

Chapter 9

Quadratic programming

The material of this chapter is mostly contained in Chapter 16 of Nocedal & Wright [26].

Some of the most popular general-purpose methods for solving a general *constrained optimization problem*

$$\begin{aligned} \min_{\mathbf{x} \in \Omega} \quad & f(\mathbf{x}) \\ \Omega = \{ \mathbf{x} \in \mathbb{R}^n \mid & c_i(\mathbf{x}) = 0, \quad i \in \mathcal{E}, \quad c_i(\mathbf{x}) \geq 0, \quad i \in \mathcal{I} \} \end{aligned}$$

are based on *sequential quadratic programming* (SQP). At each iteration k of such methods, a *quadratic programming* (QP) problem is solved. So, we'd better be able to solve QP problems well and rapidly.

Let us write the problem in the form

$$\min \quad f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T H \mathbf{x} - \mathbf{d}^T \mathbf{x} \quad (9.1a)$$

$$s.t. \quad b_i - \mathbf{a}_i^T \mathbf{x} = 0, \quad i \in \mathcal{E} \quad (9.1b)$$

$$b_i - \mathbf{a}_i^T \mathbf{x} \geq 0, \quad i \in \mathcal{I} \quad (9.1c)$$

- If there are only equality constraints in (9.1) then the KKT conditions yield a linear system of equations, as we have seen in Example 7.3. There are simple conditions that guarantee a unique solution of this linear system. The obtained critical point is a minimum point if an additional condition is satisfied. In summary, if the constraint matrix A has a full row rank, and H is a symmetric matrix satisfying

$$\mathbf{y}^T H \mathbf{y} > 0, \quad \forall \mathbf{y} \in \ker(A), \quad (9.2)$$

then there is a unique minimum which is obtained as a solution for a linear system of equations. This case is further considered in Section 9.1.

- In the more general case where there are also inequality constraints, if H is positive semi-definite then the QP is *convex*. We derive an active set method for this case in Section 9.2.
- A different method, which works particularly well for the case of non-negativity constraints, is considered in Section 9.3. It extends also for nonconvex QP.
- In the general, nonconvex case, there can be several minima and other stationary points. We won't consider this case further here.

What about an interior point method? It can certainly be derived, extending the algorithm for the LP case from Section 8.2. My resulting program turns out to be less robust than for the LP case, but it often works very well. Because of the similarity to the LP method, however, the derivation will not be repeated. See Wright [35].

Example 9.1

The QP problem

$$\min_{\mathbf{x}} \quad \frac{1}{2} \mathbf{x}^T M \mathbf{x} \quad (9.3a)$$

$$s.t. \quad \mathbf{q} + M \mathbf{x} \geq \mathbf{0} \quad (9.3b)$$

is convex if M is symmetric positive semi-definite.

The Lagrangian is $\mathcal{L} = \frac{1}{2} \mathbf{x}^T M \mathbf{x} - \boldsymbol{\lambda}^T (\mathbf{q} + M \mathbf{x})$ and $\nabla_{\mathbf{x}} \mathcal{L} = \mathbf{0}$ if $\boldsymbol{\lambda} = \mathbf{x}$. The KKT conditions can therefore be written as

$$-M \mathbf{x} + \mathbf{s} = \mathbf{q} \quad (9.4a)$$

$$X S \mathbf{e} = \mathbf{0} \quad (9.4b)$$

$$\mathbf{x} \geq \mathbf{0}, \mathbf{s} \geq \mathbf{0}. \quad (9.4c)$$

This is the famous Linear Complementarity Problem (LCP). ♦

9.1 Equality constraints and KKT systems

In this section we consider the problem

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T H \mathbf{x} - \mathbf{d}^T \mathbf{x} \quad (9.5)$$

$$A \mathbf{x} = \mathbf{b}$$

where A is $m \times n$ with a full row rank m , and H is a symmetric matrix satisfying (9.2). Under these assumptions the KKT conditions

$$\begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{d} \\ \mathbf{b} \end{pmatrix} \quad (9.6)$$

yield a unique solution which is the minimum of the QP (9.5).

Recall from Section 7.3 that it makes sense perhaps to consider elimination of some variables in terms of the others. In particular, recall the matrices Z and Y introduced in (7.17) and the reduced Hessian H_r defined in (7.19). The condition (9.2) for a minimum can be written as requiring that H_r be symmetric positive definite (s.p.d.). Under these assumptions, then, all that is left to do is discuss solution methods for the linear system (9.6).

The matrix

$$K = \begin{pmatrix} H & A^T \\ A & 0 \end{pmatrix} \quad (9.7)$$

is symmetric *indefinite*: It has n positive eigenvalues and m negative ones. Indeed, the critical point (i.e. the solution of (9.6)) is a saddle point for the Lagrangian: a minimum point in the original (primal) variables \mathbf{x} and a maximum point for the Lagrange multipliers (or, the dual variables) $\boldsymbol{\lambda}$.

Straight ahead

If n is not very large (say $n \leq 100$) and/or no other special features present themselves, then just solve (9.6) directly, using some variant of LU -decomposition. Note, though, that a Choleski decomposition may not be used, because K of (9.7) is not s.p.d.

Range-space method

If H is nonsingular and relatively easy to invert, then we can apply the following Schur decomposition. From the first block equations in (9.6), $H\mathbf{x} + A^T\boldsymbol{\lambda} = \mathbf{d} \Rightarrow \mathbf{x} = H^{-1}(\mathbf{d} - A^T\boldsymbol{\lambda})$. Then the bottom block equations yield

$$(AH^{-1}A^T)\boldsymbol{\lambda} = AH^{-1}\mathbf{d} - \mathbf{b}. \quad (9.8a)$$

Eliminating $\boldsymbol{\lambda}$ by (9.8a) we obtain

$$\begin{aligned} \mathbf{x} &= H^{-1}(\mathbf{d} - A^T\boldsymbol{\lambda}) \\ &= (I - H^{-1}A^T(AH^{-1}A^T)^{-1}A)H^{-1}\mathbf{d} + H^{-1}A^T(AH^{-1}A^T)^{-1}\mathbf{b} \end{aligned} \quad (9.8b)$$

We can write this as

$$\mathbf{x} = (I - TA)H^{-1}\mathbf{d} + T\mathbf{b}, \quad T = H^{-1}A^T(AH^{-1}A^T)^{-1}, \quad (9.8c)$$

and note that T is a projection matrix (satisfying $AT = I$). This is particularly useful when $m \ll n$, because then the matrix $AH^{-1}A^T$ is small in dimension.

Example 9.2

In rigid multibody simulations, arising routinely in robotics and virtual reality, the dynamics for a system of n coordinates with m holonomic constraints can be written as

$$\begin{aligned} M\ddot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \dot{\mathbf{x}}) - G^T(\mathbf{x})\boldsymbol{\lambda}, \\ \mathbf{g}(\mathbf{x}) &= \mathbf{0}, \end{aligned} \quad (9.9)$$

where $\mathbf{x}(t)$ are generalized positions in time t of the rigid bodies (including both locations and orientations in some coordinate system, combined into a vector function of length n), the time derivatives $\dot{\mathbf{x}}$ are the corresponding generalized velocities, and the derivatives of the generalized velocities with respect to time, $\ddot{\mathbf{x}}$, are the corresponding generalized accelerations. The $m \times n$ Jacobian matrix of the holonomic constraints is

$$G(\mathbf{x}) = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}.$$

The Lagrange multipliers $\boldsymbol{\lambda}(t)$ are, in fact, the forces applied by these constraints. Other forces are gathered in $\mathbf{f}(\mathbf{x}, \dot{\mathbf{x}})$. The matrix M , which may depend on $\mathbf{x}(t)$, is a *mass matrix*. As such it is usually s.p.d. and banded, indeed it is occasionally diagonal. Its sparsity depends on the choice of generalized coordinates, but even in relative coordinates (which cause M to be smaller and chubbier) it can be inverted rapidly.

The system (9.9) expresses Newton's second law of mechanics: masses times accelerations equal forces. For the gory details you may consult our home-made efforts [27, 4], although there are zillions of other references.

The system (9.9) is a differential-algebraic equation (DAE) (see, e.g., [5]). One wants to integrate it in time t for the generalized positions of the bodies, $\mathbf{x}(t)$. However, there are some difficulties in discretizing this DAE directly. Instead we can apply two time differentiations to the constraints (using for instance *automatic differentiation* techniques, see e.g. [26]), and obtain the system

$$\begin{pmatrix} M & G^T \\ G & 0 \end{pmatrix} \begin{pmatrix} \ddot{\mathbf{x}} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{d} \\ \mathbf{b} \end{pmatrix}, \quad (9.10)$$

where \mathbf{b} and \mathbf{d} depend on \mathbf{x} and on its first derivative $\dot{\mathbf{x}}$ (and thus on time t as well), but not directly on $\ddot{\mathbf{x}}$ and $\boldsymbol{\lambda}$.

Using the elimination technique (9.8) we now can eliminate $\boldsymbol{\lambda}(t)$ from (9.10) and obtain an ordinary differential equation (ODE) for $\mathbf{x}(t)$. This ODE is more readily integrated numerically in time. Some stabilization with respect to the constraints must be applied in general, see e.g. [5] for details. This yields a very viable practical method.

Note that in this case, indeed typically $H = M$ is easy to invert, and often $m \ll n$ as well.



Example 9.3

The Navier–Stokes equations modeling the flow of an incompressible fluid in two space dimensions and time are written as

$$u_t + uu_x + vu_y + p_x - \nu(u_{xx} + u_{yy}) = 0, \quad (9.11a)$$

$$v_t + uv_x + vv_y + p_y - \nu(v_{xx} + v_{yy}) = 0, \quad (9.11b)$$

$$u_x + v_y = 0, \quad (9.11c)$$

where the subscripts denote partial derivatives, $u(x, y, t)$ and $v(x, y, t)$ are the x - and y - components of the velocity field, $p = p(x, y, t)$ is the pressure (it's a Lagrange multiplier function!), and ν is a known positive viscosity constant.

For simplicity, consider the steady state case where the time derivatives (and dependency) drop and assume that the nonlinear terms in (9.11a) and (9.11b) do not dominate. The resulting partial differential equation (PDE) system can be written as

$$\begin{pmatrix} -\nu(\partial_{xx} + \partial_{yy}) & 0 & \partial_x \\ 0 & -\nu(\partial_{xx} + \partial_{yy}) & \partial_y \\ \partial_x & \partial_y & 0 \end{pmatrix} \begin{pmatrix} u(x, y) \\ v(x, y) \\ p(x, y) \end{pmatrix} = \begin{pmatrix} b_1(u, v) \\ b_2(u, v) \\ 0 \end{pmatrix}.$$

We have on the left a differential operator in KKT form.

The obtained set of partial differential equations is discretized by a *finite element* or a *finite volume* technique; see e.g., [15, 20, 10]. You do not need to be an expert on the numerical solution of partial differential equations in order to believe that a very large, sparse algebraic system of the form (9.6) is obtained, where H is again a mass matrix (diagonal for finite volume techniques, almost diagonal for finite elements), A is a discretization of the

div operator (∂_x, ∂_y) and A^T is a discretization of the grad operator $\begin{pmatrix} \partial_x \\ \partial_y \end{pmatrix}$.

The constraint qualification (LICQ) that A have a full row rank is now called the Babuska-Brezzi condition, and special *mixed finite element* methods may be designed to achieve this; see the clarity-challenged text [10].

The range-space method corresponds to a method for (9.11) called the *pressure-Poisson*; see, e.g. [20].



Null-space method

The range-space methods require, in particular, that H be nonsingular. For a more general case, let Z be defined as discussed in §7.3; specifically, $AZ = 0$. Then we can multiply the first block rows of (9.6) by Z^T to eliminate $\boldsymbol{\lambda}$,

$$\begin{aligned} Z^T H \mathbf{x} &= Z^T \mathbf{d}, \\ A \mathbf{x} &= \mathbf{b}. \end{aligned}$$

Assuming $AY = I$, write $\mathbf{x} = Y\mathbf{b} + Z\mathbf{x}_N$ as before, and obtain

$$H_r \mathbf{x}_N = Z^T (\mathbf{d} - HY\mathbf{b}), \quad H_r = Z^T H Z. \quad (9.12)$$

From this we may obtain \mathbf{x}_N , and then all the other unknowns may be reconstructed; specifically,

$$\mathbf{x}_B = B^{-1}(\mathbf{b} - N\mathbf{x}_N), \quad \boldsymbol{\lambda} = Y^T(\mathbf{d} - H\mathbf{x}).$$

This method is useful especially when H is singular. There are variants of this in use in multibody system simulations as described in Example 9.2.

9.1.1 Large KKT problems

If the problem is large, especially if m and $n - m$ are both large, then iterative methods are called for solving (9.6).

Range-space and null-space methods

It is customary to use one of the elimination methods described in the previous section, obtaining a large reduced problem such as (9.8) or (9.12). The reduced matrices are typically s.p.d., unlike K of (9.7). Thus, we may apply fast iterative methods such as preconditioned CG for the solution of (9.8) and (9.12), whereas we may not apply such methods for the direct, simultaneous

solution of the indefinite (9.6). Note that if we use a null-space method then the form (7.17), and not one of the more sophisticated choices, is the way to go.

For the pressure-Poisson method in Example 9.3 above, the matrix $AH^{-1}A^T$ of (9.8a) corresponds to a discretization of a well-behaved, elliptic PDE for which not only preconditioned CG, but other fast methods such as multigrid are available.

Example 9.4

Let us return once more to the distributed parameter estimation problem of Example 2.5. A typical forward problem relates u to σ through the elliptic PDE system

$$\begin{aligned} \operatorname{div}(\sigma \operatorname{grad} u) &= q, & (x, y, z) \in \mathcal{D}, \\ u|_{\partial\mathcal{D}} &= 0, \end{aligned} \quad (9.13)$$

where $q(x, y, z)$ are given sources.

Discretizing u on the same grid as that for σ using, say, a finite volume method, we get an algebraic relationship for each grid value of u . Next, we order the grid values of u , σ and q (say, lexicographically) into vectors \mathbf{u} , $\boldsymbol{\sigma}$ and \mathbf{q} , and obtain an algebraic relationship

$$\hat{A}(\boldsymbol{\sigma})\mathbf{u} = \mathbf{q}$$

where \hat{A} is a large, sparse, s.p.d. matrix. Even if \hat{A} is not exactly s.p.d. it is reasonable to assume that there are effective iterative methods for solving this system, soon to be denoted (9.14b), for $\mathbf{u} = G(\boldsymbol{\sigma})\mathbf{q}$.

But now, unlike Example 2.5, let us not rush to eliminate \mathbf{u} . We write the same optimization problem derived there as

$$\min f(\boldsymbol{\sigma}) = \frac{1}{2}\|Q\mathbf{u} - \mathbf{b}\|^2 + \frac{\beta}{2}\|W\boldsymbol{\sigma}\|^2, \quad (9.14a)$$

$$s.t. \quad \hat{A}(\boldsymbol{\sigma})\mathbf{u} = \mathbf{q}. \quad (9.14b)$$

More details are given in [22]. The objective function in the constrained optimization problem (9.14) is quadratic. Corresponding to (9.5) we have the

unknowns $\mathbf{x} \leftarrow (\boldsymbol{\sigma}, \mathbf{u})$ and the Hessian $H \leftarrow \begin{pmatrix} Q^T Q & 0 \\ 0 & \beta W^T W \end{pmatrix}$.¹ However, the constraints (9.14b) are not linear: The matrix of constraint gradients is

$$A = (\hat{A} \tilde{A}), \quad \text{where} \quad (9.15a)$$

$$\tilde{A} = \frac{\partial(\hat{A}(\boldsymbol{\sigma})\mathbf{u})}{\partial\boldsymbol{\sigma}}. \quad (9.15b)$$

¹Note that H is only positive *semi-definite* if Q is not square and nonsingular, or if $\beta = 0$.

In particular, \tilde{A} is not necessarily constant. Thus, the constrained optimization problem (9.14) is not QP.

To simplify notation, assume that secondary information is small in magnitude and approximate $\mathcal{L}_{\mathbf{xx}}(\mathbf{x}^*, \boldsymbol{\lambda}^*)$, assumed to be s.p.d., by H .² The Matrix K of (9.7) therefore becomes

$$K = \begin{pmatrix} Q^T Q & 0 & \hat{A}^T \\ 0 & \beta W^T W & \tilde{A}^T \\ \hat{A} & \tilde{A} & 0 \end{pmatrix}. \quad (9.15c)$$

Note that if the data are not available everywhere then Q cannot have a full column rank. Hence $Q^T Q$ is singular and range space methods cannot be applied.

Although (9.14b) is nonlinear, it allows a safe elimination of variables, namely, of \mathbf{u} in terms of $\boldsymbol{\sigma}$. This can be done right off the bat, obtaining an unconstrained, nonlinear least squares problem in (9.14a), or when solving a linearized system involving K of (9.15).

In the latter case, calling the variables of the sought direction vector (for

which we solve at a given iterate k) $\mathbf{p}_k = \begin{pmatrix} \delta \mathbf{u} \\ \delta \boldsymbol{\sigma} \\ \delta \boldsymbol{\lambda} \end{pmatrix}$, we can eliminate $\delta \mathbf{u}$ in

terms of $\delta \boldsymbol{\sigma}$ from the last block rows of (9.15c) and then $\delta \boldsymbol{\lambda}$ in terms of $\delta \boldsymbol{\sigma}$ from the first block rows. This is a null-space method! We obtain

$$\begin{aligned} H_r \delta \boldsymbol{\sigma} &= \mathbf{s}, \quad \text{where} & (9.16) \\ H_r &= S^T S + \beta W^T W, \quad \text{where} \\ S &= -Q \hat{A}^{-1} \tilde{A}, \end{aligned}$$

and \mathbf{s} is an appropriate right hand side vector.

The matrix H_r is certainly s.p.d. for $\beta > 0$. Precisely the same matrix as H_r of (9.16) appears in the unconstrained formulation, i.e., where (9.14b) is used to eliminate \mathbf{u} in (9.14a), assuming again the Gauss-Newton approximation. We can use preconditioned CG methods both for problems such as (9.14b) (the forward problem) and for (9.16). See, e.g., [33].

Note that when evaluating matrix-vector products involving H_r , as required by a CG method, we never form the large and *dense* matrix S . Rather, we evaluate $S\mathbf{v}$ by evaluating $\mathbf{w} = \tilde{A}\mathbf{v}$ with \tilde{A} sparse, then solve the forward problem for $\hat{A}^{-1}\mathbf{w}$, and finally multiply by the sparse Q .

◆

²This leads to a Gauss-Newton method. See [22].

Simultaneous solution of KKT equations

Suppose now that not only n but also m is large. Further, suppose that H is not particularly simple to invert as it was in Example 9.2. There is a sense of imbalance in the elimination procedures described above: why eliminate precisely certain unknowns in terms of others in the midst of some inaccurate iteration for the remaining unknowns of the linear system?? In the large, sparse context we end up with two iterative processes just for the linear system (9.6), one inside the other, where the innermost iteration is carried out to a higher degree of accuracy than the outer iteration. Instead, consider the solution of the KKT system simultaneously.

This necessitates methods for solving very large, sparse, symmetric, *indefinite* systems. Krylov-space methods for large, sparse linear systems have been a hot research topic in the 1990's; see, e.g. [29, 7]. The emerging winners for our purposes are symmetric QMR [19] and LSQR. There is really no theory to fully justify any of these methods (unlike CG for s.p.d. matrices), but a huge volume of experimental results suggests that they work well provided that they are properly preconditioned, so that the eigenvalue range of the preconditioned matrix is not too wide.

In addition to the theoretical gap for Krylov-space methods, there is another general concern in solving KKT equations inaccurately. This is somewhat similar to inexact Newton: the nonzero residuals with which we are left when terminating the *linear* iteration amount to a modification in the method for the nonlinear problem. Of course, this happens also for the reduced methods described earlier. But there, we may have a better sense of descent in a computed direction, because the problems solved for the linearized system involve s.p.d. matrices. In the KKT case we are solving approximately for a saddle point, and this yields less confidence in the approximate direction so computed. Despite this, there is a growing amount of evidence that this approach works well in practice, at least for problems involving PDEs such as optimal control involving fluid flow and distributed parameter estimation [13, 12, 8, 9, 21, 3].

Example 9.5

Continuing with Example 9.4, the big problem with (9.16) is that evaluating a matrix-vector product with H_r is not cheap: each such evaluation involves two solutions of the forward problem (9.14b) and more, and this has to be done carefully to obtain “truly conjugate” directions in the CG method. We have proposed in [21] to use rough null-space methods as preconditioners for a symmetric QMR technique. Thus, H_r (or more precisely the matrix-vector product) can be approximated by something simple when it's only a

preconditioner! The overall efficiency improvement can be impressive.³ ◆

9.2 Active set methods

For simplicity of notation, let us assume that there are only inequality constraints in (9.1) and write them as

$$\mathbf{b} - A\mathbf{x} \geq \mathbf{0}. \quad (9.17)$$

The KKT conditions read

$$H\mathbf{x}^* + A^T\boldsymbol{\lambda}^* = \mathbf{d} \quad (9.18a)$$

$$\mathbf{b} - A\mathbf{x}^* \geq \mathbf{0} \quad (9.18b)$$

$$(A\mathbf{x}^* - \mathbf{b})^T\boldsymbol{\lambda}^* = 0 \quad (9.18c)$$

$$\boldsymbol{\lambda}^* \geq \mathbf{0}. \quad (9.18d)$$

If we know which constraints are active at the critical point, i.e. if we know $\mathcal{A}(\mathbf{x}^*)$, then we can gather the active constraints into a system of equalities, denoted $\hat{A}\mathbf{x}^* = \hat{\mathbf{b}}$. For the other rows, $\lambda_i^* = 0$. Denoting the components of $\boldsymbol{\lambda}^*$ corresponding to the active rows by $\hat{\boldsymbol{\lambda}}^*$, we get from (9.18a) and (9.18b) the subsystem with equality constraints

$$\begin{aligned} H\mathbf{x}^* + \hat{A}^T\hat{\boldsymbol{\lambda}}^* &= \mathbf{d} \\ \hat{A}\mathbf{x}^* &= \hat{\mathbf{b}}. \end{aligned}$$

This can be solved by the techniques discussed in the previous section, and we are done: at the optimum we should have (9.18b) and (9.18d) holding.

But, of course we do not know the final active set $\mathcal{A}(\mathbf{x}^*)$, in general. So, we develop an *active set method*. This method can be viewed as a generalization of the Simplex method discussed in Section 8.1 for linear programming. But it is more complex (and less successful in general).

We start with a feasible \mathbf{x}_0 , which may be obtained for instance from a corresponding LP solver. At the feasible iterate \mathbf{x}_k we maintain a *working set* $\mathcal{W}_k \subset \mathcal{A}(\mathbf{x}_k)$ of constraints which are active at \mathbf{x}_k such that their gradients are linearly independent.

³However, note that using a Gauss-Newton method with a null-space approach does provide a better guarantee that if a critical point is found then it is a minimum.

1. At first, we check whether \mathbf{x}_k is optimal in the subspace defined by \mathcal{W}_k . Note that, as a function of \mathbf{p} ,

$$\begin{aligned} f(\mathbf{x}_k + \mathbf{p}) &= \frac{1}{2}(\mathbf{x}_k + \mathbf{p})^T H(\mathbf{x}_k + \mathbf{p}) - \mathbf{d}^T(\mathbf{x}_k + \mathbf{p}) \\ &= \frac{1}{2}\mathbf{p}^T H\mathbf{p} + \mathbf{p}^T(H\mathbf{x}_k - \mathbf{d}) + \text{const.} \end{aligned}$$

Thus, we ask if we can take a small step in the direction \mathbf{p} defined by the quadratic program with equalities

$$\min_{\mathbf{p}} \quad \frac{1}{2}\mathbf{p}^T H\mathbf{p} + \mathbf{g}_k^T \mathbf{p} \quad (9.19a)$$

$$s.t. \quad A_k^T \mathbf{p} = 0, \quad (9.19b)$$

where $\mathbf{g}_k = H\mathbf{x}_k - \mathbf{d}$ and A_k is composed of those rows of A which are in the working set.

The solution of the subproblem (9.19) is of course given by the solution of the KKT system

$$\begin{pmatrix} H & A_k^T \\ A_k & 0 \end{pmatrix} \begin{pmatrix} \mathbf{p} \\ \hat{\boldsymbol{\lambda}} \end{pmatrix} = \begin{pmatrix} \mathbf{d} - H\mathbf{x}_k \\ \mathbf{0} \end{pmatrix}. \quad (9.20)$$

- (a) If the solution of (9.19) is $\mathbf{p} = \mathbf{0}$ then \mathbf{x}_k is optimal in the current working subspace. Proceed to stage 2 below.
- (b) Otherwise, consider setting

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p}, \quad 0 < \alpha \leq 1. \quad (9.21)$$

The step size α must be chosen to maintain feasibility. For $i \in \mathcal{W}_k$ any α will maintain feasibility because of (9.19b), but for the other rows $i \notin \mathcal{W}_k$ we must ensure $b_i - \mathbf{a}_i^T(\mathbf{x}_k + \alpha \mathbf{p}) \geq 0$. So define (as in the Simplex method for LP)

$$\alpha = \min \left(1, \min_{\mathbf{a}_i^T \mathbf{p} > 0} \frac{b_i - \mathbf{a}_i^T \mathbf{x}_k}{\mathbf{a}_i^T \mathbf{p}} \right). \quad (9.22)$$

A constraint l which would yield $\alpha < 1$ in (9.22) is a *blocking constraint*. Add the blocking constraint to the working set \mathcal{W}_k to form \mathcal{W}_{k+1} and update the iterate by (9.21).

Otherwise, $\alpha = 1$ in (9.22). We update by (9.21) but do not change the working set, $\mathcal{W}_{k+1} = \mathcal{W}_k$.

2. Now that an optimal \mathbf{x}_k has been found for \mathcal{W}_k , we check optimality: We set $\lambda_i^k = 0$ for $i \notin \mathcal{W}_k$; the other Lagrange multipliers at this point are known from having solved (9.20).
- (a) If $\hat{\boldsymbol{\lambda}} \geq \mathbf{0}$ then all KKT conditions hold and the optimum point has been found.
 - (b) Otherwise, there is a component $\lambda_q < 0$. Then we can decrease $f(\mathbf{x})$ further by dropping q from the working set. Thus, \mathcal{W}_{k+1} is formed by dropping q from \mathcal{W}_k , and the iteration is repeated.

```
function [x,k] = qp(fqp,x0)
%
% function [x,k] = qp(fqp,x0)
%
% This is a pocket version of a quadratic
% programming solver. Problem is small enough
% for direct solution of kkt, and no degeneracies
% entertained.
%
% The problem solved is   min_x f(x) = .5x^T H x - d^T x
%                       s.t.  A_1 x = b_1
%                       A_2 x <= b_2
% These details are specified in function fqp
%
% Note x0 is a FEASIBLE starting point

% initialize

% define problem: m1 and m2 are numbers or eqs and ineqs

[H,d,m1,A1,b1,m2,A2,b2] = feval(fqp);

nmax = 200; % max number of iterations
tol = 1.e-6; % tolerance

x = x0;
n = size(x);
m = 0; % number of rows in W (only active inequalities)

for k=1:nmax

% construct KKT system for direction p
K = H;
```

```

r = d - H*x;
if m1 > 0
    % add equality constraints
    A = A1;
end
if m > 0
    % add active inequality constraints
    for j=1:m
        i = W(j);
        if j+m1 == 1
            A = A2(i,1:n);
        else
            A = [A;A2(i,1:n)];
        end
    end
end
mm1 = m1 + m;
if mm1 > 0
    K = [K, A'; A, zeros(mm1,mm1)];
    r = [r; zeros(mm1,1)];
end

y = K \ r;
p = y(1:n);

if norm(p) < tol

    % check for optimality or add to work set

    if m == 0
        x = x + p
        return % successfully
    else
        lam = y(n+m1+1:n+m1+m);
        [lmin,arg] = min(lam);
        if lmin >= 0
            x = x + p;
            return % successfully
        else

            % remove constraint from W

```

```

        m = m - 1;
        for j=arg:m
            W(j) = W(j+1);
        end

    end
end

else % x is not the minimizer in W

    count = m2+1;
    val(1:count) = 1.1;
    for j=1:m2
        if A2(j,1:n) * p > max(tol^2,1.e-15)
            val(j) = (b2(j) - A2(j,1:n) * x) / (A2(j,1:n) * p);
        end
    end
    val(count) = 1;

    [alpha,ind] = min(val); % this is the allowed step size

    x = x + alpha * p;

    if ind < count % there are blocking constraints, add one to W
        m = m + 1;
        W(m) = ind;
    end

end

end
end

```

Example 9.6

Let us return to Example 7.4, and apply our program to it.

Starting from $\mathbf{x}_0 = \mathbf{0}$, at first \mathcal{W}_0 is empty. The supproblem (9.20) yields the unconstrained minimum, $\mathbf{x}_0 + \mathbf{p}_0 = \mathbf{p}_0 = (3/2, 1/8)^T$, which is infeasible. So, in (9.22) a value $\alpha < 1$ is obtained. The blocking constraint is #1. Then $\mathbf{x}_1 = 8/13\mathbf{p}_0 = (12/13, 1/13)^T$, $\mathcal{W}_1 = \{1\}$.

Next, a nonzero direction $\mathbf{p} = \gamma (1, -1)^T$ is found in the current working set (we can lower the objective value by slipping southeast along the active constraint). In (9.22) a value $\alpha < 1$ is obtained, and the new blocking

constraint in #2. Then $\mathbf{x}_2 = (1, 0)^T$, $\mathcal{W}_2 = \{1, 2\}$.

In \mathcal{W}_2 the solution of (9.19) yields $\mathbf{p} = \mathbf{0}$, so \mathbf{x}_2 is optimal in this subspace. We next check the Lagrange multipliers and find $\hat{\boldsymbol{\lambda}} = (5/8, 3/8)^T$. Thus, all components of $\boldsymbol{\lambda}$ are found to be nonnegative. Hence \mathbf{x}_2 is the global minimizer. ◆

Example 9.7

Recall the total variation (TV) regularization of Example 5.3. In particular, recall the minimization of $f_1(\mathbf{u})$ in (5.41), where ε was introduced to avoid division by 0.

instead of this expression we could consider the QP

$$\begin{aligned} \min_{\mathbf{u}} \quad & \frac{h}{2} \sum_{i=1}^N (u_i - b_i^o)^2 + \beta h \sum_{i=1}^{N-1} v_i \\ \text{s.t.} \quad & u_i - u_{i+1} + v_i \geq 0 \\ & -u_i + u_{i+1} + v_i \geq 0, \quad i = 1, \dots, N-1. \end{aligned}$$

Here we have $n = 2N - 1$ unknowns and $m = 2N - 2$ constraints.

Connecting to the notation (9.1), (9.17), we have only inequality constraints and

$$H = \begin{pmatrix} hI & 0 \\ 0 & 0 \end{pmatrix}, \quad \mathbf{d} = \begin{pmatrix} \mathbf{b}^o \\ -\beta \mathbf{e} \end{pmatrix}, \quad A = \begin{pmatrix} \sqrt{h}W & -I \\ -\sqrt{h}W & -I \end{pmatrix}.$$

Note that H is only semi-definite. We must ensure that the KKT systems we solve are nonsingular, viz., that at any point \mathbf{x}_k with working set \mathcal{W}_k the reduced Hessian $Z_k^T H Z_k$ is symmetric positive definite, where Z_k spans the null-space of A_k .

This translates to requiring that any working set have at least N rows with either $u_i - u_{i+1} + v_i = 0$ or $-u_i + u_{i+1} + v_i = 0$ holding for each i . (In other words, $v_i = |u_{i+1} - u_i|$.) So, with a simple reformulation trick we have turned a problem which was not very difficult to solve (using lagged diffusivity, to a rough tolerance) into a monster! The number of iterations required here grows (at least linearly) with N , which it does not in Example 5.3, and the linear systems to be solved are larger and indefinite.

The moral is that just because we can reformulate a problem in a particular way does not necessarily mean we should. ◆

9.3 Gradient projection for nonnegativity constraints

The active set method of the previous section can require many iterations. For instance, suppose the initial iterate \mathbf{x}_0 is strictly in the interior of Ω , i.e., \mathcal{W}_0 is empty, and that in the final active set $\mathcal{A}(\mathbf{x}^*)$ there are N constraints (e.g., $N = \mathcal{O}(n)$). Since at each iteration only one constraint may be added the number of iterations is bounded *below* by N .

We therefore want a method that is capable of making several constraints active during one iteration. Such is the gradient projection method which we consider first for nonnegativity constraints only. Thus, the problem under consideration is

$$\min_{\mathbf{x}} \quad f(\mathbf{x}), \quad (9.23a)$$

$$s.t. \quad \mathbf{x} \geq \mathbf{0}. \quad (9.23b)$$

The Lagrangian is $\mathcal{L} = f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{x}$, so the KKT conditions also have a corresponding special form,

$$\nabla f(\mathbf{x}) = \boldsymbol{\lambda}, \quad (9.24a)$$

$$\boldsymbol{\lambda}^T \mathbf{x} = 0, \quad (9.24b)$$

$$\mathbf{x} \geq \mathbf{0}, \quad \boldsymbol{\lambda} \geq \mathbf{0}. \quad (9.24c)$$

The method can be defined for any general, twice differentiable objective function $f(\mathbf{x})$. For a good exposition see Kelley [24]. We will proceed with describing the general method in the basic form, and then specialize for the QP case with further improvements.

The idea of the method is simple. At a current, feasible iterate $\mathbf{x} = \mathbf{x}_k$, consider a steepest descent direction for the unconstrained problem, $\mathbf{p} = -\nabla f(\mathbf{x})$. We proceed in this direction until we hit a component of $\mathbf{x} = \mathbf{x} + \alpha \mathbf{p}$ which vanishes. Such a component is set to 0 and the search continues with the steepest descent direction thus modified:

$$\begin{aligned} \mathbf{p}_k &= -\nabla f_k; \\ \alpha_k &= \arg \min_{\alpha > 0} f(\max(\mathbf{x}_k + \alpha \mathbf{p}_k, \mathbf{0})); \\ \mathbf{x}_{k+1} &= \max(\mathbf{x}_k + \alpha_k \mathbf{p}_k, \mathbf{0}). \end{aligned} \quad (9.25)$$

Here $\max(\mathbf{y}, \mathbf{0})$ is the vector with entries $\max(y_i, 0)$.

For the weak line search determining α_k we can use the Armijo backtracking described in Section 4.1. Thus, we start with $\alpha = 1$ (why would

$\alpha \leq 1$ always be good is not crystal clear to me) and halve α if needed until a sufficient decrease in f is obtained.

The principle of the above algorithm can be extended for general inequality constraints. For any *convex* feasibility set Ω define the projection

$$P(\hat{\mathbf{x}}) = \arg \min_{\mathbf{x} \in \Omega} \|\mathbf{x} - \hat{\mathbf{x}}\|. \quad (9.26)$$

The algorithm (9.25) is then applicable with $P(\mathbf{x}_k + \alpha \mathbf{p}_k)$ replacing $\max(\mathbf{x}_k + \alpha \mathbf{p}_k, \mathbf{0})$. But the evaluation of this projection may become costly, even for general linear inequalities, and it is better to reformulate the problem first.

The most general situation for which the gradient projection algorithm is directly applied in practice is the bound-constrained problem, where the constraints are of the form

$$l_i \leq x_i \leq u_i, \quad i = 1, 2, \dots, n, \quad (9.27)$$

where l_i and u_i are specified lower and upper bounds for the i th variable. The projection is still very simple here.

To summarize the general part of this exposition let us repeat the algorithm for a projection P and line search by backtracking: With $d(\alpha, f) < 0$ the function guaranteeing “sufficient decrease” in (4.2),

$$\begin{aligned} \alpha_k &= 0.5^p, \quad p \text{ least integer s.t. } f(P(\mathbf{x}_k - \alpha \nabla f_k)) - f(\mathbf{x}_k) \leq d(\alpha_k, f); \\ \mathbf{x}_{k+1} &= P(\mathbf{x}_k - \alpha_k \nabla f_k). \end{aligned} \quad (9.28)$$

In the case that the objective function f is quadratic (in n variables) a more sophisticated line search procedure than in general may be applied. Now the function

$$q(\alpha) = f(P(\mathbf{x}_k - \alpha \nabla f_k))$$

is piecewise quadratic (in one variable). The breaks in between the pieces of q are where one of the components of $\mathbf{x}_k - \alpha \nabla f_k$ hits 0 and subsequently stays at that zero value as α is further increased. It is easy to determine the break points:

$$\bar{\tau}_i = \begin{cases} (\mathbf{x}_k)_i / (\nabla f_k)_i & \text{if } (\nabla f_k)_i > 0 \\ \infty & \text{otherwise} \end{cases}.$$

These values are then sorted into an increasing sequence,

$$0 = \tau_0 < \tau_1 < \tau_2 < \dots$$

For each i in turn, $i = 1, 2, \dots$, we examine the quadratic function $q(\alpha)$ for $\tau_{i-1} \leq \alpha \leq \tau_i$. If there is a minimum for this univariate quadratic in the current subinterval then set α_k to this value and exit line search. Otherwise, if $q'(\tau_{i-1}^+) > 0$ then set $\alpha_k = \tau_{i-1}$ and exit line search. Otherwise continue (i.e. perform this procedure for the next i).

Example 9.8

Consider minimizing

$$f(\mathbf{x}) = \frac{1}{2} [(x_1 - 2)^2 + (x_2 + 1)^2 + (x_3 + 2)^2]$$

subject to nonnegativity constraints (9.23b). The gradient for this admittedly too simple a function is

$$\nabla f(\mathbf{x}) = (x_1 - 2, x_2 + 1, x_3 + 2)^T.$$

Starting at $\mathbf{x}_0 = (1, 1, 1)^T$ we have $\nabla f_0 = (-1, 2, 3)^T$, hence $\mathbf{p}_0 = (1, -2, -3)^T$. The break points are, therefore, $\bar{\tau} = (\infty, 1/2, 1/3)$ and they are reordered into the sequence

$$\tau_0 = 0, \tau_1 = 1/3, \tau_2 = 1/2, \tau_3 = \infty.$$

On $[0, 1/3]$,

$$q(\alpha) = f((1, 1, 1)^T + \alpha(1, -2, -3)^T) = \frac{1}{2} [(-1 + \alpha)^2 + (2 - 2\alpha)^2 + (3 - 3\alpha)^2].$$

The minimum is obviously at $\alpha = 1$, which is outside the current subinterval. So we set $x_3 = 0$ and continue.

On $[1/3, 1/2]$,

$$q(\alpha) = \frac{1}{2} [(-1 + \alpha)^2 + (2 - 2\alpha)^2 + \text{const}_1].$$

The minimum is again at $\alpha = 1$ which is still outside the current subinterval. So we set $x_2 = 0$ and continue.

On $[1/2, \infty)$,

$$q(\alpha) = \frac{1}{2} [(-1 + \alpha)^2 + \text{const}_2].$$

The minimum is again at $\alpha = 1$, which is in the current subinterval. So we set $x_1 = 1 + \alpha = 2$. The resulting next point $\mathbf{x}_1 = (2, 0, 0)^T$ turns out to be optimal.

With the Armijo backtracking procedure outlined in (9.28) we would try $\alpha = 1$ first. This yields

$$f(P((2, -1, -2)^T)) = f((2, 0, 0)^T) = (1 + 4)/2 < (1 + 4 + 9)/2 = f(\mathbf{x}_0).$$

The optimum is therefore obtained in one iteration.

Of course, in general the situation is much more complex than this example would imply. But the example does demonstrate the feature that more than one active constraint is discovered and added to the active set in one iteration.



The gradient projection method as presented above is based on the steepest descent method for unconstrained minimization. As such it is generally slow to converge. For a discussion on how to speed it up in general see [24]. Here we continue with the QP case.

Thus, the method is typically supplemented by a *subspace minimization* technique. Essentially a faster method than steepest descent is used on the *inactive* set. Let us examine this for the case of nonnegativity constraints.

Denote the result of the step (9.25) by \mathbf{x}^c . The active set $\mathcal{A}(\mathbf{x}^c)$ consists of those variables which are at their bound, $x_i^c = 0$. (Avoid confusion with LP Simplex terminology here!) Consider the problem

$$\min_{\mathbf{x}} \quad f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T H \mathbf{x} - \mathbf{d}^T \mathbf{x}, \quad (9.29a)$$

$$s.t. \quad x_i = 0, \quad i \in \mathcal{A}(\mathbf{x}^c), \quad (9.29b)$$

$$x_i \geq 0, \quad i \notin \mathcal{A}(\mathbf{x}^c). \quad (9.29c)$$

At a first glance this problem does not look better than the original (9.23). But we do not have to solve this one very accurately. Indeed, \mathbf{x}^c is already acceptable as the next iterate \mathbf{x}_{k+1} if all else fails, and any \mathbf{x}^+ which is feasible for (9.29) and for which $f(\mathbf{x}^+) < f(\mathbf{x}^c)$ is an improvement over \mathbf{x}^c and can be taken as the next iterate \mathbf{x}_{k+1} instead.

Consider the subproblem without the inequality constraints, (9.29a), (9.29b). Here it is useful to have a quadratic objective function. We can solve the KKT conditions using a method from Section 9.1. Indeed, due to the special form of the constraints a null-space method suggests itself, as the matrix Z of (7.19), (9.12) has a particularly simple form (H_r is just a selection of inactive rows and corresponding columns of H).

Now, just solving this system may yield an infeasible solution (i.e., one that does not satisfy the nonnegativity constraints (9.29c)). Instead, we can apply *modified CG* iterations. Thus, preconditioned CG iterations are applied to (9.29a), directly filtered by the equality constraints. See [6, 2] for application of the same idea in the different context of cloth simulation in computer graphics. These iterations are not applied until convergence, but rather, until before a negative variable or a direction of negative curvature (the latter accommodates nonconvex QP) are encountered. The resulting current iterate then becomes \mathbf{x}_{k+1} .

With this subspace minimization technique the gradient projection method becomes a very attractive method for large QP problems! See [26] for a discussion on how to convert more general inequality constraints to nonnegativity constraints.