b)$$

$$r_z^+ \leftarrow r_z + \alpha Z^T G Z d_z; \quad (16.25c)$$

$$g_z^+ \leftarrow W_{zz}^{-1} r_z^+; \quad (16.25d)$$

$$\beta \leftarrow (r_z^+)^T g_z^+ / r_z^T g_z; \quad (16.25e)$$

$$d_z \leftarrow -g_z^+ + \beta d_z; \quad (16.25f)$$

$$g_z \leftarrow g_z^+; \quad r_z \leftarrow r_z^+; \quad (16.25g)$$

until a termination test is satisfied.

This iteration may be terminated when, for example, $r_z^T W_{zz}^{-1} r_z$ is sufficiently small.

In this approach, it is not necessary to form the reduced Hessian $Z^T G Z$ explicitly because the CG method requires only that we compute matrix-vector products involving this matrix. In fact, it is not even necessary to form Z explicitly as long as we are able to compute products of Z and Z^T with arbitrary vectors. For some choices of Z , these products are much cheaper to compute than Z itself, as we have seen in Chapter 15.

The preconditioner W_{zz} is a symmetric, positive definite matrix of dimension $n - m$, which might be chosen to cluster the eigenvalues of $W_{zz}^{-1/2} (Z^T G Z) W_{zz}^{-1/2}$ and to reduce the span between the smallest and largest eigenvalues. An ideal choice of preconditioner is one for which $W_{zz}^{-1/2} (Z^T G Z) W_{zz}^{-1/2} = I$, that is, $W_{zz} = Z^T G Z$. Motivated by this ideal, we consider preconditioners of the form

$$W_{zz} = Z^T H Z, \quad (16.26)$$

where H is a symmetric matrix such that $Z^T H Z$ is positive definite. Some choices of H are discussed below. Preconditioners of the form (16.26) allow us to apply the CG method in n -dimensional space, as we discuss next.

THE PROJECTED CG METHOD

It is possible to design a modification of the Algorithm 16.1 that avoids operating with the null-space basis Z , provided we use a preconditioner of the form (16.26) and a particular solution of the equation $Ax = b$. This approach works implicitly with an orthogonal matrix Z and is not affected by ill conditioning in A or by a poor choice of Z .

After the solution x_z of (16.24) has been computed by using Algorithm 16.1, it must be multiplied by Z and substituted in (16.21) to give the solution of the quadratic program (16.3). Alternatively, we may rewrite Algorithm 16.1 to work directly with the vector $x = Zx_z + Yx_y$, where the Yx_y term is fixed at the start and the x_z term is updated (implicitly) within each iteration. To specify this form of the CG algorithm, we introduce the n -vectors x, r, g , and d , which satisfy $x = Zx_z + Yx_y$, $Z^T r = r_z$, $g = Zg_z$, and $d = Zd_z$, respectively. We also define the scaled $n \times n$ projection matrix P as follows:

$$P = Z(Z^T H Z)^{-1} Z^T, \quad (16.27)$$

where H is the preconditioning matrix from (16.26). The CG iteration in n -dimensional space can be specified as follows.

Algorithm 16.2 (Projected CG Method).

Choose an initial point x satisfying $Ax = b$;
 Compute $r = Gx + c$, $g = Pr$, and $d = -g$;
repeat

$$\alpha \leftarrow r^T g / d^T G d; \quad (16.28a)$$

$$x \leftarrow x + \alpha d; \quad (16.28b)$$

$$r^+ \leftarrow r + \alpha G d; \quad (16.28c)$$

$$g^+ \leftarrow P r^+; \quad (16.28d)$$

$$\beta \leftarrow (r^+)^T g^+ / r^T g; \quad (16.28e)$$

$$d \leftarrow -g^+ + \beta d; \quad (16.28f)$$

$$g \leftarrow g^+; \quad r \leftarrow r^+; \quad (16.28g)$$

until a convergence test is satisfied.

A practical stop test is to terminate when $r^T g = r^T P r$ is smaller than a prescribed tolerance.

Note that the vector g^+ , which we call the *preconditioned residual*, has been defined to be in the null space of A . As a result, in exact arithmetic, all the search directions d generated by Algorithm 16.2 also lie in the null space of A , and thus the iterates x all satisfy $Ax = b$. It is not difficult to verify (see Exercise 16.14) that the iteration is well defined if $Z^T G Z$ and $Z^T H Z$ are positive definite. The reader can also verify that the iterates x generated by Algorithm 16.2 are related to the iterates x_z of Algorithm 16.1 via (16.21).

Two simple choices of the preconditioning matrix H are $H = \text{diag}(|G_{ii}|)$ and $H = I$. In some applications, it is effective to define H as a block diagonal submatrix of G .

Algorithm 16.2 makes use of the null-space basis Z only through the operator (16.27). It is possible, however, to compute $P r$ without knowing a representation of the null-space basis Z . For simplicity, we first consider the case in which $H = I$, so that P is the orthogonal projection operator onto the null space of A . We use P_1 to denote this special case of P , that is,

$$P_1 = Z(Z^T Z)^{-1} Z^T. \quad (16.29)$$

The computation of the preconditioned residual $g^+ = P_1 r^+$ in (16.28d) can be performed in two ways. The first is to express P_1 by the equivalent formula

$$P_1 = I - A^T (A A^T)^{-1} A \quad (16.30)$$

and thus compute $g^+ = P_1 r^+$. We can then write $g^+ = r^+ - A^T v^+$, where v^+ is the solution of the system

$$A A^T v^+ = A r^+. \quad (16.31)$$

This approach for computing the projection $g^+ = P_1 r^+$ is called the *normal equations approach*; the system (16.31) can be solved by using a Cholesky factorization of $A A^T$.

The second approach is to express the projection (16.28d) as the solution of the augmented system

$$\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} g^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} r^+ \\ 0 \end{bmatrix}, \quad (16.32)$$

which can be solved by means of a symmetric indefinite factorization, as discussed earlier. We call this approach the *augmented system approach*.

We suppose now that the preconditioning has the general form of (16.27) and (16.28d). When H is nonsingular, we can compute g^+ as follows:

$$g^+ = Pr^+, \quad \text{where } P = H^{-1}(I - A^T(AH^{-1}A^T)^{-1}AH^{-1}). \quad (16.33)$$

Otherwise, when $z^T Hz \neq 0$ for all nonzero z with $Az = 0$, we can find g^+ as the solution of the system

$$\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} g^+ \\ v^+ \end{bmatrix} = \begin{bmatrix} r^+ \\ 0 \end{bmatrix}. \quad (16.34)$$

While (16.33) is unappealing when H^{-1} does not have a simple form, (16.34) is a useful generalization of (16.32). A “perfect” preconditioner is obtained by taking $H = G$, but other choices for H are also possible, provided that $Z^T HZ$ is positive definite. The matrix in (16.34) is often called a *constraint preconditioner*.

None of these procedures for computing the projection makes use of a null-space basis Z ; only the factorization of matrices involving A is required. Significantly, all these forms allow us to compute an initial point satisfying $Ax = b$. The operator $g^+ = Pr^+$ relies on a factorization of AA^T from which we can compute $x = A^T(AA^T)^{-1}b$, while factorizations of the system matrices in (16.32) and (16.34) allow us to find a suitable x by solving

$$\begin{bmatrix} I & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ b \end{bmatrix}.$$

Therefore we can compute an initial point for Algorithm 16.2 at the cost of one backsolve, using the factorization of the system needed to perform the projection operators.

We point out that these approaches for computing g^+ can give rise to significant round-off errors, so the use of iterative refinement is recommended to improve accuracy.

16.4 INEQUALITY-CONSTRAINED PROBLEMS

In the remainder of the chapter we discuss several classes of algorithms for solving convex quadratic programs that contain both inequality and equality constraints. *Active-set methods* have been widely used since the 1970s and are effective for small- and medium-sized problems. They allow for efficient detection of unboundedness and infeasibility and typically return an accurate estimate of the optimal active set. *Interior-point methods* are more recent, having become popular in the 1990s. They are well suited for large problems but may not be the most effective when a series of related QPs must be solved. We also study a special

type of active-set methods called a *gradient projection method*, which is most effective when the only constraints in the problem are bounds on the variables.

OPTIMALITY CONDITIONS FOR INEQUALITY-CONSTRAINED PROBLEMS

We begin our discussion with a brief review of the optimality conditions for inequality-constrained quadratic programming, then discuss some of the less obvious properties of the solutions.

Theorem 12.1 can be applied to (16.1) by noting that the Lagrangian for this problem is

$$\mathcal{L}(x, \lambda) = \frac{1}{2}x^T Gx + x^T c - \sum_{i \in \mathcal{I} \cup \mathcal{E}} \lambda_i (a_i^T x - b_i). \quad (16.35)$$

As in Definition 12.1, the active set $\mathcal{A}(x^*)$ consists of the indices of the constraints for which equality holds at x^* :

$$\mathcal{A}(x^*) = \{i \in \mathcal{E} \cup \mathcal{I} \mid a_i^T x^* = b_i\}. \quad (16.36)$$

By specializing the KKT conditions (12.34) to this problem, we find that any solution x^* of (16.1) satisfies the following first-order conditions, for some Lagrange multipliers λ_i^* , $i \in \mathcal{A}(x^*)$:

$$Gx^* + c - \sum_{i \in \mathcal{A}(x^*)} \lambda_i^* a_i = 0, \quad (16.37a)$$

$$a_i^T x^* = b_i, \quad \text{for all } i \in \mathcal{A}(x^*), \quad (16.37b)$$

$$a_i^T x^* \geq b_i, \quad \text{for all } i \in \mathcal{I} \setminus \mathcal{A}(x^*), \quad (16.37c)$$

$$\lambda_i^* \geq 0, \quad \text{for all } i \in \mathcal{I} \cap \mathcal{A}(x^*). \quad (16.37d)$$

A technical point: In Theorem 12.1 we assumed that the linear independence constraint qualification (LICQ) was satisfied. As mentioned in Section 12.6, this theorem still holds if we replace LICQ by other constraint qualifications, such as linearity of the constraints, which is certainly satisfied for quadratic programming. Hence, in the optimality conditions for quadratic programming given above, we need not assume that the active constraints are linearly independent at the solution.

For convex QP, when G is positive semidefinite, the conditions (16.37) are in fact sufficient for x^* to be a global solution, as we now prove.

Theorem 16.4.

If x^ satisfies the conditions (16.37) for some λ_i^* , $i \in \mathcal{A}(x^*)$, and G is positive semidefinite, then x^* is a global solution of (16.1).*

PROOF. If x is any other feasible point for (16.1), we have that $a_i^T x = b_i$ for all $i \in \mathcal{E}$ and $a_i^T x \geq b_i$ for all $i \in \mathcal{A}(x^*) \cap \mathcal{I}$. Hence, $a_i^T(x - x^*) = 0$ for all $i \in \mathcal{E}$ and $a_i^T(x - x^*) \geq 0$ for all $i \in \mathcal{A}(x^*) \cap \mathcal{I}$. Using these relationships, together with (16.37a) and (16.37d), we have that

$$(x - x^*)^T(Gx^* + c) = \sum_{i \in \mathcal{E}} \lambda_i^* a_i^T(x - x^*) + \sum_{i \in \mathcal{A}(x^*) \cap \mathcal{I}} \lambda_i^* a_i^T(x - x^*) \geq 0. \quad (16.38)$$

By elementary manipulation, we find that

$$\begin{aligned} q(x) &= q(x^*) + (x - x^*)^T(Gx^* + c) + \frac{1}{2}(x - x^*)^T G(x - x^*) \\ &\geq q(x^*) + \frac{1}{2}(x - x^*)^T G(x - x^*) \\ &\geq q(x^*), \end{aligned}$$

where the first inequality follows from (16.38) and the second inequality follows from positive semidefiniteness of G . We have shown that $q(x) \geq q(x^*)$ for any feasible x , so x^* is a global solution. \square

By a trivial modification of this proof, we see that x^* is actually the unique global solution when G is positive definite.

We can also apply the theory from Section 12.5 to derive second-order optimality conditions for (16.1). Second-order sufficient conditions for x^* to be a local minimizer are satisfied if $Z^T G Z$ is positive definite, where Z is defined to be a null-space basis matrix for the active constraint Jacobian matrix, which is the matrix whose rows are a_i^T for all $i \in \mathcal{A}(x^*)$. In this case, x^* is a strict local solution, according to Theorem 12.6.

When G is not positive definite, the general problem (16.1) may have more than one strict local solution. As mentioned above, such problems are called “nonconvex QPs” or “indefinite QPs,” and they cause some complications for algorithms. Examples of indefinite QPs are illustrated in Figure 16.1. On the left we have plotted the feasible region and the contours of a quadratic objective $q(x)$ in which G has one positive and one negative eigenvalue. We have indicated by $+$ or $-$ that the function tends toward plus or minus infinity in that direction. Note that x^{**} is a local maximizer, x^* a local minimizer, and the center of the box is a stationary point. The picture on the right in Figure 16.1, in which both eigenvalues of G are negative, shows a global maximizer at \tilde{x} and local minimizers at x_* and x_{**} .

DEGENERACY

A second property that causes difficulties for some algorithms is *degeneracy*. Confusingly, this term has been given a variety of meanings. It refers to situations in which

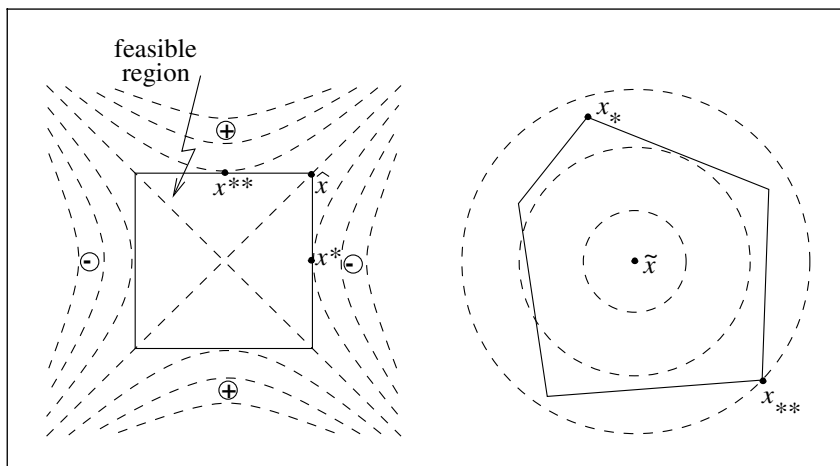


Figure 16.1 Nonconvex quadratic programs.

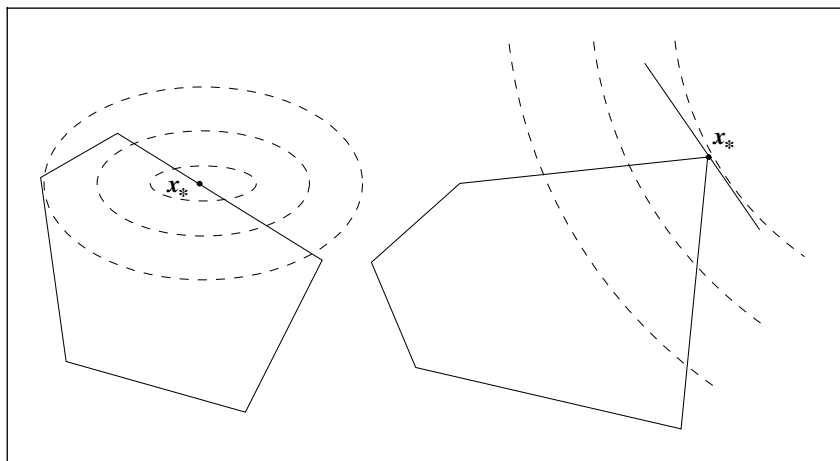


Figure 16.2 Degenerate solutions of quadratic programs.

- (a) the active constraint gradients $a_i, i \in \mathcal{A}(x^*)$, are linearly dependent at the solution x^* , and/or
- (b) the strict complementarity condition of Definition 12.5 fails to hold, that is, there is some index $i \in \mathcal{A}(x^*)$ such that all Lagrange multipliers satisfying (16.37) have $\lambda_i^* = 0$. (Such constraints are *weakly active* according to Definition 12.8.)

Two examples of degeneracy are shown in Figure 16.2. In the left-hand picture, there is a single active constraint at the solution x_* , which is also an unconstrained minimizer of the objective function. In the notation of (16.37a), we have that $G_{x_*} + c = 0$, so that

the lone Lagrange multiplier must be zero. In the right-hand picture, three constraints are active at the solution x_* . Since each of the three constraint gradients is a vector in \mathbb{R}^2 , they must be linearly dependent.

Lack of strict complementarity is also illustrated by the problem

$$\min x_1^2 + (x_2 + 1)^2 \quad \text{subject to } x \geq 0,$$

which has a solution at $x^* = 0$ at which both constraints are active. Strict complementarity does not hold at x^* because the Lagrange multiplier associated with the active constraint $x_1 \geq 0$ is zero.

Degeneracy can cause problems for algorithms for two main reasons. First, linear dependence of the active constraint gradients can cause numerical difficulties in the step computation because certain matrices that we need to factor become rank deficient. Second, when the problem contains weakly active constraints, it is difficult for the algorithm to determine whether these constraints are active at the solution. In the case of active-set methods and gradient projection methods (described below), this indecisiveness can cause the algorithm to zigzag as the iterates move on and off the weakly active constraints on successive iterations. Safeguards must be used to prevent such behavior.

16.5 ACTIVE-SET METHODS FOR CONVEX QPs

We now describe active-set methods for solving quadratic programs of the form (16.1) containing equality and inequality constraints. We consider only the convex case, in which the matrix G in (16.1a) is positive semidefinite. The case in which G is an indefinite matrix raises complications in the algorithms and is outside the scope of this book. We refer to Gould [147] for a discussion of nonconvex QPs.

If the contents of the *optimal* active set (16.36) were known in advance, we could find the solution x^* by applying one of the techniques for equality-constrained QP of Sections 16.2 and 16.3 to the problem

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

Of course, we usually do not have prior knowledge of $\mathcal{A}(x^*)$ and, as we 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. In essence, the simplex method starts by making a guess of the optimal active set, then repeatedly uses gradient and Lagrange multiplier information to drop one index from the current estimate of $\mathcal{A}(x^*)$ and add a new index, until optimality

is detected. Active-set methods for QP differ from the simplex method in that the iterates (and the solution x^*) are not necessarily vertices of the feasible region.

Active-set methods for QP come in three varieties: *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 objective function $q(x)$.

Primal active-set methods find a step from one iterate to the next by solving a quadratic subproblem in which some of the inequality constraints (16.1c), and all the equality constraints (16.1b), are imposed as equalities. This subset is referred to as the *working set* and is denoted at the k th iterate x_k by \mathcal{W}_k . 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.

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

$$p = x - x_k, \quad g_k = Gx_k + c.$$

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

$$q(x) = q(x_k + p) = \frac{1}{2}p^T Gp + g_k^T p + \rho_k,$$

where $\rho_k = \frac{1}{2}x_k^T Gx_k + c^T x_k$ is independent of p . Since we can drop ρ_k from the objective without changing the solution of the problem, we can write the QP subproblem to be solved at the k th iteration as follows:

$$\min_p \quad \frac{1}{2}p^T Gp + g_k^T p \tag{16.39a}$$

$$\text{subject to} \quad a_i^T p = 0, \quad i \in \mathcal{W}_k. \tag{16.39b}$$

We denote the solution of this subproblem by p_k . Note that for each $i \in \mathcal{W}_k$, the value of $a_i^T x$ does not change as we move along p_k , since we have $a_i^T(x_k + \alpha p_k) = a_i^T x_k = b_i$ for all α . 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 . Since G is positive definite, the solution of (16.39) can be computed by any of the techniques described in Section 16.2.

Supposing for the moment that the optimal p_k from (16.39) 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

$$x_{k+1} = x_k + \alpha_k p_k, \tag{16.40}$$

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, constraint i 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

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

To maximize the decrease in q , we want α_k to be as large as possible in $[0, 1]$ subject to retaining feasibility, so we obtain the following definition:

$$\alpha_k \stackrel{\text{def}}{=} \min \left(1, \min_{i \notin \mathcal{W}_k, a_i^T p_k < 0} \frac{b_i - a_i^T x_k}{a_i^T p_k} \right). \quad (16.41)$$

We call the constraints i for which the minimum in (16.41) 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, because 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 (16.39) has solution $p = 0$. Since $p = 0$ satisfies the optimality conditions (16.5) for (16.39), we have that

$$\sum_{i \in \hat{\mathcal{W}}} a_i \hat{\lambda}_i = g = G\hat{x} + c, \quad (16.42)$$

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 (16.37a), 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 (16.37b) and (16.37c) are satisfied at this point.

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 (16.37d) 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 have

from Theorem 16.4 that \hat{x} is a global solution of (16.1). (As noted after Theorem 16.4, \hat{x} is a strict local minimizer and the unique global solution if G is positive definite.)

If, on the other hand, one or more of the multipliers $\hat{\lambda}_j$, $j \in \hat{\mathcal{W}} \cap \mathcal{I}$, is negative, the condition (16.37d) is not satisfied and the objective function $q(\cdot)$ may be decreased by dropping one of these constraints, as shown in Section 12.3. Thus, we remove an index j corresponding to one of the negative multipliers from the working set and solve a new subproblem (16.39) 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 constraint gradients a_i for i in the working set are linearly independent. After the algorithm has been fully stated, we discuss how this property can be maintained.

Theorem 16.5.

Suppose that the point \hat{x} satisfies first-order conditions for the equality-constrained subproblem with working set $\hat{\mathcal{W}}$; that is, equation (16.42) 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 $\hat{\lambda}_j < 0$. Let p be the solution obtained by dropping the constraint j and solving the following subproblem:

$$\min_p \frac{1}{2} p^T G p + (G \hat{x} + c)^T p, \quad (16.43a)$$

$$\text{subject to } a_i^T p = 0, \text{ for all } i \in \hat{\mathcal{W}} \text{ with } i \neq j. \quad (16.43b)$$

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 (16.43), then we have that $a_j^T p > 0$, and that p is a descent direction for $q(\cdot)$.

PROOF. Since p solves (16.43), 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

$$\sum_{i \in \hat{\mathcal{W}}, i \neq j} \tilde{\lambda}_i a_i = G p + (G \hat{x} + c). \quad (16.44)$$

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 $p = Z p_z$ 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 (16.42). By subtracting (16.42) from (16.44), we obtain

$$\sum_{i \in \hat{\mathcal{W}}, i \neq j} (\tilde{\lambda}_i - \hat{\lambda}_i) a_i - \hat{\lambda}_j a_j = Gp. \quad (16.45)$$

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

$$-\hat{\lambda}_j a_j^T p = p^T Gp. \quad (16.46)$$

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

If the second-order sufficient conditions of Section 12.5 are satisfied, we have that $Z^T GZ$ defined above is positive definite. From (16.46), we can have $a_j^T p = 0$ only if $p^T Gp = p_z^T Z^T GZ p_z = 0$, which happens only if $p_z = 0$ and $p = 0$. But if $p = 0$, then by substituting into (16.45) 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 $p^T Gp > 0$ in (16.46), and therefore $a_j^T p > 0$ whenever p satisfies the second-order sufficient conditions for (16.43).

The claim that p is a descent direction for $q(\cdot)$ is proved in Theorem 16.6 below. \square

While any index j for which $\hat{\lambda}_j < 0$ usually will yield a direction p 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. As in linear programming, however, the step along the resulting direction may be short (as when it is blocked by a new constraint), so the amount of decrease in q is not guaranteed to be greater than for other possible choices of j .

We conclude with a result that shows that whenever p_k obtained from (16.39) is nonzero and satisfies second-order sufficient optimality conditions for the current working set, it is a direction of strict descent for $q(\cdot)$.

Theorem 16.6.

Suppose that the solution p_k of (16.39) 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, $Z^T GZ$ is positive definite for the matrix Z whose columns are a basis of the null space of the constraints (16.39b), we have by applying Theorem 16.2 to (16.39) that p_k is the unique global solution of (16.39). Since $p = 0$ is also a feasible point for (16.39), its objective value in (16.39a) must be larger

than that of p_k , so we have

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

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

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

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

When G is positive definite—the *strictly* convex case—the second-order sufficient conditions are satisfied for *all* feasible subproblems of the form (16.39). Hence, it follows from the result above that we obtain a strict decrease in $q(\cdot)$ whenever $p_k \neq 0$. This fact is significant when we discuss finite termination of the algorithm.

SPECIFICATION OF THE ACTIVE-SET METHOD FOR CONVEX QP

Having described the active-set algorithm for convex QP, we now present the following formal specification. We assume that the objective function q is bounded in the feasible set (16.1b), (16.1c).

Algorithm 16.3 (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  $k = 0, 1, 2, \dots$ 
    Solve (16.39) to find  $p_k$ ;
    if  $p_k = 0$ 
        Compute Lagrange multipliers  $\hat{\lambda}_i$  that satisfy (16.42),
            with  $\hat{\mathcal{W}} = \mathcal{W}_k$ ;
        if  $\hat{\lambda}_i \geq 0$  for all  $i \in \mathcal{W}_k \cap \mathcal{I}$ 
            stop with solution  $x^* = x_k$ ;
        else
             $j \leftarrow \arg \min_{j \in \mathcal{W}_k \cap \mathcal{I}} \hat{\lambda}_j$ ;
             $x_{k+1} \leftarrow x_k$ ;  $\mathcal{W}_{k+1} \leftarrow \mathcal{W}_k \setminus \{j\}$ ;
        else (*  $p_k \neq 0$  *)
            Compute  $\alpha_k$  from (16.41);
             $x_{k+1} \leftarrow x_k + \alpha_k p_k$ ;
            if there are blocking constraints
                Obtain  $\mathcal{W}_{k+1}$  by adding one of the blocking
                    constraints to  $\mathcal{W}_k$ ;
            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 for linear programming 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 a good choice based on knowledge of the QP may reduce the work needed in the Phase I step.

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

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

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

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

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 Algorithm 16.3 can be found by taking a linearly independent subset of the active constraints at the solution of the feasibility problem.

An alternative approach is a penalty (or “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 η into (16.1) to measure the constraint violation, and we solve the problem

$$\begin{aligned} & \min_{(x,\eta)} \frac{1}{2} x^T G x + x^T c + M \eta, \\ & \text{subject to } (a_i^T x - b_i) \leq \eta, \quad i \in \mathcal{E}, \\ & \quad -(a_i^T x - b_i) \leq \eta, \quad i \in \mathcal{E}, \\ & \quad b_i - a_i^T x \leq \eta, \quad i \in \mathcal{I}, \\ & \quad 0 \leq \eta, \end{aligned} \tag{16.47}$$

for some large positive value of M . It can be shown by applying the theory of exact penalty functions (see Chapter 17) that whenever there exist feasible points for the original problem (16.1), then for all M sufficiently large, the solution of (16.47) will have $\eta = 0$, 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 (16.47) by the usual means. If the solution we obtain has a positive value of η , we increase M and try again. Note that a feasible point is easy to obtain for the subproblem (16.47): We set $x = \tilde{x}$ (where, as before, \tilde{x} is the user-supplied initial guess) and choose η large enough that all the constraints in (16.47) are satisfied. This approach is, in fact, an exact penalty method using the ℓ_∞ norm; see Chapter 17.

A variant of (16.47) that penalizes the ℓ_1 norm of the constraint violation rather than the ℓ_∞ norm is as follows:

$$\begin{aligned} \min_{(x,s,t,v)} \quad & \frac{1}{2}x^T Gx + x^T c + Me_{\mathcal{E}}^T(s+t) + Me_{\mathcal{I}}^T v \\ \text{subject to} \quad & a_i^T x - b_i + s_i - t_i = 0, \quad i \in \mathcal{E}, \\ & a_i^T x - b_i + v_i \geq 0, \quad i \in \mathcal{I}, \\ & s \geq 0, \quad t \geq 0, \quad v \geq 0. \end{aligned} \quad (16.48)$$

Here, $e_{\mathcal{E}}$ is the vector $(1, 1, \dots, 1)^T$ of length $|\mathcal{E}|$; similarly for $e_{\mathcal{I}}$. The slack variables s_i , t_i , and v_i soak up any infeasibility in the constraints.

In the following example we use subscripts on the vectors x and p to denote their components, and we use superscripts to indicate the iteration index. For example, x_1 denotes the first component, while x^4 denotes the fourth iterate of the vector x .

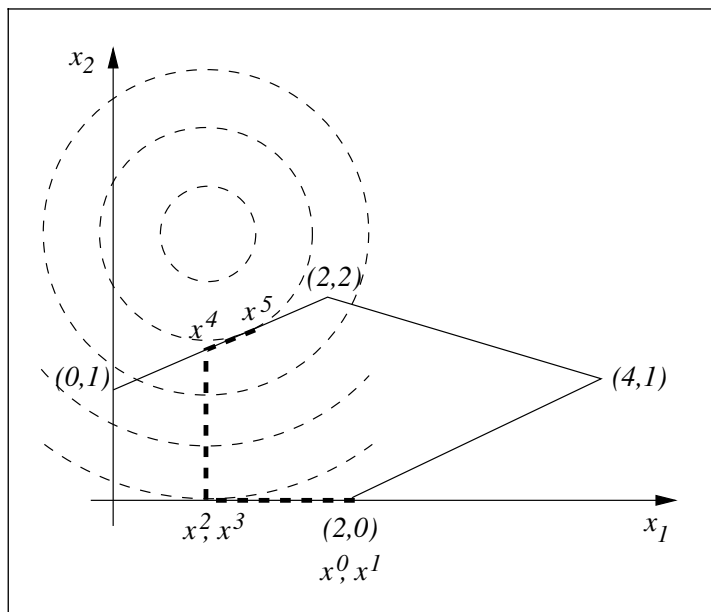


Figure 16.3 Iterates of the active-set method.

□ EXAMPLE 16.4

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

$$\min_x q(x) = (x_1 - 1)^2 + (x_2 - 2.5)^2 \quad (16.49a)$$

$$\text{subject to } x_1 - 2x_2 + 2 \geq 0, \quad (16.49b)$$

$$-x_1 - 2x_2 + 6 \geq 0, \quad (16.49c)$$

$$-x_1 + 2x_2 + 2 \geq 0, \quad (16.49d)$$

$$x_1 \geq 0, \quad (16.49e)$$

$$x_2 \geq 0. \quad (16.49f)$$

We refer the constraints, in order, by indices 1 through 5. For this problem it is easy to determine a feasible initial point; say $x^0 = (2, 0)^T$. 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 choice would lead the algorithm to perform somewhat 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 (16.39) with $k = 0$ is $p = 0$. We can then use (16.42) to find the multipliers $\hat{\lambda}_3$ and $\hat{\lambda}_5$ associated with the active constraints. Substitution of the data from our problem into (16.42) 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, because it has the most negative multiplier, and set $\mathcal{W}_1 = \{5\}$. We begin iteration 1 by finding the solution of (16.39) for $k = 1$, which is $p^1 = (-1, 0)^T$. The step-length formula (16.41) yields $\alpha_1 = 1$, and the new iterate is $x^2 = (1, 0)^T$.

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 (16.39) is $p^2 = 0$. From (16.42) 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 $p^3 = (0, 2.5)^T$. The formula (16.41) yields a step length of $\alpha_3 = 0.6$ and a new iterate $x^4 = (1, 1.5)^T$. There is a single blocking constraint (constraint 1), so we obtain $\mathcal{W}_4 = \{1\}$. The solution of (16.39) for $k = 4$ is then $p^4 = (0.4, 0.2)^T$, 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 $x^5 = (1.4, 1.7)^T$.

Finally, we solve (16.39) for $k = 5$ to obtain a solution $p^5 = 0$. The formula (16.42) yields a multiplier $\hat{\lambda}_1 = 0.8$, so we have found the solution. We set $x^* = (1.4, 1.7)^T$ 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 $p^0 = (0.2, 0.1)^T$ and a new iterate of $x^1 = (2.2, 0.1)^T$. If we had chosen $\mathcal{W}_0 = \{5\}$, we would have moved immediately to the new iterate $x^1 = (1, 0)^T$, without first performing the operation of dropping the index 3, as is done in the example. If we had selected $\mathcal{W}_0 = \emptyset$, we would have obtained $p^1 = (-1, 2.5)^T$, $\alpha_1 = \frac{2}{3}$, a new iterate of $x^1 = (\frac{4}{3}, \frac{5}{3})^T$, 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 the constraint gradients in \mathcal{W}_0 to 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, its constraint normal cannot be a linear combination of the normals a_i in the current working set (see Exercise 16.18). 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 cannot 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 noted there, strategies that are less sensitive to scaling often give better results. We do not discuss this advanced topic further.

We note that the strategy of adding or deleting at most one constraint at each iteration of the Algorithm 16.3 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 inequality 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 will 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 ACTIVE-SET ALGORITHM ON STRICTLY CONVEX QPs

It is not difficult to show that, under certain assumptions, Algorithm 16.3 converges for strictly convex QPs, 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 (16.39) is nonzero. Our argument proceeds as follows:

- If the solution of (16.39) 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 16.6. If it is not the solution of the original problem (16.1) (that is, at least one of the Lagrange multipliers is negative), Theorems 16.5 and 16.6 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 , because 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 (16.39) at least on every n th iteration. To demonstrate this claim, we note that for any k at 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 (16.39) will then be $p_k = 0$, since only the zero vector will satisfy the constraints (16.39b).
- 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 (16.39) 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 each iterate in the cycle, a constraint is dropped (as in Theorem 16.5), 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 do 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 16.3 requires the solution of the equality-constrained subproblem (16.39). 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 and/or deletion of a single row.

It follows from this observation that we can compute the matrix factors needed to solve (16.39) 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 limit our discussion to the case in which the step is computed with the null-space method (16.17)–(16.20). 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 Section 15.3 for details). Thus

$$A^T \Pi = Q \begin{bmatrix} R \\ 0 \end{bmatrix} = [Q_1 \quad Q_2] \begin{bmatrix} R \\ 0 \end{bmatrix} \quad (16.50)$$

(see (15.21)), where Π is a permutation matrix; R is square, upper triangular and nonsingular; $Q = [Q_1 \quad Q_2]$ is $n \times n$ orthogonal; and Q_1 and R both have m columns while 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 $\bar{A}^T = [A^T \quad a]$, 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 (16.50) 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 $Q_1 Q_1^T + Q_2 Q_2^T = I$, we have

$$\bar{A}^T \begin{bmatrix} \Pi & 0 \\ 0 & 1 \end{bmatrix} = [A^T \Pi \quad a] = Q \begin{bmatrix} R & Q_1^T a \\ 0 & Q_2^T a \end{bmatrix}. \quad (16.51)$$

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 (16.51) we now have

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

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 (16.50) from scratch, which is of order $n^2 m$. The updating strategy is less expensive, especially when the null space is small (that is, when $n - m \ll n$).

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 (16.50), thus disturbing 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 the first m columns of 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 has the form

$$\bar{Z} = \begin{bmatrix} \bar{z} & Z \end{bmatrix}, \quad (16.52)$$

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 general cheaper

than recomputing a QR factorization from scratch. For details of these procedures, see Gill et al. [124, Section 5].

We now consider the reduced Hessian. Because of the special form of (16.39), we have $h = 0$ in (16.5), and the step p_y given in (16.18) is zero. Thus from (16.19), the null-space component p_z is the solution of

$$(Z^T G Z)p_z = -Z^T g. \quad (16.53)$$

We can sometimes find ways of updating the factorization of the reduced Hessian $Z^T G Z$ 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 (16.52), gaining a column after deletion of a constraint. A series of inexpensive, elementary operations can 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 16.7, we can update the reduced gradient $Z^T g$ at the same time as we update Z to \bar{Z} .

16.6 INTERIOR-POINT METHODS

The interior-point approach can be applied to convex quadratic programs through a simple extension of the linear-programming algorithms described in Chapter 14. The resulting primal-dual algorithms are easy to describe and are quite efficient on many types of problems. Extensions of interior-point methods to nonconvex problems are discussed in Chapter 19.

For simplicity, we restrict our attention to convex quadratic programs with inequality constraints, which we write as follows:

$$\min_x \quad q(x) = \frac{1}{2}x^T Gx + x^T c \quad (16.54a)$$

$$\text{subject to} \quad Ax \geq b, \quad (16.54b)$$

where G is symmetric and positive semidefinite and where the $m \times n$ matrix A and right-hand side b are defined by

$$A = [a_i]_{i \in \mathcal{I}}, \quad b = [b_i]_{i \in \mathcal{I}}, \quad \mathcal{I} = \{1, 2, \dots, m\}.$$

(If equality constraints are also present, they can be accommodated with simple extensions to the approaches described below.) Rewriting the KKT conditions (16.37) in this notation,

we obtain

$$\begin{aligned} Gx - A^T\lambda + c &= 0, \\ Ax - b &\geq 0, \\ (Ax - b)_i\lambda_i &= 0, \quad i = 1, 2, \dots, m, \\ \lambda &\geq 0. \end{aligned}$$

By introducing the slack vector $y \geq 0$, we can rewrite these conditions as

$$Gx - A^T\lambda + c = 0, \quad (16.55a)$$

$$Ax - y - b = 0, \quad (16.55b)$$

$$y_i\lambda_i = 0, \quad i = 1, 2, \dots, m, \quad (16.55c)$$

$$(y, \lambda) \geq 0. \quad (16.55d)$$

Since we assume that G is positive semidefinite, these KKT conditions are not only necessary but also sufficient (see Theorem 16.4), so we can solve the convex quadratic program (16.54) by finding solutions of the system (16.55).

Given a current iterate (x, y, λ) that satisfies $(y, \lambda) > 0$, we can define a complementarity measure μ by

$$\mu = \frac{y^T\lambda}{m}. \quad (16.56)$$

As in Chapter 14, we derive path-following, primal-dual methods by considering the *perturbed* KKT conditions given by

$$F(x, y, \lambda; \sigma\mu) = \begin{bmatrix} Gx - A^T\lambda + c \\ Ax - y - b \\ \mathcal{Y}\Lambda e - \sigma\mu e \end{bmatrix} = 0, \quad (16.57)$$

where

$$\mathcal{Y} = \text{diag}(y_1, y_2, \dots, y_m), \quad \Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_m), \quad e = (1, 1, \dots, 1)^T,$$

and $\sigma \in [0, 1]$. The solutions of (16.57) for all positive values of σ and μ define the *central path*, which is a trajectory that leads to the solution of the quadratic program as $\sigma\mu$ tends to zero.

By fixing μ and applying Newton's method to (16.57), we obtain the linear system

$$\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & \mathcal{Y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p \\ -\Lambda\mathcal{Y}e + \sigma\mu e \end{bmatrix}, \quad (16.58)$$

where

$$r_d = Gx - A^T\lambda + c, \quad r_p = Ax - y - b. \quad (16.59)$$

We obtain the next iterate by setting

$$(x^+, y^+, \lambda^+) = (x, y, \lambda) + \alpha(\Delta x, \Delta y, \Delta \lambda), \quad (16.60)$$

where α is chosen to retain the inequality $(y^+, \lambda^+) > 0$ and possibly to satisfy various other conditions.

In the rest of the chapter we discuss several enhancements of this primal-dual iteration that make it effective in practice.

SOLVING THE PRIMAL-DUAL SYSTEM

The major computational operation in the interior-point method is the solution of the system (16.58). The coefficient matrix in this system can be much more costly to factor than the matrix (14.9) arising in linear programming because of the presence of the Hessian matrix G . It is therefore important to exploit the structure of (16.58) by choosing a suitable direct factorization algorithm, or by choosing an appropriate preconditioner for an iterative solver.

As in Chapter 14, the system (16.58) may be restated in more compact forms. The “augmented system” form is

$$\begin{bmatrix} G & -A^T \\ A & \Lambda^{-1}\mathcal{Y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p + (-y + \sigma\mu\Lambda^{-1}e) \end{bmatrix}. \quad (16.61)$$

After a simple transformation to symmetric form, a symmetric indefinite factorization scheme can be applied to the coefficient matrix in this system. The “normal equations” form (14.44a) is

$$(G + A^T\mathcal{Y}^{-1}\Lambda A)\Delta x = -r_d + A^T\mathcal{Y}^{-1}\Lambda[-r_p - y + \sigma\mu\Lambda^{-1}e], \quad (16.62)$$

which can be solved by means of a modified Cholesky algorithm. This approach is effective if the term $A^T(\mathcal{Y}^{-1}\Lambda)A$ is not too dense compared with G , and it has the advantage of being much smaller than (16.61) if there are many inequality constraints.

The projected CG method of Algorithm 16.2 can also be effective for solving the primal-dual system. We can rewrite (16.58) in the form

$$\begin{bmatrix} G & 0 & -A^T \\ 0 & \mathcal{Y}^{-1}\Lambda & I \\ A & -I & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -\Lambda e + \sigma\mu\mathcal{Y}^{-1}e \\ -r_p \end{bmatrix}, \quad (16.63)$$

and observe that these are the optimality conditions for an equality-constrained convex quadratic program of the form (16.3), in which the variable is $(\Delta x, \Delta y)$. Hence, we can make appropriate substitutions and solve this system using Algorithm 16.2. This approach may be useful for problems in which the direct factorization cannot be performed due to excessive memory demands. The projected CG method does not require that the matrix G be formed or factored; it requires only matrix-vector products.

STEP LENGTH SELECTION

We mentioned in Chapter 14 that interior-point methods for linear programming are more efficient if different step lengths α^{pri} , α^{dual} are used for the primal and dual variables. Equation (14.37) indicates that the greatest reduction in the residuals r_b and r_c is obtained by choosing the largest admissible primal and dual step lengths. The situation is different in quadratic programming. Suppose that we define the new iterate as

$$(x^+, y^+) = (x, y) + \alpha^{\text{pri}}(\Delta x, \Delta y), \quad \lambda^+ = \lambda + \alpha^{\text{dual}}\Delta\lambda, \quad (16.64)$$

where α^{pri} and α^{dual} are step lengths that ensure the positivity of (y^+, λ^+) . By using (16.58) and (16.59), we see that the new residuals satisfy the following relations:

$$r_p^+ = (1 - \alpha^{\text{pri}})r_p, \quad (16.65a)$$

$$r_d^+ = (1 - \alpha^{\text{dual}})r_d + (\alpha^{\text{pri}} - \alpha^{\text{dual}})G\Delta x. \quad (16.65b)$$

If $\alpha^{\text{pri}} = \alpha^{\text{dual}} = \alpha$ then both residuals decrease linearly for all $\alpha \in (0, 1)$. For different step lengths, however, the dual residual r_d^+ may increase for certain choices of α^{pri} , α^{dual} , possibly causing divergence of the interior-point iteration.

One option is to use equal step lengths, as in (16.60), and to set $\alpha = \min(\alpha_\tau^{\text{pri}}, \alpha_\tau^{\text{dual}})$, where

$$\alpha_\tau^{\text{pri}} = \max\{\alpha \in (0, 1] : y + \alpha\Delta y \geq (1 - \tau)y\}, \quad (16.66a)$$

$$\alpha_\tau^{\text{dual}} = \max\{\alpha \in (0, 1] : \lambda + \alpha\Delta\lambda \geq (1 - \tau)\lambda\}; \quad (16.66b)$$

the parameter $\tau \in (0, 1)$ controls how far we back off from the maximum step for which the conditions $y + \alpha\Delta y \geq 0$ and $\lambda + \alpha\Delta\lambda \geq 0$ are satisfied. Numerical experience has shown, however, that using different step lengths in the primal and dual variables often leads to faster convergence. One way to choose unequal step lengths is to select $(\alpha^{\text{pri}}, \alpha^{\text{dual}})$ so as to (approximately) minimize the optimality measure

$$\|Gx^+ - A^T\lambda^+ + c\|_2^2 + \|Ax^+ - y^+ - b\|_2^2 + (y^+)^T z^+,$$

subject to $0 \leq \alpha^{\text{pri}} \leq \alpha_\tau^{\text{pri}}$ and $0 \leq \alpha^{\text{dual}} \leq \alpha_\tau^{\text{dual}}$, where x^+ , y^+ , λ^+ are defined as a function of the step lengths through (16.64).

A PRACTICAL PRIMAL-DUAL METHOD

The most popular interior-point method for convex QP is based on Mehrotra's predictor-corrector, originally developed for linear programming (see Section 14.2). The extension to quadratic programming is straightforward, as we now show.

First, we compute an affine scaling step $(\Delta x^{\text{aff}}, \Delta y^{\text{aff}}, \Delta \lambda^{\text{aff}})$ by setting $\sigma = 0$ in (16.58). We improve upon this step by computing a corrector step, which is defined following the same reasoning that leads to (14.31). Next, we compute the centering parameter σ using (14.34). The total step is obtained by solving the following system (cf. (14.35)):

$$\begin{bmatrix} G & 0 & -A^T \\ A & -I & 0 \\ 0 & \Lambda & \mathcal{Y} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} -r_d \\ -r_p \\ -\Lambda \mathcal{Y} e - \Delta \Lambda^{\text{aff}} \Delta \mathcal{Y}^{\text{aff}} e + \sigma \mu e \end{bmatrix}. \quad (16.67)$$

We now specify the algorithm. For simplicity, we will assume in our description that equal step lengths are used in the primal and dual variables though, as noted above, unequal step lengths can give slightly faster convergence.

Algorithm 16.4 (Predictor-Corrector Algorithm for QP).

Compute (x_0, y_0, λ_0) with $(y_0, \lambda_0) > 0$;
for $k = 0, 1, 2, \dots$
 Set $(x, y, \lambda) = (x_k, y_k, \lambda_k)$ and solve (16.58) with $\sigma = 0$ for
 $(\Delta x^{\text{aff}}, \Delta y^{\text{aff}}, \Delta \lambda^{\text{aff}})$;
 Calculate $\mu = y^T \lambda / m$;
 Calculate $\hat{\alpha}_{\text{aff}} = \max\{\alpha \in (0, 1] \mid (y, \lambda) + \alpha(\Delta y^{\text{aff}}, \Delta \lambda^{\text{aff}}) \geq 0\}$;
 Calculate $\mu_{\text{aff}} = (y + \hat{\alpha}_{\text{aff}} \Delta y^{\text{aff}})^T (\lambda + \hat{\alpha}_{\text{aff}} \Delta \lambda^{\text{aff}}) / m$;
 Set centering parameter to $\sigma = (\mu_{\text{aff}} / \mu)^3$;
 Solve (16.67) for $(\Delta x, \Delta y, \Delta \lambda)$;
 Choose $\tau_k \in (0, 1)$ and set $\hat{\alpha} = \min(\alpha_{\tau_k}^{\text{pri}}, \alpha_{\tau_k}^{\text{dual}})$ (see (16.66));
 Set $(x_{k+1}, y_{k+1}, \lambda_{k+1}) = (x_k, y_k, \lambda_k) + \hat{\alpha}(\Delta x, \Delta y, \Delta \lambda)$;
end (for)

We can choose τ_k to approach 1 as the iterates approach the solution, to accelerate the convergence.

As for linear programming, efficiency and robustness of this approach is greatly enhanced if we choose a good starting point. This selection can be done in several ways. The following simple heuristic accepts an initial point $(\bar{x}, \bar{y}, \bar{\lambda})$ from the user and moves it far enough away from the boundary of the region $(y, \lambda) \geq 0$ to permit the algorithm to take long steps on early iterations. First, we compute the affine scaling step $(\Delta x^{\text{aff}}, \Delta y^{\text{aff}}, \Delta \lambda^{\text{aff}})$

from the user-supplied initial point $(\bar{x}, \bar{y}, \bar{\lambda})$, then set

$$y_0 = \max(1, |\bar{y} + \Delta y^{\text{aff}}|), \quad \lambda_0 = \max(1, |\bar{\lambda} + \Delta \lambda^{\text{aff}}|), \quad x_0 = \bar{x},$$

where the max and absolute values are applied component-wise.

We conclude this section by contrasting some of the properties of active-set and interior-point methods for convex quadratic programming. Active-set methods generally require a large number of steps in which each search direction is relatively inexpensive to compute, while interior-point methods take a smaller number of more expensive steps. Active-set methods are more complicated to implement, particularly if the procedures for updating matrix factorizations try to take advantage of sparsity or structure in G and A . By contrast, the nonzero structure of the matrix to be factored at each interior-point iteration remains the same at all iterations (though the numerical values change), so standard sparse factorization software can be used to obtain the steps. For particular sparsity structures (for example, bandedness in the matrices A and G), efficient customized solvers for the linear system arising at each interior-point iteration can be devised.

For very large problems, interior-point methods are often more efficient. However, when an estimate of the solution is available (a “warm start”), the active-set approach may converge rapidly in just a few iterations, particularly if the initial value of x is feasible. Interior-point methods are less able to exploit a warm start, though research efforts to improve their performance in this regard are ongoing.

16.7 THE GRADIENT PROJECTION METHOD

In the active-set method described in Section 16.5, the active set and working set change slowly, usually by a single index at each iteration. This method may thus 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 (nondegenerate) solution, then at least 200 iterations of the active-set method will be required to reach the solution.

The gradient projection method allows the active set to change rapidly from iteration to iteration. It is most efficient when the constraints are simple in form—in particular, when there are only bounds on the variables. Accordingly, we restrict our attention to the following bound-constrained problem:

$$\min_x \quad q(x) = \frac{1}{2}x^T Gx + x^T c \quad (16.68a)$$

$$\text{subject to} \quad l \leq x \leq u, \quad (16.68b)$$

where G is symmetric and l and u are vectors of lower and upper bounds on the components of x . We do not make any positive definiteness assumptions on G in this section, because the gradient projection approach can be applied to both convex and nonconvex problems. The

feasible region defined by (16.68b) is sometimes called a “box” because of its rectangular shape. Some components of x may lack an upper or a lower bound; we handle these cases formally by setting the appropriate components of l and u to $-\infty$ and $+\infty$, respectively.

Each iteration of the gradient projection algorithm consists of two stages. In the first stage, we search along the steepest descent direction from the current point x , that is, the direction $-g$, where $g = Gx + c$; see (16.6). Whenever a bound is encountered, the search direction is “bent” so that it stays feasible. We search along the resulting piecewise-linear path and locate the first local minimizer of q , which we denote by x^c and refer to as the *Cauchy point*, by analogy with our terminology of Chapter 4. The working set is now defined to be the set of bound constraints that are active at the Cauchy point, denoted by $\mathcal{A}(x^c)$. In the second stage of each gradient projection iteration, we explore the face of the feasible box on which the Cauchy point lies by solving a subproblem in which the active components x_i for $i \in \mathcal{A}(x^c)$ are fixed at the values x_i^c .

We describe the gradient projection method in detail in the rest of this section. Our convention in this section is to denote the iteration number by a superscript (that is, x^k) and use subscripts to denote the elements of a vector.

CAUCHY POINT COMPUTATION

We now derive an explicit expression for the piecewise-linear path obtained by projecting the steepest descent direction onto the feasible box, and outline the search procedure for identifying the first local minimum of q along this path.

The projection of an arbitrary point x onto the feasible region (16.68b) is defined as follows. The i th component is given by

$$P(x, l, u)_i = \begin{cases} l_i & \text{if } x_i < l_i, \\ x_i & \text{if } x_i \in [l_i, u_i], \\ u_i & \text{if } x_i > u_i. \end{cases} \quad (16.69)$$

(We assume, without loss of generality, that $l_i < u_i$ for all i .) The piecewise-linear path $x(t)$ starting at the reference point x and obtained by projecting the steepest descent direction at x onto the feasible region (16.68b) is thus given by

$$x(t) = P(x - tg, l, u), \quad (16.70)$$

where $g = Gx + c$; see Figure 16.4.

The Cauchy point x^c , is defined as the first local minimizer of the univariate, piecewise-quadratic function $q(x(t))$, for $t \geq 0$. This minimizer is obtained by examining each of the line segments that make up $x(t)$. To perform this search, we need to determine the values of t at which the kinks in $x(t)$, or *breakpoints*, occur. We first identify the values of t for which each component reaches its bound along the chosen direction $-g$. These values \bar{t}_i are given

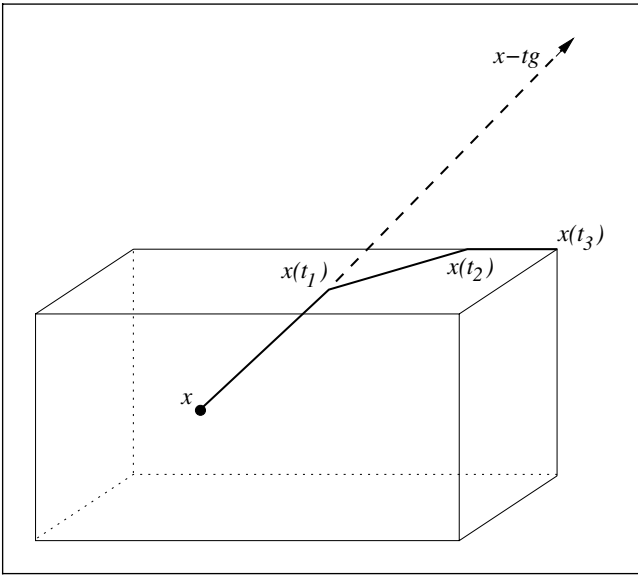


Figure 16.4 The piecewise-linear path $x(t)$, for an example in \mathbb{R}^3 .

by the following explicit formulae:

$$\bar{t}_i = \begin{cases} (x_i - u_i)/g_i & \text{if } g_i < 0 \text{ and } u_i < +\infty, \\ (x_i - l_i)/g_i & \text{if } g_i > 0 \text{ and } l_i > -\infty, \\ \infty & \text{otherwise.} \end{cases} \quad (16.71)$$

The components of $x(t)$ for any t are therefore

$$x_i(t) = \begin{cases} x_i - tg_i & \text{if } t \leq \bar{t}_i, \\ x_i - \bar{t}_i g_i & \text{otherwise.} \end{cases}$$

To search for the first local minimizer along $P(x - tg, l, u)$, we eliminate the duplicate values and zero values of \bar{t}_i from the set $\{\bar{t}_1, \bar{t}_2, \dots, \bar{t}_n\}$, to obtain a sorted, reduced set of breakpoints $\{t_1, t_2, \dots, t_l\}$ with $0 < t_1 < t_2 < \dots$. We now examine the intervals $[0, t_1], [t_1, t_2], [t_2, t_3], \dots$ in turn. Suppose we have examined up to t_{j-1} and have not yet found a local minimizer. For the interval $[t_{j-1}, t_j]$, we have that

$$x(t) = x(t_{j-1}) + (\Delta t)p^{j-1},$$

where

$$\Delta t = t - t_{j-1} \in [0, t_j - t_{j-1}],$$

and

$$p_i^{j-1} = \begin{cases} -g_i & \text{if } t_{j-1} < \bar{t}_i, \\ 0 & \text{otherwise.} \end{cases} \quad (16.72)$$

We can then write the quadratic (16.68a) on the line segment $[x(t_{j-1}), x(t_j)]$ as follows:

$$q(x(t)) = c^T(x(t_{j-1}) + (\Delta t)p^{j-1}) + \frac{1}{2}(x(t_{j-1}) + (\Delta t)p^{j-1})^T G(x(t_{j-1}) + (\Delta t)p^{j-1}).$$

Expanding and grouping the coefficients of 1, Δt , and $(\Delta t)^2$, we find that

$$q(x(t)) = f_{j-1} + f'_{j-1}\Delta t + \frac{1}{2}f''_{j-1}(\Delta t)^2, \quad \Delta t \in [0, t_j - t_{j-1}], \quad (16.73)$$

where the coefficients f_{j-1} , f'_{j-1} , and f''_{j-1} are defined by

$$\begin{aligned} f_{j-1} &\stackrel{\text{def}}{=} c^T x(t_{j-1}) + \frac{1}{2}x(t_{j-1})^T Gx(t_{j-1}), \\ f'_{j-1} &\stackrel{\text{def}}{=} c^T p^{j-1} + x(t_{j-1})^T Gp^{j-1}, \\ f''_{j-1} &\stackrel{\text{def}}{=} (p^{j-1})^T Gp^{j-1}. \end{aligned}$$

Differentiating (16.73) with respect to Δt and equating to zero, we obtain $\Delta t^* = -f'_{j-1}/f''_{j-1}$. The following cases can occur. (i) If $f'_{j-1} > 0$ there is a local minimizer of $q(x(t))$ at $t = t_{j-1}$; else (ii) $\Delta t^* \in [0, t_j - t_{j-1}]$ there is a minimizer at $t = t_{j-1} + \Delta t^*$; (iii) in all other cases we move on to the next interval $[t_j, t_{j+1}]$ and continue the search.

For the next search interval, we need to calculate the new direction p^j from (16.72), and we use this new value to calculate f_j , f'_j , and f''_j . Since p^j differs from p^{j-1} typically in just one component, computational savings can be made by updating these coefficients rather than computing them from scratch.

SUBSPACE MINIMIZATION

After the Cauchy point x^c has been computed, the components of x^c that are at their lower or upper bounds define the active set

$$\mathcal{A}(x^c) = \{i \mid x_i^c = l_i \text{ or } x_i^c = u_i\}.$$

In the second stage of the gradient projection iteration, we approximately solve the QP obtained by fixing the components x_i for $i \in \mathcal{A}(x^c)$ at the values x_i^c . The remaining

components are determined from the subproblem

$$\min_x q(x) = \frac{1}{2}x^T Gx + x^T c \quad (16.74a)$$

$$\text{subject to } x_i = x_i^c, \quad i \in \mathcal{A}(x^c), \quad (16.74b)$$

$$l_i \leq x_i \leq u_i, \quad i \notin \mathcal{A}(x^c). \quad (16.74c)$$

It is not necessary to solve this problem exactly. Nor is it desirable in the large-dimensional case, because the subproblem may be almost as difficult as the original problem (16.68). In fact, to obtain global convergence of the gradient projection procedure, we require only that the approximate solution x^+ of (16.74) is feasible with respect to (16.68b) and has an objective function value no worse than that of x^c , that is, $q(x^+) \leq q(x^c)$. A strategy that is intermediate between choosing $x^+ = x^c$ as the approximate solution (on the one hand) and solving (16.74) exactly (on the other hand) is to compute an approximate solution of (16.74) by using the conjugate gradient iteration described in Algorithm 16.1 or Algorithm 16.2. Note that for the equality constraints (16.74b), the Jacobian A and the null-space basis matrix Z have particularly simple forms. We could therefore apply conjugate gradient to the problem (16.74a), (16.74b) and terminate as soon as a bound $l \leq x \leq u$ is encountered. Alternatively, we could continue to iterate, temporarily ignoring the bounds and projecting the solution back onto the box constraints. The negative-curvature case can be handled as in Algorithm 7.2, the method for approximately solving possibly indefinite trust-region subproblems in unconstrained optimization.

We summarize the gradient projection algorithm for quadratic programming as follows.

Algorithm 16.5 (Gradient Projection Method for QP).

Compute a feasible starting point x^0 ;

for $k = 0, 1, 2, \dots$

if x^k satisfies the KKT conditions for (16.68)

stop with solution $x^* = x^k$;

 Set $x = x^k$ and find the Cauchy point x^c ;

 Find an approximate solution x^+ of (16.74) such that $q(x^+) \leq q(x^c)$

 and x^+ is feasible;

$x^{k+1} \leftarrow x^+$;

end (for)

If the algorithm approaches a solution x^* at which the Lagrange multipliers associated with all the active bounds are nonzero (that is, strict complementarity holds), the active sets $\mathcal{A}(x^c)$ generated by the gradient projection algorithm are equal to the optimal active set for all k sufficiently large. That is, constraint indices do not repeatedly enter and leave the active set on successive iterations. When the problem is degenerate, the active set may not settle down at its optimal value. Various devices have been proposed to prevent this undesirable behavior from taking place.

While gradient projection methods can be applied in principle to problems with general linear constraints, significant computation may be required to perform the projection onto the feasible set in such cases. For example, if the constraint set is defined as $a_i^T x \geq b_i$, $i \in \mathcal{I}$, we must solve the following convex quadratic program to compute the projection of a given point \bar{x} onto this set:

$$\max_x \|x - \bar{x}\|^2 \quad \text{subject to } a_i^T x \geq b_i \text{ for all } i \in \mathcal{I}.$$

The expense of solving this “projection subproblem” may approach the cost of solving the original quadratic program, so it is usually not economical to apply gradient projection to this case.

When we use duality to replace a strictly convex quadratic program with its dual (see Example 12.12), the gradient projection method may be useful in solving the bound-constrained dual problem, which is formulated in terms of the Lagrange multipliers λ as follows:

$$\max_{\lambda} \tilde{q}(\lambda) = -\frac{1}{2}(A^T \lambda - c)^T G^{-1}(A^T \lambda - c) + b^T \lambda, \quad \text{subject to } \lambda \geq 0.$$

(Note that the dual is conventionally written as a maximization problem; we can equivalently minimize $-\tilde{q}(\lambda)$ and note that this transformed problem is convex.) This approach is most useful when G has a simple form, for example, a diagonal or block-diagonal matrix.

16.8 PERSPECTIVES AND SOFTWARE

Active-set methods for convex quadratic programming are implemented in QPOPT [126], VE09 [142], BQPD [103], and QPA [148]. Several commercial interior-point solvers for QP are available, including CPLEX [172], XPRESS-MP [159] and MOSEK [5]. The code QPB [146] uses a two-phase interior-point method that can handle convex and nonconvex problems. OOPS [139] and OOQP [121] are object-oriented interior-point codes that allow the user to customize the linear algebra techniques to the particular structure of the data for an application. Some nonlinear programming interior-point packages, such as LOQO [294] and KNITRO [46], are also effective for convex and nonconvex quadratic programming.

The numerical comparison of active-set and interior-point methods for convex quadratic programming reported in [149] indicates that interior-point methods are generally much faster on large problems. If a warm start is required, however, active-set methods may be generally preferable. Although considerable research has been focused on improving the warm-start capabilities of interior-point methods, the full potential of such techniques is now yet known.

We have assumed in this chapter that all equality-constrained quadratic programs have linearly independent constraints, that is, the $m \times n$ constraint Jacobian matrix A has rank m .

If redundant constraints are present, they can be detected by forming a SVD or rank-revealing QR factorization of A^T , and then removed from the formulation. When A is larger, sparse Gaussian elimination techniques can be applied to A^T instead, but they are less reliable.

The KNITRO and OOPS software packages provide the option of solving the primal-dual equations (16.63) by means of the projected CG iteration of Algorithm 16.2.

We have not considered active-set methods for the case in which the Hessian matrix G is indefinite because these methods can be quite complicated to describe and it is not well understood how to adapt them to the large dimensional case. We make some comments here on the principal techniques.

Algorithm 16.3, 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 (16.53). If the reduced Hessian $Z^T G Z$ is positive definite, then this step p points to the minimizer of the subproblem (16.39), and the logic of the iteration need not be changed. If $Z^T G Z$ has negative eigenvalues, however, p points only to a saddle point of (16.39) and is therefore not always a suitable step. Instead, we seek an alternative direction s_z that is a direction of *negative curvature* for $Z^T G Z$. We then have that

$$q(x + \alpha Zs_z) \rightarrow -\infty \quad \text{as } \alpha \rightarrow \infty. \quad (16.75)$$

Additionally, we change the sign of s_z if necessary to ensure that Zs_z is a non-ascent direction for q at the current point x , that is, $\nabla q(x)^T Zs_z \leq 0$. By moving along the direction Zs_z , we will encounter a constraint that can 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 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, 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. 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. 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.

Various algorithms for indefinite QP differ in the way that indefiniteness is detected, in the computation of the negative curvature direction, and in the handling of the working set; see Fletcher [99] and Gill and Murray [126].

NOTES AND REFERENCES

The problem of determining whether a feasible point for a nonconvex QP (16.1) is a global minimizer is NP-hard (Murty and Kabadi [219]); so is the problem of determining whether a given point is a local minimizer (Vavasis [296, Theorem 5.1]). Various algorithms for convex QP with polynomial convexity are discussed in Nesterov and Nemirovskii [226].

The portfolio optimization problem was formulated by Markowitz [201].

For a discussion on the QMR, LSQR, and GMRES methods see, for example, [136, 272, 290]. The idea of using the projection (16.30) in the CG method dates back to at least Polyak [238]. The alternative (16.34), and its special case (16.32), are proposed in Coleman [64]. Although it can give rise to substantial rounding errors, they can be corrected by iterative refinement; see Gould et al. [143]. More recent studies on preconditioning of the projected CG method include Keller et al. [176] and Lukšan and Vlček [196].

For further discussion on the gradient projection method see, for example, Conn, Gould, and Toint [70] and Burke and Moré [44].

In some areas of application, the KKT matrix (16.7) not only is sparse but also contains special structure. For instance, the quadratic programs that arise in many control problems have banded matrices G and A (see Wright [315]), which can be exploited by interior-point methods via a suitable symmetric reordering of K . When active-set methods are applied to this problem, however, the advantages of bandedness and sparsity are lost after just a few updates of the factorization.

Further details of interior-point methods for convex quadratic programming can be found in Wright [316] and Vanderbei [293]. The first inertia-controlling method for indefinite quadratic programming was proposed by Fletcher [99]. See also Gill et al. [129] and Gould [142] for a discussion of methods for general quadratic programming.



EXERCISES




16.1

- (a) Solve the following quadratic program and illustrate it geometrically.

$$\begin{aligned} \min f(x) &= 2x_1 + 3x_2 + 4x_1^2 + 2x_1x_2 + x_2^2, \\ \text{subject to } x_1 - x_2 &\geq 0, \quad x_1 + x_2 \leq 4, \quad x_1 \leq 3. \end{aligned}$$

- (b) If the objective function is redefined as $q(x) = -f(x)$, does the problem have a finite minimum? Are there local minimizers?

 **16.2** The problem of finding the shortest distance from a point x_0 to the hyperplane $\{x \mid Ax = b\}$, where A has full row rank, can be formulated as the quadratic program

$$\min \frac{1}{2}(x - x_0)^T(x - x_0) \text{ subject to } Ax = b.$$


Show that the optimal multiplier is


$$\lambda^* = (AA^T)^{-1}(b - Ax_0)$$


and that the solution is


$$x^* = x_0 + A^T(AA^T)^{-1}(b - Ax_0).$$


Show that in the special case in which A is a row vector, the shortest distance from x_0 to the solution set of $Ax = b$ is $|b - Ax_0|/\|A\|_2$.


 **16.3** Use Theorem 12.1 to verify that the first-order necessary conditions for (16.3) are given by (16.4).


 **16.4** Suppose that G is positive semidefinite in (16.1) and that x^* satisfies the KKT conditions (16.37) for some $\lambda_i^*, i \in \mathcal{A}(x^*)$. Suppose in addition that second-order sufficient conditions are satisfied, that is, $Z^T G Z$ is positive definite where the columns of Z span the null space of the active constraint Jacobian matrix. Show that x^* is in fact the unique global solution for (16.1), that is, $q(x) > q(x^*)$ for all feasible x with $x \neq x^*$.


 **16.5** Verify that the inverse of the KKT matrix is given by (16.16).


 **16.6** Use Theorem 12.6 to show that if the conditions of Lemma 16.1 hold, then the second-order sufficient conditions for (16.3) are satisfied by the vector pair (x^*, λ^*) that satisfies (16.4).

 **16.7** Consider (16.3) and suppose that the projected Hessian matrix $Z^T G Z$ has a negative eigenvalue; that is, $u^T Z^T G Z u < 0$ for some vector u . Show that if there exists any vector pair (x^*, λ^*) that satisfies (16.4), then the point x^* is only a stationary point of (16.3) and not a local minimizer. (Hint: Consider the function $q(x^* + \alpha Z u)$ for $\alpha \neq 0$, and use an expansion like that in the proof of Theorem 16.2.)

 **16.8** By using the QR factorization and a permutation matrix, show that for a full-rank $m \times n$ matrix A (with $m < n$) one can find an orthogonal matrix Q and an $m \times m$ upper triangular matrix \hat{U} such that $AQ = \begin{bmatrix} 0 & \hat{U} \end{bmatrix}$. (Hint: Start by applying the standard QR factorization to A^T .)


 **16.9** Verify that the first-order conditions for optimality of (16.1) are equivalent to (16.37) when we make use of the active-set definition (16.36).


 **16.10** For each of the alternative choices of initial working set \mathcal{W}_0 in the example (16.49) (that is, $\mathcal{W}_0 = \{3\}$, $\mathcal{W}_0 = \{5\}$, and $\mathcal{W}_0 = \emptyset$) work through the first two iterations of Algorithm 16.3.

 **16.11** Program Algorithm 16.3, and use it to solve the problem


$$\begin{aligned} \min \quad & x_1^2 + 2x_2^2 - 2x_1 - 6x_2 - 2x_1x_2 \\ \text{subject to} \quad & \frac{1}{2}x_1 + \frac{1}{2}x_2 \leq 1, \quad -x_1 + 2x_2 \leq 2, \quad x_1, x_2 \geq 0. \end{aligned}$$

Choose three initial starting points: one in the interior of the feasible region, one at a vertex, and one at a non-vertex point on the boundary of the feasible region.


 **16.12** Show that the operator P defined by (16.27) is independent of the choice of null-space basis Z . (Hint: First show that any null-space basis Z can be written as $Z = QB$ where Q is an orthogonal basis and B is a nonsingular matrix.)


 **16.13**

- (a) Show that the the computation of the preconditioned residual g^+ in (16.28d) can be performed with (16.29) or (16.30).
- (b) Show that we can also perform this computation by solving the system (16.32).
- (c) Verify (16.33).


 **16.14**

- (a) Show that if $Z^T G Z$ is positive definite, then the denominator in (16.28a) is nonzero.
- (b) Show that if $Z^T r = r_z \neq 0$ and $Z^T H Z$ is positive definite, then the denominator in (16.28e) is nonzero.

 **16.15** Consider problem (16.3), and assume that A has full row rank and that Z is a basis for the null space of A . Prove that there are no finite solutions if $Z^T G Z$ has negative eigenvalues.


 **16.16**


- (a) Assume that $A \neq 0$. Show that the KKT matrix (16.7) is indefinite.
- (b) Prove that if the KKT matrix (16.7) is nonsingular, then A must have full rank.


 **16.17** Consider the quadratic program


$$\begin{aligned} \max \quad & 6x_1 + 4x_2 - 13 - x_1^2 - x_2^2, \\ \text{subject to} \quad & x_1 + x_2 \leq 3, \quad x_1 \geq 0, \quad x_2 \geq 0. \end{aligned} \tag{16.76}$$

First solve it graphically, and then use your program implementing the active-set method given in Algorithm 16.3.

 **16.18** Using (16.39) and (16.41), explain briefly why the gradient of each blocking constraint cannot be a linear combination of the constraint gradients in the current working set \mathcal{W}_k .


 **16.19** Let W be an $n \times n$ symmetric matrix, and suppose that Z is of dimension $n \times t$. Suppose that $Z^T W Z$ is positive definite and that \bar{Z} is obtained by removing a column from Z . Show that $\bar{Z}^T W \bar{Z}$ is positive definite.


 **16.20** Find a null-space basis matrix Z for the equality-constrained problem defined by (16.74a), (16.74b).

 **16.21** Write down KKT conditions for the following convex quadratic program with mixed equality and inequality constraints:

$$\min q(x) = \frac{1}{2}x^T Gx + x^T c \quad \text{subject to} \quad Ax \geq b, \quad \bar{A}x = \bar{b},$$


where G is symmetric and positive semidefinite. Use these conditions to derive an analogue of the generic primal-dual step (16.58) for this problem.


 **16.22** Explain why for a bound-constrained problems the number of possible active sets is at most 3^n .

 **16.23**

(a) Show that the primal-dual system (16.58) can be solved using the augmented system (16.61) or the normal equations (16.62). Describe in detail how all the components $(\Delta x, \Delta y, \Delta \lambda)$ are computed.

(b) Verify (16.65).

 **16.24** Program Algorithm 16.4 and use it to solve problem (16.76). Set all initial variables to be the vector $e = (1, 1, \dots, 1)^T$.

 **16.25** Let $\bar{x} \in R^n$ be given, and let x^* be the solution of the projection problem


$$\min \|x - \bar{x}\|^2 \quad \text{subject to} \quad l \leq x \leq u. \quad (16.77)$$

For simplicity, assume that $-\infty < l_i < u_i < \infty$ for all $i = 1, 2, \dots, n$. Show that the solution of this problem coincides with the projection formula given by (16.69) that is, show that $x^* = P(\bar{x}, l, u)$. (*Hint*: Note that the problem is separable.)

 **16.26** Consider the bound-constrained quadratic problem (16.68) with

$$G = \begin{bmatrix} 4 & 1 \\ 1 & 2 \end{bmatrix}, \quad c = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad l = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \text{and} \quad u = \begin{bmatrix} 5 \\ 3 \end{bmatrix}. \quad (16.78)$$

Suppose $x^0 = (0, 2)^T$. Find $\bar{t}_1, \bar{t}_2, t_1, t_2, p^1, p^2$ and $x(t_1), x(t_2)$. Find the minimizer of $q(x(t))$.

 **16.27** Consider the search for the one dimensional minimizer of the function $q(x(t))$ defined by (16.73). There are 9 possible cases since f, f', f'' can each be positive, negative, or zero. For each case, determine the location of the minimizer. Verify that the rules described in Section 16.7 hold.