The Sequential Quadratic Programming Method

Roger Fletcher

May 9, 2007

1 Introduction

Sequential (or Successive) Quadratic Programming (SQP) is a technique for the solution of Nonlinear Programming (NLP) problems. It is, as we shall see, an idealized concept, permitting and indeed necessitating many variations and modifications before becoming available as part of a reliable and efficient production computer code. In this monograph we trace the evolution of the SQP method through some important special cases of nonlinear programming, up to the most general form of problem. To fully understand these developments it is important to have a thorough grasp of the underlying theoretical concepts, particularly in regard to optimality conditions. In this monograph we include a simple yet rigorous presentation of optimality conditions, which yet covers most cases of interest.

A nonlinear programming problem is the minimization of a nonlinear *objective function* $f(\mathbf{x})$, $\mathbf{x} \in \mathrm{I\!R}^n$, of n variables, subject to equation and/or inequality *constraints* involving a vector of nonlinear functions $\mathbf{c}(\mathbf{x})$. A basic statement of the problem, useful for didactic purposes is

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathrm{I\!R}^n}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & c_i(\mathbf{x}) \le 0 \quad i = 1, 2, \ldots, m. \end{array} \tag{1.1}$$

In this formulation, equation constraints must be encoded as two opposed inequality constraints, that is $c(\mathbf{x}) = 0$ is replaced by $c(\mathbf{x}) \le 0$ and $-c(\mathbf{x}) \le 0$, which is usually not convenient. Thus in practice a more detailed formulation is appropriate, admitting also equations, linear constraints and simple bounds. One way to do this is to add slack variables to the constraints, which together with simple bounds on the natural variables, gives rise to

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathrm{I\!R}^n}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & A^T\mathbf{x} = \mathbf{b} \\ & \mathbf{c}(\mathbf{x}) = 0 \\ & \mathbf{l} \le \mathbf{x} \le \mathbf{u}. \end{array} \tag{1.2}$$

in which either u_i or $-l_i$ can be set to a very large number if no bound is present. Alternatively one might have

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{l} \leq \begin{pmatrix} \mathbf{x} \\ A^T\mathbf{x} \\ \mathbf{c}(\mathbf{x}) \end{pmatrix} \leq \mathbf{u} \end{array} \tag{1.3}$$

in which the user can specify an equation by having $l_i = u_i$.

There are a number of special cases of NLP which are important in their own right, and for which there are special cases of the SQP method for their solution. These include *systems of nonlinear equations*, *unconstrained optimization*, and *linearly constrained optimization*. Understanding the theoretical and practical issues associated with these special cases is important when it comes to dealing with the general NLP problem as specified above. A common theme in all these cases is that there are *'linear' problems* which can be solved in a finite number of steps (ignoring the effects of round-off error), *nonlinear problems* which can usually only be solved approximately by iteration, and *Newton methods* for solving nonlinear problems by successive solution of linear problems, obtained by making Taylor series expansions about a current iterate. This context is underpinned by a famous theorem of Dennis and Moré [10] which states, subject to a regularity condition, that superlinear convergence occurs if and only if an iterative method is asymptotically equivalent to a Newton method. Loosely speaking, this tells us that we can only expect rapid convergence if our iterative method is closely related to a Newton method. The SQP method is one realization of a Newton method.

Generally $\mathbf{x} \in \mathbb{R}^n$ will denote the variables (unknowns) in the problem, $\mathbf{x}^*$ denotes a (local) solution, $\mathbf{x}^{(k)}$, $k = 1, 2\ldots$ are iterates in some iterative method, and $\mathbf{g}^{(k)}$ for example denotes the function $\mathbf{g}(\mathbf{x})$ evaluated at $\mathbf{x}^{(k)}$. Likewise $\mathbf{g}^*$ would denote $\mathbf{g}(\mathbf{x})$ evaluated at $\mathbf{x}^*$. It is important to recognise that solutions to problems may not exist, or on the other hand, there may exist multiple or non-unique solutions. Generally algorithms are only able to find local minima, and indeed guaranteeing to find a global (best local) minimizer is impractical for problems of any significant size. A more extensive and detailed coverage of topics in this monograph can be found for example in Fletcher [13].

2 Newton Methods and Local Optimality

In this section we trace the development of Newton methods from the simplest case of nonlinear equations, through to the general case of nonlinear programming with equations and inequalities.

2.1 Systems of n simultaneous equations in n unknowns

In this case the 'linear' problem referred to above is the well known system of linear equations

$$A^T\mathbf{x} = \mathbf{b} \tag{2.1}$$

in which A is a given $n \times n$ matrix of coefficients and $\mathbf{b}$ is a given vector of right hand sides. For all except the very largest problems, the system is readily solved by computing factors $PA^T = LU$ using elimination and partial pivoting. If A is a very sparse matrix, techniques are available to enable large systems to be solved. A less well known approach is to compute implicit factors $LPA = U$ (see for example Fletcher [14]) which can be advantageous in certain contexts. The system is *regular* when A is nonsingular in which case a unique solution exists. Otherwise there may be non-unique solutions, or more usually no solutions.

The corresponding nonlinear problem is the system of nonlinear equations

$$\mathbf{r}(\mathbf{x}) = \mathbf{0} \tag{2.2}$$

in which $\mathbf{r}(\mathbf{x})$ is a given vector of n nonlinear functions. We assume that $\mathbf{r}(\mathbf{x})$ is continuously differentiable and denote the $n \times n$ *Jacobian matrix* by $A = \boldsymbol{\nabla}\mathbf{r}^T$, that is A is the matrix whose columns are the gradients of the component functions in $\mathbf{r}$. If $\mathbf{r} = A^T\mathbf{x} - \mathbf{b}$ is in fact linear then its Jacobian matrix is A, which accounts for the use of A^T rather than the more usual A in (2.1). A Taylor series about the current iterate $\mathbf{x}^{(k)}$ gives

$$\mathbf{r}(\mathbf{x}^{(k)} + \mathbf{d}) = \mathbf{r}^{(k)} + A^{(k)T}\mathbf{d} + o(\|\mathbf{d}\|). \tag{2.3}$$

Truncating the negligible term in $\mathbf{d}$ and setting the left hand side to zero yields the system of linear equations

$$A^{(k)T}\mathbf{d} = -\mathbf{r}^{(k)}. \tag{2.4}$$

This system forms the basis of the *Newton-Raphson (NR) method* in which (2.4) is solved for a displacement $\mathbf{d} = \mathbf{d}^{(k)}$, and $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$ becomes the next iterate.

A solution $\mathbf{x}^*$ is said to be *regular* iff A^* is nonsingular. Assuming that A is Lipschitz continuous in a neighbourhood of $\mathbf{x}^*$, then the Newton-Raphson method converges locally (that is if some $\mathbf{x}^{(k)}$ is sufficiently close to $\mathbf{x}^*$) and the order of convergence is second order, that is

$$\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| = O(\|\mathbf{x}^{(k)} - \mathbf{x}^*\|^2).$$

We prove these properties in the next subsection. This property is very desirable, indicating as it does that the number of significant figures of accuracy doubles on each iteration, in the limit. Moreover a theorem of Dennis and Moré [10] indicates that if $\mathbf{x}^*$ is regular then any sequence $\{\mathbf{x}^{(k)}\}$ converging to $\mathbf{x}^*$ exhibits superlinear convergence (that is $\|\mathbf{x}^{(k+1)} - \mathbf{x}^*\| = o(\|\mathbf{x}^{(k)} - \mathbf{x}^*\|)$ if and only if the displacements converge asymptotically to those of the Newton-Raphson method, in the sense that

$$\mathbf{d}^{(k)} = \mathbf{d}^{(k)}_{NR} + o(\|\mathbf{d}^{(k)}_{NR}\|).$$

We shall see in what follows that many successful methods of optimization (Newton methods) are derived from the Newton-Raphson method, including the SQP method, which indicates the fundamental importance of the Dennis and Moré result.

An important class of Newton methods are the so-called *quasi-Newton methods* in which $A^{(k)}$ is approximated by a matrix $B^{(k)}$. Initially $B^{(k)}$ is arbitrary, and is *updated* after each iteration to take advantage of information gained about how $\mathbf{r}(\mathbf{x})$ behaves.

Unfortunately this favourable theoretical profile of the NR method is only valid locally in a neighbourhood of a regular solution. If $\mathbf{x}^{(1)}$ is remote from $\mathbf{x}^*$ then there is no guarantee of convergence, and indeed $A^{(k)}$ can be singular in which case (2.4) usually has no solution and the method aborts. Modifications of the NR method to promote global convergence (that is convergence when $\mathbf{x}^{(1)}$ is remote from $\mathbf{x}^*$) are a subject of much active research interest, some of which is described later in this monograph.

2.2 Local Convergence of the Newton-Raphson Method

Let $\mathbf{x}^*$ be a solution of (2.2). Then the following theorem holds.

Theorem Assume that $\mathbf{r}(\mathbf{x})$ is continuously differentiable in a neighbourhood of $\mathbf{x}^*$, and that A^* is nonsingular. Then there exists a neighbourhood of $\mathbf{x}^*$ such that if any point $\mathbf{x}^{(k)}$ is within this neighbourhood, then the Newton-Raphson method converges to $\mathbf{x}^*$ and the order of convergence is superlinear.

Proof The conditions of the theorem ensure that (2.3) is valid. Let $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}^*$ denote the error in $\mathbf{x}^{(k)}$. Setting $\mathbf{d} = -\mathbf{e}^{(k)}$ in (2.3) gives

$$\mathbf{0} = \mathbf{r}^* = \mathbf{r}^{(k)} - A^{(k)T}\mathbf{e}^{(k)} + o(\|\mathbf{e}^{(k)}\|). \tag{2.5}$$

Because A^* is nonsingular, there exists a neighbourhood of $\mathbf{x}^*$ in which $(A(\mathbf{x}))^{-1}$ is bounded above. If $\mathbf{x}^{(k)}$ is in this neighbourhood, substituting $\mathbf{r}^{(k)}$ from (2.4) into (2.5), and multiplying through by ${A^{(k)}}^{-T}$ gives

$$\mathbf{0} = -\mathbf{d}^{(k)} - \mathbf{e}^{(k)} + o(\|\mathbf{e}^{(k)}\|). \tag{2.6}$$

Then from $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$ and the definition of $\mathbf{e}^{(k)}$, it follows that

$$\mathbf{e}^{(k+1)} = o(\|\mathbf{e}^{(k)}\|). \tag{2.7}$$

Thus if $\mathbf{e}^{(k)}$ is sufficiently small, there exists a K and an $\alpha \in (0,\,1)$ such that

$$\|\mathbf{e}^{(k+1)}\| \le \alpha\,\|\mathbf{e}^{(k)}\| \tag{2.8}$$

for all $k \ge K$. Thus $\mathbf{x}^{(k)} \to \mathbf{x}^*$ and (2.7) shows that the order of convergence is superlinear.

QED

Corollary If, in addition, the Jacobian matrix $A(\mathbf{x})$ satisfies a Lipschitz condition, then the order of convergence is second order.
Proof In this case we can write the Taylor series (2.3) in the form

$$\mathbf{r}(\mathbf{x}^{(k)} + \mathbf{d}) = \mathbf{r}^{(k)} + A^{(k)T}\mathbf{d} + O(\|\mathbf{d}\|^2). \tag{2.9}$$

Following a similar argument to the above, we deduce that

$$\mathbf{e}^{(k+1)} = O(\|\mathbf{e}^{(k)}\|^2) \tag{2.10}$$

which is the definition of second order convergence. **QED**

2.3 Unconstrained Optimization

In this case the 'linear' problem is that of minimizing a quadratic function of n variables

$$q(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{h}^T\mathbf{x} \tag{2.11}$$

where the Hessian matrix of second derivatives G is symmetric and positive definite. The corresponding nonlinear problem is to find a local minimizing point $\mathbf{x}^*$ of a given non-quadratic function $f(\mathbf{x})$. We refer to the gradient (column) vector of first partial derivatives of $f(\mathbf{x})$ by $\mathbf{g}(\mathbf{x}) = \boldsymbol{\nabla} f(\mathbf{x})$ and the Hessian by $G(\mathbf{x}) = \boldsymbol{\nabla}\mathbf{g}(\mathbf{x})^T$. If $\mathbf{x}^*$ is a local minimizer of $f(\mathbf{x})$ then clearly it is a minimizer along any line

$$\mathbf{x}(\alpha) = \mathbf{x}^* + \alpha\mathbf{s}, \qquad \mathbf{s} \neq \mathbf{0}$$

through $\mathbf{x}^*$. It follows from this and the chain rule that the slope

$$df(\mathbf{x}(\alpha))/d\alpha|_{\alpha=0} = \mathbf{g}^*\mathbf{s} = 0$$

for any $\mathbf{s}$. Consequently a necessary condition for $\mathbf{x}^*$ to be a local minimizer of $f(\mathbf{x})$ is that $\mathbf{g}^* = \mathbf{0}$. Points $\mathbf{x}$ which satisfy $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ are referred to as *stationary points*, and include saddle points and maximizers as well as minimizers. Similarly another necessary condition for a minimizer is that the second derivative of $f(\mathbf{x}(\alpha))$ at $\alpha = 0$ is non-negative, that is $\mathbf{s}^T G^*\mathbf{s} \geq 0$ for all $\mathbf{s}$, which is the condition that G^* is positive semi-definite. On the other hand, if both $\mathbf{g}^* = \mathbf{0}$ and G^* is positive definite, then this is a *sufficient* condition for $\mathbf{x}^*$ to be a local minimizer.

The stationary point condition $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ is a system of nonlinear equations that can be solved by the NR method to find a stationary point of $f(\mathbf{x})$. The Jacobian $\boldsymbol{\nabla}\mathbf{g}(\mathbf{x})^T$ is just the Hessian $G(\mathbf{x})$. For minimization we are interested in the case that G is positive definite. In the case of a quadratic function (2.11), $\mathbf{g}(\mathbf{x}) = G\mathbf{x} + \mathbf{h}$ and we solve the system $G\mathbf{x} = -\mathbf{h}$. In the non-quadratic case, the appropriate regularity condition is that G^* is positive definite, and the NR iteration formula (2.4) becomes $G^{(k)}\mathbf{d} = -\mathbf{g}^{(k)}$. These linear systems are most efficiently solved using Choleski factors $G = LL^T$. We refer to this method as the *Newton method for minimization.* The method inherits all the favourable local properties of the NR method, described in the previous subsection. Correspondingly the method must be modified to promote global convergence. There is an additional issue that not only might $G^{(k)}$ be singular, whence the method aborts, but also $G^{(k)}$ might become indefinite, remote from the solution. In this case the local quadratic approximating function $q(\mathbf{d}) = \frac{1}{2}\mathbf{d}^T G^{(k)}\mathbf{d} + \mathbf{g}^{(k)T}\mathbf{d}$ no longer has a minimizer, and the resulting displacement $\mathbf{d}^{(k)}$ obtained by finding the stationary point of $q(\mathbf{d})$ is unlikely to be useful. Thus the most effective way of promoting global convergence is still a subject of some interest. In this respect, quasi-Newton methods are of particular interest because there exist methods of updating a Hessian approximating matrix $B^{(k)}$ so as to maintain the property that $B^{(k)}$ is symmetric positive definite.

2.4 Optimization with Linear Equality Constraints

This section is mainly important for the techniques it introduces in regard to handling linear constraints, which form a major feature of the Quadratic Programming (QP) method that is the subproblem of the SQP method. The 'linear' problem that we consider is the Equality QP (EQP) problem

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{h}^T\mathbf{x} \\ \text{subject to} & A^T\mathbf{x} = \mathbf{b}. \end{array} \tag{2.12}$$

In this problem A is an $n \times m$ matrix with $m \leq n$. The constraints are regular iff $\operatorname{rank}(A) = m$, which we assume to be the case. When $m = n$ the solution is simply that given by (2.1) and $q(\mathbf{x})$ plays no part. We shall focus therefore on the case that $m < n$, whence the equations in (2.12) are under-determined.

We can express the general solution of $A^T\mathbf{x} = \mathbf{b}$ as

$$\mathbf{x} = \mathbf{x}^\circ + Z\mathbf{t} \tag{2.13}$$

where $\mathbf{x}^\circ$ is a *particular solution* of $A^T\mathbf{x} = \mathbf{b}$, Z is an $n \times (n-m)$ matrix whose columns are a basis for the null space of A^T, that is $\operatorname{null}(A^T) = \{\mathbf{z} \,|\, A^T\mathbf{z} = \mathbf{0}\}$, and $\mathbf{t} \in \mathbb{R}^{n-m}$ is an arbitrary vector. The $Z\mathbf{t}$ term in (2.13) expresses the non-uniqueness of solutions of $A^T\mathbf{x} = \mathbf{b}$. Neither $\mathbf{x}^\circ$ nor Z are uniquely defined and any valid choice is acceptable, although there are possible considerations relating to ill-conditioning. There are various ways of finding a suitable $\mathbf{x}^\circ$ and Z. One is to reduce A^T to upper echelon form, as described in any basic linear algebra text, using pivoting to avoid ill-conditioning. More relevant to QP software is to find any matrix, V say, such that $[A \,|\, V]$ is nonsingular, and also well-conditioned, insofar as that is possible. Denote $[A \,|\, V]^{-T} = [Y \,|\, Z]$ where Z has $n - m$ columns. Then it can readily be verified using $A^T Y = I$, $A^T Z = 0$ and

$\mathrm{rank}(Z) = n - m$ that $\mathbf{x}^\circ = Y\mathbf{b}$ is a particular solution and columns of Z are a basis for $\mathrm{null}(A^T)$.

What follows is known as the *Null Space Method* for solving EQP problems. Simply, we substitute the general solution (2.13) into the definition of $q(\mathbf{x})$, giving a *reduced QP*

$$Q(\mathbf{t}) = q(\mathbf{x}^\circ + Z\mathbf{t}) = \tfrac{1}{2}(\mathbf{x}^\circ + Z\mathbf{t})^T G(\mathbf{x}^\circ + Z\mathbf{t}) + \mathbf{h}^T(\mathbf{x}^\circ + Z\mathbf{t}).$$

We find a stationary point of $Q(\mathbf{t})$ by applying the condition $\boldsymbol{\nabla}_{\mathbf{t}} Q(\mathbf{t}) = \mathbf{0}$, giving rise to the system of equations

$$Z^T G Z \mathbf{t} = -Z^T(\mathbf{h} + G\mathbf{x}^\circ) \tag{2.14}$$

which is solved for $\mathbf{t}$. Then (2.13) defines the solution $\mathbf{x}^*$. The solution is a unique minimizer if and only if $Z^T G Z$, referred to as the *reduced Hessian matrix*, is positive definite.

The Null Space Method extends to solve any linear equality constrained problem (LECP) in which the objective function is non-quadratic, that is

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & A^T\mathbf{x} = \mathbf{b}. \end{array} \tag{2.15}$$

Again, we just substitute for $\mathbf{x}$ using (2.13) giving a reduced problem

$$\underset{\mathbf{t} \in \mathbb{R}^{n-m}}{\text{minimize}}\, F(\mathbf{t}) = f(\mathbf{x}^\circ + Z\mathbf{t}).$$

This is now a non-quadratic unconstrained minimization problem and can be solved by the methods of the previous subsection.

3 Optimization with Nonlinear Equations

In this section we consider the Equality constrained NLP problem

$$\text{ENLP}\left\{ \begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{c}(\mathbf{x}) = \mathbf{0} \end{array} \right.$$

where in general $\mathbf{c}(\mathbf{x})$ is a vector of m nonlinear functions. We assume that these functions are continuous and continuously differentiable ($\mathbb{C}^1$) functions of $\mathbf{x}$. In the null space method we looked for a parametrization of the feasible region which allows us to eliminate the linear constraints $A^T\mathbf{x} = \mathbf{b}$, and so solve an unconstrained problem. In this section we seek to do the same when the constraint manifold is nonlinear. The development of this section provides an elegant and concise introduction to the concept of so-called *Lagrange multipliers* and their relation to the reduced optimization problems seen in the previous section. We make use of a locally valid nonlinear transformation, which, although computationally unattractive, does enable us to state necessary optimality conditions, and to derive a Newton method which is ultimately the basis for the SQP method.

3.1 Stationary points and Lagrange multipliers

In this section we follow the rationale of the null space method and attempt to derive an equivalent reduced unconstrained optimization problem. In this case however it is necessary to make a nonlinear transformation of variables, and there exist exceptional situations in which this is not possible. In order therefore to ensure that our transformation is well defined, local to a solution $\mathbf{x}^*$ of the ENLP, we make the *regularity assumption* that the columns of the Jacobian matrix A^* are linearly independent, or equivalently that $\text{rank}(A^*) = m$.

Existence and some properties of the transformation are a consequence of the *Inverse Function Theorem*, an important result which can be found in texts on real variable calculus. It may be stated as follows. Let $\mathbf{r}(\mathbf{x})$, $\mathbb{R}^n \to \mathbb{R}^n$, be a $\mathbb{C}^1$ nonlinear mapping, and let $\mathbf{x}^*$ be such that $\boldsymbol{\nabla}\mathbf{r}(\mathbf{x}^*)$ is nonsingular. Then open neighbourhoods of $\mathbf{x}^*$ and $\mathbf{r}^*$ $(= \mathbf{r}(\mathbf{x}^*))$ exist within which a $\mathbb{C}^1$ inverse mapping $\mathbf{x}(\mathbf{r})$ is uniquely defined, so that $\mathbf{x}(\mathbf{r}(\mathbf{x})) = \mathbf{x}$ and $\mathbf{r}(\mathbf{x}(\mathbf{r})) = \mathbf{r}$. Moreover, derivatives of the mappings are related by

$$\boldsymbol{\nabla}_{\mathbf{r}}\mathbf{x}^T = (\boldsymbol{\nabla}_{\mathbf{x}}\mathbf{r}^T)^{-1}. \tag{3.1}$$

In the case of the ENLP above, we choose any fixed matrix V such that $[A^* \,|\, V]$ is nonsingular (this is possible by virtue of the regularity assumption), and consider the nonlinear mapping

$$\mathbf{r}(\mathbf{x}) = \begin{pmatrix} \mathbf{c}(\mathbf{x}) \\ V^T(\mathbf{x} - \mathbf{x}^*) \end{pmatrix}, \tag{3.2}$$

noting that $\mathbf{r}(\mathbf{x}^*) = \mathbf{0}$. The Jacobian of the transformation is $\boldsymbol{\nabla}_{\mathbf{x}}\mathbf{r}^T = [A \,|\, V]$ which is nonsingular at $\mathbf{x}^*$. It follows by virtue of the inverse function therem that a well defined inverse function $\mathbf{x}(\mathbf{r})$ exists in a neighbourhood of $\mathbf{x}^*$. We consider the constrained form of $\mathbf{x}(\mathbf{r})$ in which $\mathbf{r} = \binom{\mathbf{0}}{\mathbf{t}}$. This defines a function $\mathbf{x}(\mathbf{t})$, $\mathbf{t} \in \mathbb{R}^{n-m}$, for which $\mathbf{c}(\mathbf{x}(\mathbf{t})) = \mathbf{0}$, and so provides a parametrization of the feasible region of the ENLP which is valid local to $\mathbf{x}^*$. Moreover, for any $\mathbf{x}$ local to $\mathbf{x}^*$ there corresponds a unique value of $\mathbf{t} = V^T(\mathbf{x} - \mathbf{x}^*)$, and $\mathbf{x} = \mathbf{x}^*$ corresponds to $\mathbf{t} = \mathbf{0}$. It also follows from (3.1) that

$$\boldsymbol{\nabla}_{\mathbf{r}}\mathbf{x}(\mathbf{r}) = [A \,|\, V]^{-1} = \begin{bmatrix} Y^T \\ Z^T \end{bmatrix}, \tag{3.3}$$

say, so that

$$\boldsymbol{\nabla}_{\mathbf{t}}\mathbf{x}(\mathbf{t}) = Z^T \qquad \text{and hence} \qquad \partial x_i / \partial t_j = z_{ij}. \tag{3.4}$$

Note that Z is no longer a constant matrix, and the expressions are only valid local to $\mathbf{x}^*$, in contrast to the null space method for handling linear constraints.

We can now state an equivalent reduced unconstrained optimization problem for the ENLP, that is

$$\underset{\mathbf{t} \in \mathbb{R}^{n-m}}{\text{minimize}}\, F(\mathbf{t}), \tag{3.5}$$

where $F(\mathbf{t}) = f(\mathbf{x}(\mathbf{t}))$. A stationary point of $F(\mathbf{t})$ is defined by the condition that $\boldsymbol{\nabla}_{\mathbf{t}} F(\mathbf{t}) = \mathbf{0}$. By the chain rule

$$\frac{\partial}{\partial t_j} = \sum_i \frac{\partial x_i}{\partial t_j} \frac{\partial}{\partial x_i} = \sum_i z_{ij} \frac{\partial}{\partial x_i}$$

from (3.4), or

$$\boldsymbol{\nabla}_{\mathbf{t}} = Z^T \boldsymbol{\nabla}_{\mathbf{x}}. \tag{3.6}$$

This result shows how derivatives in the reduced space are related to those in the full space. Applying these derivatives to $F(\mathbf{t})$ and $f(\mathbf{x})$ at $\mathbf{x}^*$, we obtain a stationary point condition for the ENLP problem

$$Z^{*T} \mathbf{g}(\mathbf{x}^*) = \mathbf{0}, \tag{3.7}$$

where Z^* denotes the matrix given by (3.3) when $\mathbf{x} = \mathbf{x}^*$ and $A = A^*$. The vector $Z^T \mathbf{g}$ is referred to as the *reduced gradient* and is zero at a stationary point of the ENLP problem.

There is also an alternative formulation of the stationary point condition that can be deduced. Arising from (3.3) and the regularity assumption $\operatorname{rank}(A^*) = m$, we know that both A^* and Z^* have linearly independent columns. The definition of the inverse in (3.3) implies that $A^{*T} Z^* = 0$ showing that columns of Z^* are in $\operatorname{null}(A^{*T})$. But we might equally write $Z^{*T} A^* = 0$, showing that columns of A^* are in $\operatorname{null}(Z^{*T})$, and indeed provide a basis for $\operatorname{null}(Z^{*T})$ by virtue of linear independence. Now the stationary point condition (3.7) states that $\mathbf{g}^* \in \operatorname{null}(Z^{*T})$. Thus we can express $\mathbf{g}^*$ as a linear combination of basis vectors for $\operatorname{null}(Z^{*T})$, that is

$$\mathbf{g}^* = A^* \boldsymbol{\lambda}^* = \sum_{i=1}^{m} \mathbf{a}_i^* \lambda_i^*. \tag{3.8}$$

The multipliers $\boldsymbol{\lambda}^*$ in the linear combination are referred to as *Lagrange multipliers*.

Equations (3.7) and (3.8) provide alternative and equivalent statements of the stationary point conditions for an ENLP problem. They are often referred to as *first order necessary conditions* for a local solution of a regular ENLP problem. We can also express (3.8) as $\mathbf{g}^* \in \operatorname{range}(A^*)$. These alternative viewpoints of null space and range space formulations pervade much of both the theoretical and computational aspects of quadratic programming and nonlinear programming, as we shall see below.

Satisfying (3.8) and feasibility provides a method for solving the ENLP, that is find $\mathbf{x}^*, \boldsymbol{\lambda}^*$ to solve the system of $n + m$ equations

$$\mathbf{r}(\mathbf{x}, \boldsymbol{\lambda}) = \begin{pmatrix} \mathbf{g} - A\boldsymbol{\lambda} \\ -\mathbf{c} \end{pmatrix} = \mathbf{0}, \tag{3.9}$$

in the $n + m$ variables $\mathbf{x}, \boldsymbol{\lambda}$, where $\mathbf{g}$, A and $\mathbf{c}$ are functions of $\mathbf{x}$. This is the so-called *method of Lagrange multipliers*. However, the system is generally nonlinear in $\mathbf{x}$ and may not be straightforward to solve, as we have seen above. It also can only be expected to

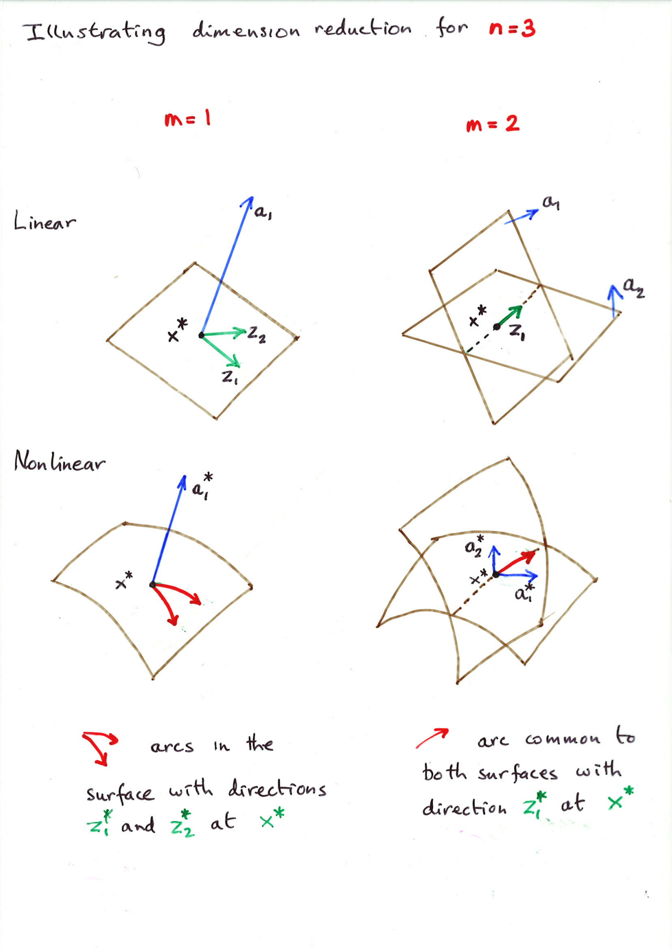
Illustrating dimension reduction for n=3
m=1
m=2
Linear
a₁
x*
z₂
z₁
a₁
a₂
x*
z₁
Nonlinear
a₁*
x*
a₂*
x*
a₁*
arcs in the surface with directions z₁* and z₂* at x*
arc common to both surfaces with direction z₁* at x*

Figure 1: Illustrating dimension reduction for $n = 3$

yield stationary points of the ENLP. Equation (3.9) can also be interpreted as finding a stationary point of a *Lagrangian function*

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x}), \tag{3.10}$$

since $\boldsymbol{\nabla}_{\mathbf{x}}\mathcal{L} = \mathbf{g} - A\boldsymbol{\lambda}$ and $\boldsymbol{\nabla}_{\boldsymbol{\lambda}}\mathcal{L} = -\mathbf{c}$. The Lagrangian function plays a central rôle in both the theoretical and computational aspects of nonlinear programming and the SQP method.

Lagrange multipliers also have a useful interpretation in terms of the sensitivity of the

ENLP to perturbations in the constraints. Consider an ENLP problem

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{c}(\mathbf{x}) = \boldsymbol{\varepsilon} \end{array} \tag{3.11}$$

in which the right hand sides of the constraints have been perturbed by an amount $\boldsymbol{\varepsilon}$. Let $\mathbf{x}(\boldsymbol{\varepsilon})$, $\boldsymbol{\lambda}(\boldsymbol{\varepsilon})$ be the solution and multipliers of the perturbed problem, and consider $f(\mathbf{x}(\boldsymbol{\varepsilon}))$. Then

$$df(x(\boldsymbol{\varepsilon}))/d\varepsilon_i = \lambda_i, \tag{3.12}$$

showing that λ_i measures the change of $f(\mathbf{x}(\boldsymbol{\varepsilon}))$ with respect to a change ε_i in constraint i, to first order. To prove this result, let the Lagrangian of the perturbed ENLP problem be

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\varepsilon}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T(\mathbf{c}(\mathbf{x}) - \boldsymbol{\varepsilon}),$$

and observe that $\mathcal{L}(\mathbf{x}(\boldsymbol{\varepsilon}), \boldsymbol{\lambda}(\boldsymbol{\varepsilon}), \boldsymbol{\varepsilon}) = f(\mathbf{x}(\boldsymbol{\varepsilon}))$. Then the chain rule gives

$$\frac{df}{d\varepsilon_i} = \frac{d\mathcal{L}}{d\varepsilon_i} = \frac{\partial \mathbf{x}^T}{\partial \varepsilon_i}\boldsymbol{\nabla}_{\mathbf{x}}\mathcal{L} + \frac{\partial \boldsymbol{\lambda}^T}{\partial \varepsilon_i}\boldsymbol{\nabla}_{\boldsymbol{\lambda}}\mathcal{L} + \frac{\partial \mathcal{L}}{\partial \varepsilon_i} = \lambda_i$$

by virtue of the stationarity of the Lagrangian function with respect to $\mathbf{x}$ and $\boldsymbol{\lambda}$.

3.2 Second Order Conditions for the ENLP problem

Let $\mathbf{x}^*$ solve the ENLP and let $\text{rank}(A^*) = m$ (regularity). Then we have seen that (3.5) is an equivalent reduced unconstrained minimization problem. Thus, from Section 2.2, a second order necessary condition is that the Hessian matrix $\nabla^2_{\mathbf{t}}F(\mathbf{t})$ is positive semi-definite. To relate this to the ENLP, we use equation (3.6) which relates derivatives in the reduced and full systems. Thus

$$\nabla^2_{\mathbf{t}}F(\mathbf{t}) = \boldsymbol{\nabla}_{\mathbf{t}}(\boldsymbol{\nabla}_{\mathbf{t}}F(\mathbf{t}))^T = Z^T\boldsymbol{\nabla}_{\mathbf{x}}(\mathbf{g}^TZ).$$

When the constraints are linear, such as in (2.15), Z is a constant matrix so we can differentiate further to get

$$Z^T\boldsymbol{\nabla}_{\mathbf{x}}(\mathbf{g}^TZ) = Z^T(\boldsymbol{\nabla}_{\mathbf{x}}\mathbf{g}^T)Z = Z^TGZ$$

where G is the Hessian matrix of $f(\mathbf{x})$. Thus the second order necessary condition in this case is that the *reduced Hessian matrix* Z^TG^*Z is positive semi-definite. Moreover, $Z^T\mathbf{g}^* = \mathbf{0}$ and Z^TG^*Z being positive definite are sufficient conditions for $\mathbf{x}^*$ to solve (3.5).

For an NELP with nonlinear constraints, Z depends on $\mathbf{x}$ and the last differentiation is not valid. To make progress, we observe that the ENLP is equivalent to the problem

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^*) \\ \text{subject to} & \mathbf{c}(\mathbf{x}) = \mathbf{0} \end{array} \tag{3.13}$$

since $f(\mathbf{x}) = \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ when $\mathbf{c}(\mathbf{x}) = \mathbf{0}$. We now define $\mathbf{x}(\mathbf{t})$ as in (3.2), and consider the problem of minimizing a reduced function $F(\mathbf{t}) = \mathcal{L}(\mathbf{x}(\mathbf{t}), \boldsymbol{\lambda}^*)$. Then $\boldsymbol{\nabla}_{\mathbf{t}} F = \mathbf{0}$ becomes $Z^T \boldsymbol{\nabla}_{\mathbf{x}} \mathcal{L} = \mathbf{0}$, or $Z^T(\mathbf{g} - A\boldsymbol{\lambda}^*) = \mathbf{0}$. At $\mathbf{x}^*$, it follows that $Z^{*T}\mathbf{g}^* = \mathbf{0}$ which is the first order necessary condition. For second derivatives,

$$\nabla^2_{\mathbf{t}} F(\mathbf{t}) = \boldsymbol{\nabla}_{\mathbf{t}}(\boldsymbol{\nabla}_{\mathbf{t}} F(\mathbf{t}))^T = Z^T \boldsymbol{\nabla}_{\mathbf{x}}((\mathbf{g} - A\boldsymbol{\lambda}^*)^T Z).$$

At $\mathbf{x}^*$, derivatives of Z are multiplied by $\mathbf{g}^* - A^*\boldsymbol{\lambda}^*$ which is zero, so we have

$$\nabla^2_{\mathbf{t}} F(\mathbf{t}^*) = Z^T \boldsymbol{\nabla}_{\mathbf{x}}((\mathbf{g} - A\boldsymbol{\lambda}^*)^T) Z|_{\mathbf{x}^*} = Z^{*T} W^* Z^*,$$

where

$$W(\mathbf{x}, \boldsymbol{\lambda}) = \nabla^2_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \lambda) = \nabla^2 f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i \nabla^2 c_i(\mathbf{x}) \tag{3.14}$$

is the Hessian with respect to $\mathbf{x}$ of the Lagrangian function, and $W^* = W(\mathbf{x}^*, \boldsymbol{\lambda}^*)$. Thus the second order necessary condition for the regular ENLP problem is that the reduced Hessian of the Lagrangian function is positive semi-definite. As above, a sufficient condition is that the reduced Hessian is positive definite and the reduced gradient is zero.

3.3 The SQP method for the ENLP problem

We have seen in (3.9) that a stationary point of a regular ENLP can be found by solving a system of nonlinear equations. Applying the Newton-Raphson method to these equations enables us to derive a Newton type method with rapid local convergence properties. First however we consider solving (3.9) in the case of an EQP problem (2.12). In this case, $\mathbf{g} = G\mathbf{x} + \mathbf{h}$ and $\mathbf{c} = A^T\mathbf{x} - \mathbf{b}$, so (3.9) can be written as the system of $n + m$ linear equations in $n + m$ unknowns

$$\begin{bmatrix} G & -A \\ -A^T & 0 \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \boldsymbol{\lambda} \end{pmatrix} = - \begin{pmatrix} \mathbf{h} \\ \mathbf{b} \end{pmatrix}. \tag{3.15}$$

Although symmetric, the coefficient matrix in (3.15) is indefinite so cannot be solved by using Choleski factors. The Null Space Method (2.14) is essentially one way of solving (3.15), based on eliminating the constraints $A^T\mathbf{x} = \mathbf{b}$. When G is positive definite, and particularly when G permits sparse Choleski $G = LL^T$ factors to be obtained, it can be more effective to use the first block equation to eliminate $\mathbf{x} = G^{-1}(A\boldsymbol{\lambda} - \mathbf{h})$. Then the system

$$A^T G^{-1} A \, \boldsymbol{\lambda} = \mathbf{b} + A^T G^{-1} \mathbf{h} \tag{3.16}$$

is used to determine $\boldsymbol{\lambda}$, and hence implicitly $\mathbf{x}$. Of course, operations with G^{-1} are carried out by making triangular solves with the Choleski factor L. This method might be regarded as the *Range Space Method for EQP*, in contrast to the Null Space Method described earlier.

We now proceed to consider the ENLP problem in the general case of nonlinear constraints. In this case we attempt to solve the equations $\mathbf{r}(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$ in (3.9) by the Newton-Raphson method. First we need the Jacobian matrix of this system, which is the $(n+m) \times (n+m)$ matrix

$$\begin{pmatrix} \boldsymbol{\nabla}_{\mathbf{x}} \\ \boldsymbol{\nabla}_{\boldsymbol{\lambda}} \end{pmatrix} \mathbf{r}^T = \begin{bmatrix} W & -A \\ -A^T & 0 \end{bmatrix} \tag{3.17}$$

where W, defined in (3.14), is the Hessian with respect to $\mathbf{x}$ of the Lagrangian function. The current iterate in the method is the pair of vectors $\mathbf{x}^{(k)}$, $\boldsymbol{\lambda}^{(k)}$, and the iteration formula, generalising (2.4), is

$$\begin{bmatrix} W^{(k)} & -A^{(k)} \\ -A^{(k)T} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{d}^{(k)} \\ \boldsymbol{\delta}^{(k)} \end{pmatrix} = -\mathbf{r}^{(k)} = \begin{pmatrix} A^{(k)}\boldsymbol{\lambda}^{(k)} - \mathbf{g}^{(k)} \\ c^{(k)} \end{pmatrix} \tag{3.18}$$

where superscript (k) denotes quantities calculated from $\mathbf{x}^{(k)}$ and $\boldsymbol{\lambda}^{(k)}$. Then updated values for the next iteration are defined by $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$ and $\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} + \boldsymbol{\delta}^{(k)}$. These formulae may be rearranged by moving the $A^{(k)}\boldsymbol{\lambda}^{(k)}$ term to the left hand side of (3.19), giving

$$\begin{bmatrix} W^{(k)} & -A^{(k)} \\ -A^{(k)T} & 0 \end{bmatrix} \begin{pmatrix} \mathbf{d}^{(k)} \\ \boldsymbol{\lambda}^{(k+1)} \end{pmatrix} = \begin{pmatrix} -\mathbf{g}^{(k)} \\ \mathbf{c}^{(k)} \end{pmatrix}. \tag{3.19}$$

This then is a Newton iteration formula for finding a stationary point of an EQP problem. For rapid local convergence to a stationary point $\mathbf{x}^*$ with multipliers $\boldsymbol{\lambda}^*$, we require that the Jacobian matrix

$$\begin{bmatrix} W^* & -A^* \\ -A^{*T} & 0 \end{bmatrix} \tag{3.20}$$

is nonsingular.

So where does the SQP method come in? There are two important observations to make. First, if the constraints in the ENLP problem are regular ($\text{rank}(A^*) = m$), and the ENLP satisfies second order sufficiency conditions ($Z^{*T}W^*Z^*$ is positive definite), then it is a nice exercise in linear algebra to show that the matrix (3.20) is nonsingular (see [13]). Thus the local rapid convergence of (3.19) is assured. Moreover, it also follows that $\text{rank}(A^{(k)}) = m$ and $Z^{(k)T}W^{(k)}Z^{(k)}$ is positive definite in a neighbourhood of $\mathbf{x}^*$, $\boldsymbol{\lambda}^*$. Under these conditions, the EQP problem

$$\text{EQP}^{(k)} \left\{ \begin{array}{ll} \underset{\mathbf{d} \in \mathbb{R}^n}{\text{minimize}} & \frac{1}{2}\mathbf{d}^T W^{(k)} \mathbf{d} + \mathbf{d}^T \mathbf{g}^{(k)} \\ \text{subject to} & \mathbf{c}^{(k)} + A^{(k)T}\mathbf{d} = \mathbf{0} \end{array} \right.$$

is regular and has a unique local minimizer, which can be found by solving the stationary point condition (see (3.15)), which for $\text{EQP}^{(k)}$ is none other than (3.19). Thus, for finding a local minimizer of an ENLP problem, it is better to replace the iteration formula (3.19) by one based on solving $\text{EQP}^{(k)}$ for a correction $\mathbf{d}^{(k)} = \mathbf{d}$ and multiplier vector $\boldsymbol{\lambda}^{(k+1)}$.

This correctly accounts for the second order condition required by a local minimizer to an ENLP problem. In particular, a solution of (3.19) which corresponds to a saddle point of EQP$^{(k)}$ is not accepted. (EQP$^{(k)}$ is unbounded in this situation.) Also EQP$^{(k)}$ has a nice interpretation: the constraints are linear Taylor series approximations about $\mathbf{x}^{(k)}$ to those in the ENLP problem, and the objective function is a quadratic Taylor series approximation about $\mathbf{x}^{(k)}$ to the objective function in the ENLP, plus terms in $W^{(k)}$ that account for constraint curvature. The objective function can equally be viewed as a quadratic approximation to the Lagrangian function.

To summarize, solving an ENLP in this way may be interpreted as a Sequential EQP method (SEQP) with the following basic structure

```
initialize  $\mathbf{x}^{(1)}, \boldsymbol{\lambda}^{(1)}$
for  $k = 1, 2, \dots$  until converged
    solve  EQP$^{(k)}$  giving  $\mathbf{d}^{(k)}$  and  multipliers  $\boldsymbol{\lambda}^{(k+1)}$
    set  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$
end
```

As with other Newton methods, the method may not converge globally, and EQP$^{(k)}$ may have no solution, for example it may be unbounded, or possibly infeasible if $A^{(k)}$ is rank deficient. It is therefore essential that extra features are included in any practical implementation. We return to this subject later in the monograph.

4 Inequality Constraints and Nonlinear Programming

In this section we examine the extra complication caused by having inequality constraints in the formulation of an optimization problem. As above we discuss 'linear' problems which can be solved in a finite number of steps, and nonlinear problems for which iteration is required, leading to a general formulation of the SQP method. We also discuss changes to optimality conditions to accommodate inequality constraints.

4.1 Systems of Inequalities

Corresponding to the development of Section 2.1, the 'linear' problem we now consider is that of a system of linear inequalities

$$A^T\mathbf{x} \geq \mathbf{b} \tag{4.1}$$

in which A is a given $n \times m$ matrix of coefficients and $\mathbf{b}$ is a given vector of right hand sides. Usually $m > n$ when there is no objective function present, although in general, $m \leq n$ is also possible. Each inequality $\mathbf{a}_i^T\mathbf{x} \geq b_i$ in (4.1) divides $\mathbb{R}^n$ into two parts, a *feasible* side and an *infeasible* side, with respect to the inequality. Equality holds on the boundary. Any n independent such equations define a point of intersection, referred to as a *vertex* in this context. Usually methods for solving (4.1) attempt to locate a feasible vertex. Each vertex can be found by solving a system of linear equations as in (2.1). There are only a finite number of vertices so the process will eventually find a solution, or establish that none exists. However, there may be as many as $\binom{m}{n}$ vertices, which can be extremely large for problems of any size. Thus it is important to enumerate the vertices in an efficient way. This can be done by a modification known as *Phase I* of the so-called Simplex Method for Linear Programming, which we describe briefly below (see also [13] for example). An important saving in this case is that adjacent vertices differ by only one equation, and this can be used to update matrix factors to gain efficiency.

The corresponding nonlinear problem is the system of nonlinear inequalities

$$\mathbf{r}(\mathbf{x}) \geq \mathbf{0} \tag{4.2}$$

in which $\mathbf{r}(\mathbf{x})$ is a given vector of m nonlinear functions. This problem can be solved by a Newton type algorithm in which a sequence of linearized subproblems are solved, each being obtained by a linear Taylor series approximation to $\mathbf{c}(\mathbf{x})$ about a current iterate $\mathbf{x}^{(k)}$. This just becomes a special case of the SQP method, and we shall defer discussion on it until later.

In cases when no solution exists to (4.1) or (2.1), it is often of interest to find a 'best' solution which minimizes some measure of constraint infeasibility. Exactly what measure to choose is a decision for the user, and is one which has implications for the type of method that is possible.

4.2 Optimization with Inequality Constraints

In this case our generic linear problem, which can be solved finitely, is the QP problem

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{h}^T\mathbf{x} \\ \text{subject to} & A^T\mathbf{x} \geq \mathbf{b}. \end{array} \tag{4.3}$$

In this problem A is an $n \times m$ matrix with no restrictions on the value of m. The corresponding nonlinear problem is the NLP problem (1.1). There is also the intermediate stage of a linearly constrained problem (LCP say) in which the objective function is non-quadratic, that is

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & A^T\mathbf{x} \geq \mathbf{b}. \end{array} \tag{4.4}$$

As in Section 1, a more general formulation of the QP and NLP problems is often appropriate for practical use, but the simplified form is convenient for introducing the main features. An important special case of QP, which can also be solved finitely, is the *Linear Programming* (LP) problem, characterized by $G = 0$ in (4.3).

In this section our main aim is to discuss optimality conditions for an NLP problem. An important concept is that of an *active constraint.* The set of active constraints at a point $\mathbf{x}$ is defined by

$$\mathcal{A}(\mathbf{x}) = \{i \,|\, c_i(\mathbf{x}) = 0\} \tag{4.5}$$

so that $i \in \mathcal{A}(\mathbf{x})$ indicates that $\mathbf{x}$ is on the boundary of constraint i. The set of active constraints at the solution is denoted by $\mathcal{A}^*$. Constraints not in $\mathcal{A}^*$ have no influence on the behaviour of the NLP problem, local to a local solution $\mathbf{x}^*$ of (1.1).

Clearly $\mathbf{x}^*$ solves the following ENLP problem

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & c_i(\mathbf{x}) = 0 \quad i \in \mathcal{A}^*. \end{array} \tag{4.6}$$

Let the gradient vectors $\mathbf{a}_i^* = \boldsymbol{\nabla} c_i^*$, $i \in \mathcal{A}^*$ be linearly independent (regularity). Then from (3.8) in Section 3.1 there exist multipliers $\boldsymbol{\lambda}^*$ such that

$$\mathbf{g}^* = \sum_{i \in \mathcal{A}^*} \mathbf{a}_i^* \lambda_i^* = \sum_{i=1}^{m} \mathbf{a}_i^* \lambda_i^* = A^* \boldsymbol{\lambda}^*, \tag{4.7}$$

denoting $\lambda_i^* = 0$ for inactive constraints $i \notin \mathcal{A}^*$. Moreover, by virtue of regularity, we can perturb the right hand side of (4.6) by a sufficiently small amount $\varepsilon_i > 0$, and $\varepsilon_j = 0$, $j \neq i$, and still retain feasibility in (1.1). Then, if $\lambda_i^* < 0$, it follows from (3.12) that $df/d\varepsilon_i = \lambda_i^* < 0$, which contradicts optimality. Thus

$$\lambda_i^* \geq 0 \tag{4.8}$$

for an *inequality constraint* is also necessary. (The multiplier of an equality constraint can take either sign.) The convention that the multiplier of an inactive constraint is zero may also be expressed as

$$\lambda_i^* c_i^* = 0 \tag{4.9}$$

which is referred to as the *complementarity condition.* If $\lambda_i^* > 0$ for all active inequality constraints, then *strict complementarity* is said to hold. Collectively, feasibility in (1.1), (3.8), (4.8) for an inequality constraint, and (4.9) are known as KT (Kuhn-Tucker) (or KKT (Karush-Kuhn-Tucker)) conditions (Karush [31], Kuhn and Tucker [32]). Subject to a regularity assumption of some kind, they are necessary conditions for a local solution of (1.1). A point $\mathbf{x}^*$ which satisfies KT conditions for some $\boldsymbol{\lambda}^*$ is said to be a *KT point.*

A second order necessary conditions that can be deduced from (4.6) is that $Z^{*T} W^* Z^*$ is positive semi-definite, where Z^* is the null-space basis matrix for (4.6), and W is defined in (3.14). A sufficient condition is that $\mathbf{x}^*$ is a KT point, strict complementarity holds, and $Z^{*T} W^* Z^*$ is positive definite.

4.3 Quadratic Programming

Before describing the SQP method, it is important to know how to solve a QP problem (4.3) that contains inequality constraints. A method with finite termination is the Active Set Method (ASM), which has features that are favourable in the context of SQP. The method solves (4.3) by solving a sequence of EQP problems, whilst retaining feasibility in (4.3), until the correct active set is determined. The method is described in the case that G is positive definite, all the contraints are inequalities, and there is no degeneracy (see below). The method is initialized by finding a feasible vertex, $\mathbf{x}^{(1)}$ say, as described in Section 4.1. We let $\mathcal{A}$ denote the current set of active constraints at $\mathbf{x}^{(k)}$. The current EQP is defined by

$$\begin{array}{lll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2}\mathbf{x}^T G\mathbf{x} + \mathbf{h}^T\mathbf{x} & \\ \text{subject to} & \mathbf{a}_i^T\mathbf{x} = b_i & i \in \mathcal{A}. \end{array} \tag{4.10}$$

Because $\mathbf{x}^{(1)}$ is a vertex, it is in fact the solution of the current EQP defined by $\mathcal{A}$. The ASM has two major steps.

(i) If $\mathbf{x}^{(k)}$ solves the current EQP, then find the corresponding multipliers $\boldsymbol{\lambda}^{(k)}$. Choose any $i : \lambda_i^{(k)} < 0$ (if none exist, then **finish** with $\mathbf{x}^* = \mathbf{x}^{(k)}$). Otherwise, remove i from $\mathcal{A}$ and goto step (ii).

(ii) Find the solution, $\hat{\mathbf{x}}$ say, of the current EQP. If $\hat{\mathbf{x}}$ is feasible in the QP problem then [set $\mathbf{x}^{(k+1)} = \hat{\mathbf{x}}$, $k = k + 1$ and goto step (i)]. Otherwise, set $\mathbf{x}^{(k+1)}$ as the closest feasible point to $\hat{\mathbf{x}}$ along the line segment from $\mathbf{x}^{(k)}$ to $\hat{\mathbf{x}}$. Add the index of a newly active constraint to $\mathcal{A}$. Set $k = k+1$. If $|\mathcal{A}| = n$ then goto step (i) else goto step (ii).

The motivation for the algorithm is provided by the observation in Section 4.2 that if there exists $i : \lambda_i^{(k)} < 0$ at the solution of the current EQP, then it is possible to relax constraint i whilst reducing the objective function. If $G = 0$ in (4.3) then we have a Linear Programming (LP) problem, and the ASM is essentially the same as the *Simplex method* for LP, although it is not often explained in this way.

Special linear algebra techniques are required to make the method efficient in practice. Changes to the current active set involve either adding or subtracting one constraint index. Updates to matrices such as Z and Z^TGZ can be performed much more quickly than re-evaluating the matrices. For large problems, it is important to take advantage of sparsity in A and possibly G.

There are some complicating factors for the ASM. If the Hessian G is not positive definite, then it is possible that the EQP obtained by removing i from $\mathcal{A}$ may be unbounded, so that $\hat{\mathbf{x}}$ does not exist. In this case an arbitrary choice of feasible descent direction is chosen, to make progress. If G has negative eigenvalues, then the QP problem may have local solutions, and the ASM does not guarantee to find a global solution. Any solution found by the ASM will be a KT point of the QP problem, but may not be a local solution unless strict complementarity holds. A more serious complicating factor is that of *degeneracy* which refers to the situation where regularity of the active constraints at the solution of an EQP problem fails to hold. An example would be where there are more than n active constraints at a feasible vertex. In this case, deciding whether $\mathbf{x}^{(k)}$ solves the current EQP, or whether a feasible descent direction exists, is a more complex issue, although a finite algorithm to decide the issue is possible. Degeneracy is often present in practical instances of QP problems, and it is important that it is correctly accounted for in a computer code.

More recently an alternative class of methods has become available for the solution of LP or QP problems in which G is positive semi-definite. These *interior point methods* have the advantage that they avoid the worst case behaviour of ASM and Simplex methods, in which the number of iterations required to locate the solution may grow exponentially with n. However, interior point methods also have some disadvantages in an SQP context.

4.4 The SQP Method

We are now in a position to describe the basic SQP method for an NLP (1.1) with inequality constraints. The method was first suggested in a thesis of Wilson (1960), [47], and became well known due to the work of Beale [1]. The idea follows simply from the SEQP method for an ENLP problem, where the equality constraints $\mathbf{c}(\mathbf{x}) = \mathbf{0}$ are approximated by the linear Taylor series $\mathbf{c}^{(k)} + A^{(k)T}\mathbf{d} = \mathbf{0}$ in the subproblem EQP$^{(k)}$. In an NLP with inequality constraints $\mathbf{c}(\mathbf{x}) \geq \mathbf{0}$ we therefore make the same approximation, leading to a QP subproblem with linear inequality constraints $\mathbf{c}^{(k)} + A^{(k)T}\mathbf{d} \geq \mathbf{0}$, that is

$$\mathrm{QP}^{(k)} \left\{ \begin{array}{ll} \underset{\mathbf{d}\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2}\mathbf{d}^T W^{(k)}\mathbf{d} + \mathbf{d}^T\mathbf{g}^{(k)} \\ \text{subject to} & \mathbf{c}^{(k)} + A^{(k)T}\mathbf{d} \geq \mathbf{0}. \end{array} \right.$$

The basic form of the algorithm therefore is that described at the end of Section 3.3, with the substitution of QP$^{(k)}$ for EQP$^{(k)}$. To view this method as a Newton-type method, we need to assume that that strict complementarity $\lambda_i^* > 0$, $i \in \mathcal{A}^*$ holds at a regular solution to (1.1). Then, if $\mathbf{x}^{(k)}$, $\boldsymbol{\lambda}^{(k)}$ is sufficiently close to $\mathbf{x}^*$, $\boldsymbol{\lambda}^*$, it follows that the solution of EQP$^{(k)}$ with active constraints $\mathcal{A}^*$, also satisfies the sufficient conditions for QP$^{(k)}$. Thus we can ignore inactive constraints $i \notin \mathcal{A}^*$, and the SQP method is identical to the SEQP method on the active constraint set $\mathcal{A}^*$. Thus the SQP method inherits the local rapid convergence of a Newton type method under these circumstances.

The progress of the SQP method on the NLP problem

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^2}{\text{minimize}} & f(\mathbf{x}) = -x_1 - x_2 \\ \text{subject to} & c_1(\mathbf{x}) = x_2 - x_1^2 \geq 0 \\ & c_2(\mathbf{x}) = 1 - x_1^2 - x_2^2 \geq 0 \end{array}$$

is illustrated in Table 1, and has some instructive features. Because the initial multiplier estimate is zero, and $f(\mathbf{x})$ is linear, the initial $W^{(1)}$ matrix is zero, and QP$^{(1)}$ is in fact an LP problem. Consequently, $\mathbf{x}^{(1)}$ has to be chosen carefully to avoid an unbounded subproblem (or alternatively one could add simple upper and lower bounds to the NLP problem). The solution of QP$^{(1)}$ delivers some non-zero multipliers for $\boldsymbol{\lambda}^{(2)}$, so that $W^{(2)}$ becomes positive definite. The solution of QP$^{(2)}$ predicts that constraint 1 is inactive, and we see that $\lambda_1^{(3)}$ is zero. This situation persists on all subsequent iterations. For this NLP problem, the active set is $\mathcal{A}^* = \{2\}$, and we see for $k \geq 3$, that the SQP method converges to the solution in the same way as the SEQP method with the single equality constraint $1 - x_1^2 - x_2^2 = 0$. The onset of rapid local convergence, characteristic of a Newton method, can also be observed.

Table 1: A Numerical Example of the SQP Method

k	$x_1^{(k)}$	$x_2^{(k)}$	$\lambda_1^{(k)}$	$\lambda_2^{(k)}$	$c_1^{(k)}$	$c_2^{(k)}$
1	$\frac{1}{2}$	1	0	0	$\frac{3}{4}$	$-\frac{1}{2}$
2	$\frac{11}{12}$	$\frac{2}{3}$	$\frac{1}{3}$	$\frac{2}{3}$	-0.173611	-0.284722
3	0.747120	0.686252	0	0.730415	0.128064	-0.029130
4	0.708762	0.706789	0	0.706737	0.204445	-0.001893
5	0.707107	0.707108	0	0.707105	0.207108	$-0.28_{10} - 5$

However, the basic method can fail to converge if $\mathbf{x}^{(k)}$ is remote from $\mathbf{x}^*$ (it is not as important to have $\boldsymbol{\lambda}^{(k)}$ close to $\boldsymbol{\lambda}^*$ because if $\mathbf{x}^{(k)}$ is close to $\mathbf{x}^*$, one solution of QP$^{(k)}$ will give an accurate multiplier estimate). It is also possible that QP$^{(k)}$ has no solution, either because it is unbounded, or because the linearized constraints are infeasible.

For these reasons, the SQP method is only the starting point for a fully developed NLP solver, and extra features must be added to promote convergence from remote initial values. This is the subject of subsequent sections of this monograph. Nonetheless it has been and still is is the method of choice for many researchers. The success of the method is critically dependent on having an efficient, flexible and reliable code for solving the QP subproblem. It is important to be able to take advantage of *warm starts*, that is, initializing the QP solver with the active set from a previous iteration. Also important is the ability to deal with the situation that the matrix $W^{(k)}$ is not positive semi-definite. For both these reasons, an active set method code for solving the QP subproblems is likely to be preferred to an interior point method. However, NLP solvers are still a very active research area, and the situation is not at all clear, especially when dealing with very large scale NLPs.

4.5 SLP-EQP Algorithms

An early idea for solving NLP problems is the successive linear programming (SLP) algorithm in which LP subproblem is solved ($W^{(k)} = 0$ in QP$^{(k)}$). This is able to take advantage of fast existing software for large scale LP. However, unless the solution of the NLP problem is at a vertex, convergence is slow because of the lack of second derivative information. A more recent development is the SLP-EQP algorithm, introduced by Fletcher and Sainz de la Maza [22], in which the SLP subproblem is used to determine the active set and multipliers, but the resulting step $\mathbf{d}$ is not used. Instead an SEQP calculation using the subproblem EQP$^{(k)}$ in Section 3.3 is made to determine $\mathbf{d}^{(k)}$. The use of a trust region in the LP subproblem (see below) is an essential feature in the calculation. The method is another example of a Newton-type method and shares the rapid local convergence properties.The idea has proved quite workable, as a recent software product SLIQUE of Byrd, Gould, Nocedal and Waltz [3] demonstrates.

4.6 Representing the Lagrangian Hessian $W^{(k)}$

An important issue for the development of an SQP code is how to represent the Hessian matrix $W^{(k)}$ that arises in the SQP subproblem. As defined in QP$^{(k)}$, it requires evaluation of all the Hessian matrices of f and c_i, $i = 1, 2, \ldots, m$ at $\mathbf{x}^{(k)}$, and their combination using the multipliers $\boldsymbol{\lambda}^{(k)}$. Writing code from which to evaluate second derivatives can be quite error prone, and in the past, this option has not always been preferred. However, the use of an exact $W^{(k)}$ matrix has been given new impetus through the availability of easy-to-use automatic differentiation within modelling languages such as AMPL, GAMS and TOMLAB (see Section 7). In large problems the exact $W^{(k)}$ may be a sparse matrix, which provides another reason to consider this option. On the other hand, it may be that the globalization strategy requires $W^{(k)}$ to be positive definite, in which case it will usually not be possible to use $W^{(k)}$ directly (except for certain 'convex' problems, W can be indefinite, even at the solution). However, if $W^{(k)}$ is readily and cheaply available, it is probably best to make use of it in some way, as this can be expected to keep the iteration count low. There are various ideas for modifying $W^{(k)}$ to obtain a positive definite matrix. One is to add a suitable multiple of a unit matrix to $W^{(k)}$. Another is to add outer products using active constraint gradients, as when using an augmented Lagrangian penalty function.

Otherwise, the simplest approach, generically referred to as *quasi-Newton* SQP, is to update a symmetric matrix $B^{(k)}$ which approximates $W^{(k)}$. In this method, $B^{(1)}$ is initialized to some suitable positive definite or semi-definite matrix (often a multiple of the unit matrix), and $B^{(k)}$ is updated after each iteration to build up information about second derivatives. A suitable strategy (e.g. Nocedal and Overton [36]) is usually based on evaluating difference vectors

$$\boldsymbol{\delta}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} \tag{4.11}$$

in $\mathbf{x}$, and

$$\boldsymbol{\gamma}^{(k)} = \boldsymbol{\nabla}_{\mathbf{x}}\mathcal{L}(\mathbf{x}^{(k+1)},\, \boldsymbol{\lambda}^{(k)}) - \boldsymbol{\nabla}_{\mathbf{x}}\mathcal{L}(\mathbf{x}^{(k)},\, \boldsymbol{\lambda}^{(k)}) \tag{4.12}$$

in the gradient of the Lagrangian function $\mathcal{L}(\mathbf{x},\, \boldsymbol{\lambda}^{(k)})$, using the latest available estimate $\boldsymbol{\lambda}^{(k)}$ of the multipliers. Then the updated matrix $B^{(k+1)}$ is chosen to satisfy the *secant condition* $B^{(k+1)}\boldsymbol{\delta}^{(k)} = \boldsymbol{\gamma}^{(k)}$.

There are many ways in which one might proceed. For small problems, where it is required to maintain a positive definite $B^{(k)}$ matrix, the BFGS formula (see [13]) might be used, in which case it is necessary to have $\boldsymbol{\delta}^{(k)T}\boldsymbol{\gamma}^{(k)} > 0$. It is not immediately obvious how best to meet this requirement in an NLP context, although a method suggested by Powell [40] has been used widely with some success. For large problems, some form of *limited memory update* is a practical proposition. The L-BFGS method, Nocedal [35], as implemented by Byrd, Nocedal and Schnabel [4] is attractive, although other ideas have also been tried. Another method which permits low costs is the low rank Hessian approximation $B = UU^T$ (Fletcher [15]), where U has relatively few columns. For ENLP, updating the reduced Hessian matrix $M \approx Z^TWZ$, $B = VMV^T$, using differences in reduced gradients, is appropriate, essentially updating the Hessian of the reduced objective function $F(\mathbf{t})$ in (3.5). However, this idea does not translate easily into the context of NLP with inequality constraints, due to the change in dimension of M when the number of active constraints changes.

An intermediate situation for large scale SQP is to update an approximation which takes the sparsity pattern of $W^{(k)}$ into account, and updates only the non-sparse elements. The LANCELOT project (see Conn, Gould and Toint [8] for many references) makes use of *partially separable* functions in which $B^{(k)}$ is the sum of various low dimensional *element Hessians*, for which the symmetric rank one update is used. Other sparsity respecting updates have also been proposed, for example Toint [44], Fletcher, Grothey and Leyffer [17], but the coding is complex, and there are some difficulties.

Various important conditions exist regarding rapid local convergence, relating to the asymptotic properties of WZ or Z^TWZ (see [13] for references). Significantly, low storage methods like L-BFGS do not satisfy these conditions, and indeed slow convergence is occasionally observed, especially when the true reduced Hessian $Z^{*T}W^*Z^*$ is ill-conditioned. For this reason, obtaining rapid local convergence when the null space dimension is very large is still a topic of research interest. Indeed the entire subject of how best to provide second derivative information in an SQP method is very much an open issue.

5 Globalization of NLP Methods

In this section we examine the transition from Newton type methods with rapid local convergence, such as the SQP method, to globally convergent methods suitable for incorporation into production NLP software. By globally convergent, we refer to the ability to converge to local solutions of an NLP problem from globally selected initial iterates which may be remote from any solution. This is not to be confused with the problem of guaranteeing to find global solutions of an NLP problem in the sense of the best local solution, which is computationally impractical for problems of any size (perhaps >40 variables, say), unless the problem has some special convexity properties, which is rarely the case outside of LP and QP. We must also be aware that NLP problems may have no solution, mainly due to the constraints being infeasible (that is, no feasible point exists). In this case the method should ideally be able to indicate that this is the case, and not spend an undue amount of time in searching for a non-existent solution. In practice even this is an unrealistic aim, akin to that of finding a global minimizer of some measure of constraint infeasibility. What is practical is to locate a point which is *locally infeasible* in the sense that the first order Taylor series approximation to the constraints set is infeasible at that point. Again the main requirement is that the method should be able to converge rapidly to such a point, and exit with a suitable indication of local infeasibility. Another possibility, which can be excluded by bounding the feasible region, is that the NLP is unbounded, that is $f(\mathbf{x})$ is not bounded below on the feasible region, or that there are no KT points in the feasible region. Again the software has to recognize the situation and terminate accordingly.

Ultimately the aim is to be able to effectively solve NLP problems created by scientists, engineers, economists etc., who have a limited background in optimization methods. For this we must develop general purpose software which is

- Efficient
- Reliable
- Well documented
- Thoroughly tested
- Flexible
- Easy to use, and has
- Large scale capacity.

Efficiency and reliability are self evident, as is being well-documented, Being thoroughly tested involves monitoring the behaviour on test problems (e.g. CUTE [2]) which encompass a wide range of possible situations, including problems with no solution as referred to above. Flexibility includes the ability to specify simple bounds, linear constraints, etc., to take account of sparsity in the formulation, and to be able to make warm starts from the solution of a previously solved problem. It can also include the flexibility to decide whether or not to supply second derivatives (indeed some codes have been developed which require no derivative information to be supplied, but these are very limited in the size of problem that can be solved, and the accuracy that can be achieved). By easy to use we envisage issues such as the provision of default parameter values which do not need tuning by the user, and access to the software via modelling languages like AMPL, GAMS or TOMLAB. Large scale capacity is required for the software to make a significant impact: even run of the mill applications of optimization now have 1000+ variables and/or constraints and are computationally intractable without the use of sparse matrix techniques, or special iterative methods such as Conjugate Gradients.

5.1 Penalty and Barrier Functions

From an historical perspective, almost all general purpose NLP solvers until about 1996 aimed to promote global convergence by constructing an auxiliary function from $f(\mathbf{x})$ and $\mathbf{c}(\mathbf{x})$ known variously as a penalty, barrier, or merit function. In the earlier days, the idea was to apply successful existing techniques for unconstrained minimization to the auxiliary function, in such a way as to find the solution of the NLP problem. Later, there came the idea of using the auxiliary function to decide whether or not to accept the step given by the SQP method, hence the term merit function.

For an ENLP problem, an early idea was the Courant [9] penalty function

$$\phi(\mathbf{x};\sigma) = f(\mathbf{x}) + \tfrac{1}{2}\sigma\mathbf{c}^T\mathbf{c} = f(\mathbf{x}) + \tfrac{1}{2}\sigma\sum_{i=1}^{m} c_i^2 \qquad \text{where} \qquad \mathbf{c} = \mathbf{c}(\mathbf{x}), \tag{5.1}$$

where $\sigma > 0$ is a parameter. The $\mathbf{c}^T\mathbf{c}$ term 'penalizes' points which violate the constraints, and σ determines the strength of the penalty. Let a minimizer, $\mathbf{x}(\sigma)$ say, of $\phi(\mathbf{x};\sigma)$ be found by some technique for unconstrained minimization. Then we choose a sequence of values of $\sigma \to \infty$, for example $\sigma = \{1,\, 10,\, 100,\, \ldots\}$ and observe the behaviour of $\mathbf{x}(\sigma)$. Under some assumptions, it can be shown that the NLP solution is given by $\mathbf{x}^* = \lim_{\sigma\to\infty}\mathbf{x}(\sigma)$. A simple modification for the NLP problem is to include terms $(\min(c_i,\, 0))^2$ in the summation for any inequality constraints $c_i(\mathbf{x}) \geq 0$. In practice the methods are slow as compared with SQP techniques, and more seriously, suffer serious effects due to ill-conditioning of the Hessian matrix of ϕ as $\sigma \to \infty$.

Another early idea for NLP with inequality constraints was the Frisch [24] *log barrier function*

$$\phi(\mathbf{x};\mu) = f(\mathbf{x}) - \mu\sum_{i=1}^{m}\log_e c_i(\mathbf{x}), \tag{5.2}$$

where $\mu > 0$ is a parameter. In this case we require $\mathbf{x}$ to lie in the interior of the feasible region where $c_i(\mathbf{x}) > 0$, $i = 1,\, 2,\, \ldots, m$, so that the log terms are well defined. Then each term $-\log_e c_i(\mathbf{x})$ approaches $+\infty$ as $\mathbf{x}$ approaches the boundary of the feasible region, and creates a 'barrier' which prevents iterates from escaping out of the feasible region. The parameter μ determines the extent to which the barrier extends into the interior of the feasible region. For any fixed value of μ we again find $\mathbf{x}(\mu)$ to minimize $\phi(\mathbf{x};\mu)$. Then we choose a sequence of values of $\mu \to 0$, for example $\mu = \{1,\, \frac{1}{10},\, \frac{1}{100}, \ldots\}$ and observe the behaviour of $\mathbf{x}(\mu)$. Under some assumptions, it can be shown that $\mathbf{x}^* = \lim_{\mu\to 0}\mathbf{x}(\mu)$.

Again the methods are slow and suffer from ill-conditioning of the Hessian. Moreover the need to find a strictly feasible starting point is usually a difficult problem in its own right. More recently the log barrier function has been used as a merit function in conjunction with interior point methods.

5.2 Multiplier Penalty and Barrier Functions

From around 1969 onwards, a more satisfactory class of auxiliary function came into use, involving an additional parameter $\boldsymbol{\lambda} \in \mathbb{R}^m$ which could be interpreted as an estimate of the Lagrange multiplier vector. For ENLP the Lagrangian function is augmented with a penalty term giving

$$\phi(\mathbf{x};\, \boldsymbol{\lambda}, \sigma) = f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{c} + \tfrac{1}{2}\sigma \mathbf{c}^T \mathbf{c} \qquad \text{where} \qquad \mathbf{c} = \mathbf{c}(\mathbf{x}), \tag{5.3}$$

sometimes referred to as the *augmented Lagrangian function* (Hestenes [30], Powell [39]). We note that

$$\boldsymbol{\nabla}_{\mathbf{x}}\phi = \mathbf{g} - A\boldsymbol{\lambda} + \sigma A\mathbf{c}, \tag{5.4}$$

so that $\boldsymbol{\nabla}\phi(\mathbf{x}^*;\, \boldsymbol{\lambda}^*, \sigma) = \mathbf{0}$. Moreover if the local solution is regular and satisfies second order sufficient conditions then $\nabla^2\phi(\mathbf{x}^*;\, \boldsymbol{\lambda}^*, \sigma)$ is positive definite if σ is chosen sufficiently large (see [13]). Thus it is not necessary to drive σ to infinity, in contrast to the methods of the previous subsection, and this avoids the worst effects of the ill-conditioning. Unfortunately $\boldsymbol{\lambda}^*$ is not known a-priori, so a sequential minimization technique must be used. In an outer iteration a sequence of parameters $\boldsymbol{\lambda}^{(k)} \to \boldsymbol{\lambda}^*$ is chosen, whilst in the inner iteration, a minimizer $\mathbf{x}(\boldsymbol{\lambda}^{(k)}, \sigma)$ of $\phi(\mathbf{x};\, \boldsymbol{\lambda}^{(k)}, \sigma)$ is found by means of an unconstrained minimization technique. The behaviour of $\mathbf{x}(\boldsymbol{\lambda}^{(k)}, \sigma)$ is observed as $\boldsymbol{\lambda}^{(k)}$ changes. If the iteration is not converging to $\mathbf{x}^*$, it may be necessary to increase σ at some stage. To update $\boldsymbol{\lambda}^{(k)}$, a formula

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} - (A^T W A)^{-1} \mathbf{c}^{(k)} \tag{5.5}$$

derived fron the SQP iteration formula (3.19) may be used. For large σ this may be approximated by the scheme

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} - \sigma \mathbf{c}^{(k)}, \tag{5.6}$$

see [13].

A multiplier penalty function for the NLP (1.1) with inequality constraints is

$$\phi(\mathbf{x};\, \boldsymbol{\lambda}, \sigma) = f(\mathbf{x}) + \sum_i \begin{cases} -\lambda_i c_i + \frac{1}{2}\sigma c_i^2 & \text{if} \quad c_i \le \lambda_i/\sigma \\ -\frac{1}{2}\lambda_i^2/\sigma & \text{if} \quad c_i \ge \lambda_i/\sigma \end{cases} \tag{5.7}$$

suggested by Rockafellar [43]. The piecewise term does not cause any discontinuity in first derivatives, and any second derivative discontinuities occur away from the solution. Otherwise its use is similar to that in (5.3) above.

A multiplier based modification of the Frisch barrier function due to Polyak [38] is

$$\phi(\mathbf{x};\, \boldsymbol{\lambda}, \mu) = f(\mathbf{x}) - \mu \sum_{i=1}^{m} \lambda_i \log_e(c_i/\mu + 1), \tag{5.8}$$

in which the boundary occurs where $c_i = -\mu$, which is strictly outside the feasible region. Thus the discontinuity of the Frisch function at the solution $\mathbf{x}^*$ is moved away into the infeasible region. We note that

$$\boldsymbol{\nabla}_{\mathbf{x}}\phi(\mathbf{x}^*;\, \boldsymbol{\lambda}^*, \mu) = \mathbf{g}^* - \mu \sum_{i=1}^{m} \frac{\lambda_i^* \mathbf{a}_i^*/\mu}{(c_i^*/\mu + 1)} = \mathbf{0}$$

using KT conditions (including complementarity). If the solution $\mathbf{x}^*$ is regular, and sufficient conditions hold, with strict complementarity, then it can also be shown that $\nabla^2_{\mathbf{x}}\phi(\mathbf{x}^*;\, \boldsymbol{\lambda}^*, \mu)$ is positive definite if μ is sufficiently small. Thus a suitable fixed value of $\mu > 0$ can be found and the worst effects of ill-conditioning are avoided. Both the Rockafellar and Polyak functions are used in a sequential manner with an outer iteration in which $\boldsymbol{\lambda}^{(k)} \to \boldsymbol{\lambda}^*$.

All the above proposals involve sequential unconstrained minimization, and as such, are inherently less effective than the SQP method, particularly in regard to the rapidity of local convergence. Errors in $\boldsymbol{\lambda}^{(k)}$ are reflected in an errors of similar order in $\mathbf{x}(\boldsymbol{\lambda}^{(k)})$, which is not the case for the SQP method. Many other auxiliary functions have been suggested for solving NLP or ENLP problems in ways related to the above. In a later section we shall investigate so-called exact penalty functions which avoid the sequential unconstrained minimization aspect.

A major initiative to provide robust and effective software with large scale capability based on the augmented Lagrangian function was the LANCELOT code of Conn, Gould and Toint (see [8] for references). The code applies to an NLP in the form

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{c}(\mathbf{x}) = 0 \\ & \mathbf{l} \le \mathbf{x} \le \mathbf{u}. \end{array} \tag{5.9}$$

which treats simple bounds explicitly but assumes that slack variables have been added to any other inequality constraints. In the inner iteration, the augmented Lagrangian function (5.3) is minimized subject to the simple bounds $\mathbf{l} \le \mathbf{x} \le \mathbf{u}$ on the variables. A potential disadvantage of this approach for large scale computation is that the Hessian $\nabla^2_{\mathbf{x}}\phi(\mathbf{x};\, \boldsymbol{\lambda}^{(k)}, \sigma)$ is likely to be much less sparse than the Hessian $W^{(k)}$ in the SQP method. To avoid this difficulty, LANCELOT uses a simple bound minimization technique based on the use of the preconditioned conjugate gradient method, and solves the subproblem to lower accuracy when $\boldsymbol{\lambda}^{(k)}$ is inaccurate. It also uses an innovative idea of building the Hessian from a sum of elementary Hessians through the concept of group partial separability. LANCELOT has been successfully used to solve problems with upwards of 10^4 variables, particularly those with large dimensional null spaces arising for example from the discretization of a partial differential equation. It is less effective for problems with low dimensional null spaces and does not take advantage of any linear constraints.

5.3 Augmented Lagrangians with SQP

A fruitful way to take advantage of linear constraints has been to merge the globalization aspect of the augmented Lagrangian function with the rapid local convergence of the SQP method. This was the motivation of the very successful MINOS code of Murtagh and Saunders [34] , which was arguably the first SQP-like NLP solver with large scale capability. In fact MINOS was influenced by a method due to Robinson [42] which is not an SQP method in the strict sense described above, but is closely related to it. We describe Robinson's method in the context of the NLP problem (1.1). The method generates a sequence of major iterates $\mathbf{x}^{(k)}$, $\boldsymbol{\lambda}^{(k)}$ and solves the LCP problem

$$\text{LCP}^{(k)} \left\{ \begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & f(\mathbf{x}) - \boldsymbol{\lambda}^{(k)T}(\mathbf{c}(\mathbf{x}) - \mathbf{s}) \\ \text{subject to} & \mathbf{s} = \mathbf{c}^{(k)} + A^{(k)T}(\mathbf{x} - \mathbf{x}^{(k)}) \geq \mathbf{0}, \end{array} \right.$$

where $\mathbf{s} = \mathbf{s}(\mathbf{x}, \mathbf{x}^{(k)}) = \mathbf{c}^{(k)} + A^{(k)T}(\mathbf{x} - \mathbf{x}^{(k)})$ is the first order Taylor series approximation about the current point, and the quantity $\mathbf{c}(\mathbf{x}) - \mathbf{s}$ may be thought of as the *deviation from linearity*. The solution and multipliers of LCP$^{(k)}$ then become the iterates $\mathbf{x}^{(k+1)}$, $\boldsymbol{\lambda}^{(k+1)}$ for the next major iteration. The method differs from SQP, firstly in that LCP$^{(k)}$ cannot be solved finitely, which is a disadvantage, and secondly that second derivatives are not required, which is an advantage. Robinson intended that LCP$^{(k)}$ should be solved by a reduced Hessian quasi-Newton method. If a Taylor expansion of the objective function in LCP$^{(k)}$ about the current point is made, then it agrees with that of SQP$^{(k)}$ up to and including second order terms. Also the method has the same fixed point property as SQP that if $\mathbf{x}^{(k)}$, $\boldsymbol{\lambda}^{(k)}$ is equal to $\mathbf{x}^*$, $\boldsymbol{\lambda}^*$, then $\mathbf{x}^*$, $\boldsymbol{\lambda}^*$ is the next iterate, and the process terminates. Consequently the method has the same rapid local convergence properties as the SQP method, assuming that the LCP$^{(k)}$ subproblem is solved sufficiently accurately. However there is no global convergence result available, for example there is no mechanism to force the iterates $\mathbf{x}^{(k)}$ to accumulate at a feasible point.

The MINOS code attempts to mitigate the lack of a global convergence property by augmenting the objective function in LCP$^{(k)}$ with a squared penalty term. As with LANCELOT, the method is applicable to an NLP in the form (5.9), and the LCP subproblem that is solved on the k-th major iteration is

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & f(\mathbf{x}) - \boldsymbol{\lambda}^{(k)T}(\mathbf{c}(\mathbf{x}) - \mathbf{s}) + \frac{1}{2}\sigma(\mathbf{c}(\mathbf{x}) - \mathbf{s})^T(\mathbf{c}(\mathbf{x}) - \mathbf{s}) \\ \text{subject to} & \mathbf{s} = \mathbf{c}^{(k)} + A^{(k)T}(\mathbf{x} - \mathbf{x}^{(k)}) \geq \mathbf{0} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}. \end{array}$$

In the original source, MINOS refers to the active set method used to solve this LCP subproblem, and MINOS/AUGMENTED refers to the major iterative procedure for solving (5.9). However it is more usual now to refer to the NLP solver by MINOS. The code has sparse matrix facilities, and also allows 'linear variables' to be designated, so allowing the use of a smaller Hessian approximation. MINOS was probably the first SQP-type code with the capability to solve large scale problems, and as such has been very successful and is still in use.

A development of MINOS is the SNOPT code of Gill, Murray and Saunders [25] which first appeared in about 1992. In place of an LCP subproblem it solves a QP subproblem, but using an approximate Hessian matrix. Slack variables $\mathbf{s}$ are explicitly included, and each iteration involves the solution of a line search subproblem based on the MINOS augmented Lagrangian

$$\phi(\mathbf{x},\mathbf{s},\boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^{(k)T}(\mathbf{c}(\mathbf{x})-\mathbf{s}) + \tfrac{1}{2}(\mathbf{c}(\mathbf{x})-\mathbf{s})^T D(\mathbf{c}(\mathbf{x})-\mathbf{s}), \tag{5.10}$$

where $D = \operatorname{diag}\sigma_i$ is a diagonal matrix of penalty parameters. The entire triple $\mathbf{x}^{(k)}$, $\mathbf{s}^{(k)}$, $\boldsymbol{\lambda}^{(k)}$ is varied in the line search. Various treatments of the Hessian approximation are possible, depending for example on the size of the problem. Another difference from MINOS is the use of 'elastic mode' (essentially the l_1 penalty function of the next subsection) to resolve significant deviations from infeasibility. It is impossible to do justice to all the features of the code, and the reader is referred to the comprehensive description in [25], although it is quite likely that further development of the code has taken place. For NLP problems in which the null space dimension is not too large, up to 1000 say, SNOPT is currently amongst the best currently available NLP solvers (see Byrd, Gould, Nocedal and Waltz [3]).

5.4 The l_1 Exact Penalty Function

The penalty and barrier functions in Sections 5.1 and 5.2 are inherently sequential, that is the solution of the NLP problem is obtained by a sequence of unconstrained minimization calculations. It is however possible to construct a so-called *exact penalty function*, that is a penalty function of which $\mathbf{x}^*$, the solution of the NLP problem, is a local minimizer. It is convenient here to consider an NLP problem in the form

$$\begin{array}{ll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & f(\mathbf{x}) \\ \text{subject to} & c(\mathbf{x}) \le \mathbf{0}. \end{array} \tag{5.11}$$

The most well known exact penalty function (Pietrzykowski [37], see [13] for more references) is the l_1 *exact penalty function* (l_1EPF)

$$\phi(\mathbf{x};\sigma) = f(\mathbf{x}) + \sigma\|\mathbf{c}^+(\mathbf{x})\|_1 = f(\mathbf{x}) + \sigma\sum_{i=1}^{m} c_i^+(\mathbf{x}), \tag{5.12}$$

where $c_i^+ = \max(c_i, 0)$ is the amount by which the $i-$th constraint is violated. The parameter σ controls the strength of the penalty.

First we consider optimality conditions for a local minimizer of $\phi(\mathbf{x}, \sigma)$. The function is nonsmooth due to the discontinuity in derivative of $\max(c_i, 0)$ at zero, so we cannot refer to the stationary point condition of Section 2.2. In fact we proceed very much as in Section 4.2, by defining the set of active constraints as in (4.5), and the set of *infeasible constraints*

$$\mathcal{I}(\mathbf{x}) = \{i \,|\, c_i(\mathbf{x}) > 0\}. \tag{5.13}$$

Infeasible constraints are assigned a multiplier $\lambda_i^* = \sigma$. Constraints that are neither active nor infeasible at $\mathbf{x}^*$ play no part in the conditions and can be ignored. As before we assign a zero multiplier to them. We observe that if $\mathbf{x}^*$ minimizes $\phi(\mathbf{x}, \sigma)$, then it must solve the problem

$$\begin{array}{lll} \underset{\mathbf{x}\in\mathbb{R}^n}{\text{minimize}} & f(\mathbf{x}) + \sigma \sum_{i\in\mathcal{I}} c_i(\mathbf{x}) & \\ \text{subject to} & c_i(\mathbf{x}) \le 0 & i \in \mathcal{A}. \end{array} \tag{5.14}$$

since the penalty term comes only from the infeasible constraints. Therefore if we make the same regularity assumption that the vectors $\mathbf{a}_i^*$, $i \in \mathcal{A}^*$ are linearly independent, then the KT conditions for this problem are also necessary for a minimizer of (5.12). Moreover, if we perturb the right hand side of a constraint $i \in \mathcal{A}$ in (5.14) by a sufficiently small $\varepsilon_i > 0$, $\varepsilon_j = 0$, $j \neq i$, we make constraint i infeasible, but do not change the status of any other constraints. This causes an increase of $\sigma\varepsilon_i$ in the penalty term. Moreover the change in $f(\mathbf{x}) + \sigma \sum_{i\in\mathcal{I}} c_i(\mathbf{x})$ to first order is $-\lambda_i^*\varepsilon_i$. (The negative sign holds because of the sign change in (5.11)). Hence the change in $\phi(\mathbf{x}, \sigma)$ to first order is $\varepsilon_i(\sigma - \lambda_i^*)$. If $\lambda_i^* > \sigma$ then ϕ is reduced by the perturbation, which contradicts the optimality of $\mathbf{x}^*$ in the l_1 EPF. Thus the condition $\lambda_i^* \le \sigma$, $i \in \mathcal{A}^*$ is also necessary. This result tells us that unless the penalty parameter σ is sufficiently large, a local minimizer will not be created. We can therefore summarize the first order necessary conditions as

$$\mathbf{g}^* + A^*\boldsymbol{\lambda}^* = \mathbf{g}^* + \sum_{i=1}^{m} \mathbf{a}_i^*\lambda_i^* = \mathbf{0} \tag{5.15}$$

$$\left.\begin{array}{r} 0 \le \lambda_i^* \le \sigma \\ c_i^* < 0 \Rightarrow \lambda_i^* = 0 \\ c_i^* > 0 \Rightarrow \lambda_i^* = \sigma \end{array}\right\} \quad i = 1, 2, \ldots, m. \tag{5.16}$$

If $\Rightarrow$ can be replaced by $\Leftrightarrow$ then *strict complementarity* is said to hold. Second order necessary conditions are the same as for (5.14), that is $Z^{*T}W^*Z^*$ is positive semi-definite. Sufficient are that first order conditions hold, with strict complementarity, and $Z^{*T}W^*Z^*$ is positive definite.

We see that these conditions are very similar to those for solving the NLP problem (1.1). In fact if $\mathbf{x}^*$ satisfies second order sufficient conditions for a solution of (1.1), and $\sigma > \|\boldsymbol{\lambda}^*\|_\infty$, then $\mathbf{x}^*$ is also a minimizer of the l_1EPF.

5.5 SQP with the l_1EPF

The SQP method first came into prominent use when used in conjunction with the l_1EPF in papers by Han [29] and Powell [40]. The vector $\mathbf{d}^{(k)}$ generated by the SQP subproblem QP$^{(k)}$ is regarded as a direction of search, and the l_1EPF is used as a merit function, so that the next iterate is $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)}\mathbf{d}^{(k)}$, with $\alpha^{(k)}$ being chosen to obtain a sufficient reduction in $\phi(\mathbf{x}, \sigma)$. For this to be possible requires that $\mathbf{d}^{(k)}$ is a descent direction at $\mathbf{x}^{(k)}$ for $\phi(\mathbf{x}, \sigma)$. If the Hessian $W^{(k)}$ (or its approximation) is positive definite, it is possible to ensure that this is the case, if necessary by increasing σ. Early results with this technique were quite promising, when compared with sequential unconstrained penalty and barrier methods. However the use of a nonsmooth merit function is not without its difficulties. In particular the discontinuities in derivative cause 'curved valleys', with sides whose steepness depends on the size of σ. If σ is large, the requirement to monotonically improve ϕ on every iteration can only be achieved by taking correspondingly small steps, leading to slow convergence. Unfortunately, increasing σ to obtain descent exacerbates this situation.

A way round this is the Sequential l_1 Quadratic Programming (Sl_1QP) method of Fletcher [11]. The idea (also applicable to an l_∞ exact penalty function) is to solve a subproblem which more closely models the l_1EPF, by moving the linearized constraints into the objective function, in an l_1 penalty term. Thus the l_1QP subproblem is

$$\begin{array}{ll} \underset{\mathbf{d}\in\mathbb{R}^n}{\text{minimize}} & \mathbf{g}^{(k)T}\mathbf{d} + \tfrac{1}{2}\mathbf{d}^T W^{(k)}\mathbf{d} + \sigma\|(\mathbf{c}^{(k)} + A^{(k)T}\mathbf{d})^+\|_1 \\ \text{subject to} & \|\mathbf{d}\|_\infty \leq \rho. \end{array} \tag{5.17}$$

It it necessary that σ is sufficiently large as discussed in Section 5.4, and also below. The restriction on $\mathbf{d}$ is the trust region constraint and ρ is the trust region radius. Solving the subproblem ensures descent, and quite strong results regarding global convergence can be proved by using standard trust region ideas. The use of an l_∞ trust region (a 'box constraint') fits conveniently into a QP type framework.

Even so, there are still some issues to be resolved. Firstly, (5.17) is not a QP in standard form due to the presence of l_1 terms in the objective, although it is still a problem that can be solved in a finite number of steps. Ideally a special purpose l_1QP solver with sparse matrix capabilities would be used. This would enable an efficient l_1 piecewise quadratic line search to be used within the solver. Unfortunately a fully developed code of this type is not easy to come by. The alternative is to transform (5.17) into a regular QP by the addition of extra variables. For example a constraint $\mathbf{l} \leq \mathbf{c}(\mathbf{x}) \leq \mathbf{u}$ can be written as $\mathbf{l} \leq \mathbf{c}(\mathbf{x}) - \mathbf{v} + \mathbf{w} \leq \mathbf{u}$ where $\mathbf{v} \geq \mathbf{0}$ and $\mathbf{w} \geq \mathbf{0}$ are auxiliary variables, and a penalty term of the form $\sigma\|\mathbf{v} + \mathbf{w}\|_1$, which is linear in $\mathbf{v}$ and $\mathbf{w}$, would then be appropriate. However $2m$ extra variables need be added, which is cumbersome, and the benefit of the piecewise quadratic line search is not obtained. A related idea is to use the SLP-EQP idea of Fletcher and Sainz de la Maza [22], referred to in Section 4.5. In this case an l_1LP subproblem would be used to find an active set and multipliers, followed by an EQP calculation to obtain the step $\mathbf{d}^{(k)}$. As above, the l_1LP subproblem can be converted to an LP problem by the addition of extra variables, and this allows fast large scale LP software to be used.

It is also not easy for the user to choose a satisfactory value of σ. If it is chosen too small, then a local minimizer may not be created, if too large then the difficulties referred to above become apparent. There is also a possibility of the *Maratos effect* [33] occurring, in which, close to the solution, the Newton-type step given by the SQP method increases ϕ and cannot be accepted if monotonic improvement in ϕ is sought.Thus the expected rapid local convergence is not realised. More recently, ideas for circumventing these difficulties have been suggested, including *second order corrections* [12], the *watchdog technique* [7], and a *non-monotonic line search* [27].

6 Filter Methods

Filtter methods were introduced in response to the perceived difficulties in using penalty function methods for globalization, that is the difficulty of choosing suitable penalty parameters, the inefficiency of sequential methods, and the slow convergence associated with monotonic minimization methods, particularly in the case of nonsmooth exact penalty functions. Along with this is the observation that the basic SQP method is able to quickly solve a significant proportion of test problems without the need for modifications to induce global convergence. Thus the goal of filter methods is to provide global optimization safeguards that allow the full SQP step to be taken much more often. In this section we describe the main ideas and possible pitfalls, and discuss the way in which a global convergence result for a filter method has been constructed.

A penalty function is an artefact to combine two competing aims in NLP, namely the minimization of $f(\mathbf{x})$ and the need to obtain feasibility with respect to the constraints. The latter aim can equivalently be expressed as the minimization of some measure $h(\mathbf{c}(\mathbf{x}))$ of constraint violation. For example, in the context of (5.11) we could define $h(\mathbf{c}) = \|\mathbf{c}^+\|$ in some convenient norm. Thus, in a filter method, we view NLP as the resolution of two competing aims of minimizing $f(\mathbf{x})$ and $h(\mathbf{c}(\mathbf{x}))$. This is the type of situation addressed by Pareto (multi-objective) optimization, but in our context the minimization of $h(\mathbf{c}(\mathbf{x}))$ has priority, in that it is essential to find a Pareto solution that corresponds to a feasible point. However it is useful to borrow the concept of *domination* from multi-objective optimization. Let $\mathbf{x}^{(k)}$ and $\mathbf{x}^{(l)}$ be two points generated during the progress of some method. We say that $\mathbf{x}^{(k)}$ dominates $\mathbf{x}^{(l)}$ if and only if $h^{(k)} \le h^{(l)}$ and $f^{(k)} \le f^{(l)}$. That is to say, there is no reason to prefer $\mathbf{x}^{(l)}$ on the basis of either measure. Now we define a *filter* to be a list of pairs $(h^{(k)}, f^{(k)})$ such that no pair dominates any other. As the algorithm progresses, a filter is built up from all the points that have been sampled by the algorithm. A typical filter is shown in Figure 2, where the shaded region shows the region dominated by the filter entries (the outer vertices of this shaded region). The contours of the l_1 exact penalty function would be straight lines with slope $-\sigma$ on this plot, indicating that at least for

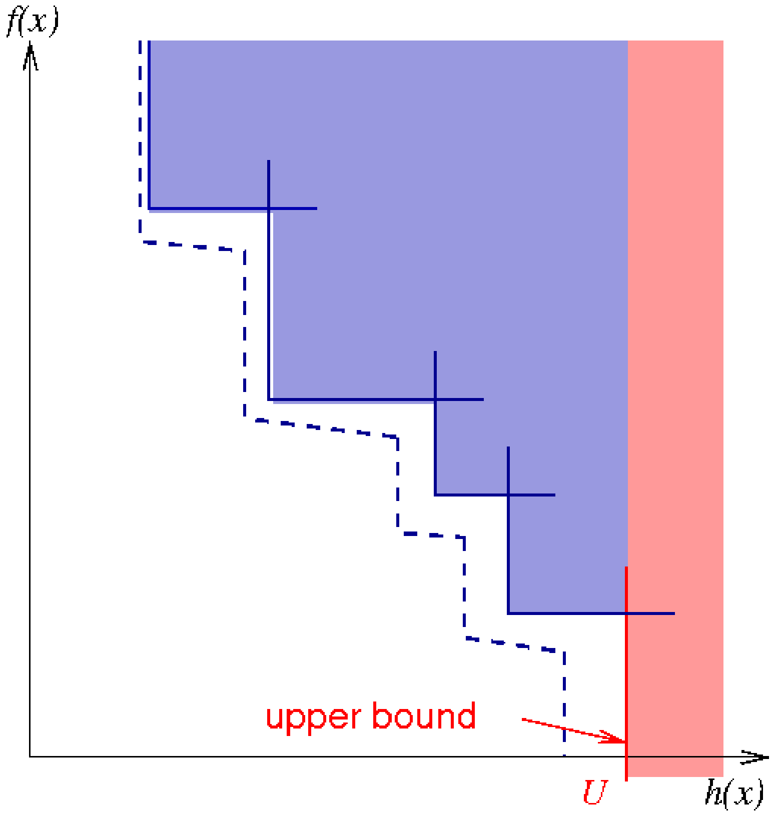

Figure 2: A Typical Filter Diagram

a single entry, the filter provides a less restrictive acceptance condition than the penalty function.

6.1 SQP Filter Methods

Filter methods were first introduced in the context of trust region SQP methods in 1997 by Fletcher and Leyffer [18], making use of a subproblem

$$\mathrm{QP}^{(k)}(\rho)\left\{\begin{array}{ll} \underset{\mathbf{d}\in\mathbb{R}^n}{\text{minimize}} & \frac{1}{2}\mathbf{d}^T W^{(k)}\mathbf{d} + \mathbf{d}^T\mathbf{g}^{(k)} \\ \text{subject to} & \mathbf{c}^{(k)} + A^{(k)T}\mathbf{d} \geq \mathbf{0}. \\ & \|\mathbf{d}\|_\infty \leq \rho, \end{array}\right.$$

obtained by adding a trust region constraint to $\mathrm{QP}^{(k)}$. A *trust region* (see [13]) is a heuristic aimed at restricting the step size to lie in a region in which there is 'adequate' agreement between the true functions and their Taylor series approximations. The trust region radius ρ is adjusted during or after each iteration to achieve this aim.

A first attempt at a filter algorithm might go as follows. On iteration $k = 1$ the filter $\mathcal{F}^{(1)}$ is empty. On iteration k we solve $\mathrm{QP}^{(k)}(\rho)$ giving a step $\mathbf{d}$ and evaluate $f = f(\mathbf{x}^{(k)}+\mathbf{d})$ and $h = h(\mathbf{c}(\mathbf{x}^{(k)} + \mathbf{d}))$. If the resulting pair $(h,\, f)$ is acceptable to $\mathcal{F}^{(k)}$ (that is, it is not dominated by any of the entries in $\mathcal{F}^{(k)}$), then we update $\mathbf{x}^{(k)}$ and $\boldsymbol{\lambda}^{(k)}$ as described in Section 4.4. We also update the filter, adding the new pair $(h,\, f)$ and removing any entries that are dominated by it. Possibly we might also increase the trust region radius. If, on the other hand, the pair is not acceptable, then we reject it, reduce the trust region radius, and re-solve $\mathrm{QP}^{(k)}(\rho)$.

There are various ways in which this simple approach can fail, or become unsatisfactory. One is that unacceptably large violations in the constraints may occur. This is readily handled by imposing an upper bound U on constraint violations, and initializing $\mathcal{F}^{(1)}$ to $(U,\, -\infty)$ (see Figure 2). More serious is the possibility that, if the current point $\mathbf{x}^{(k)}$ is infeasible ($h^{(k)} > 0$), and if ρ is reduced sufficiently in the algorithm, then the constraints of $\mathrm{QP}^{(k)}(\rho)$ can become incompatible, and the algorithm stalls. In this case our approach has been to enter a *feasibility restoration* phase, in which a different SQP-like algorithm (see Fletcher and Leyffer [19] for a filter-like approach) is invoked to find a new acceptable point $\mathbf{x}^{(k+1)}$ for which the TR subproblem is solvable. Of course, we cannot predict the effect on $f(\mathbf{x})$ and we must be prepared for it to increase. Another feature that the algorithm lacks is any sense of *sufficient reduction* in either f or h, as is used in other convergence proofs. For instance we would not like new pairs to become arbitrary close to existing filter entries, because this might allow convergence to a non-KT point. With this in mind we strengthen the acceptance condition by adding a *filter envelope* around the current filter entries, so as to extend the set of unacceptable points (see Figure 2). Most recently we use the *sloping envelope* of Chin [5], Chin and Fletcher [6], in which acceptability of a pair $(h,\, f)$ with respect to a filter $\mathcal{F}$ is defined by

$$h \le \beta h_i \quad \text{or} \quad f \le f_i - \gamma h \qquad \forall (h_i,\, f_i) \in \mathcal{F} \tag{6.1}$$

where β and γ are constants in $(0,\, 1)$. Typical values might be $\beta = 0.9$ and $\gamma = 0.01$. (In earlier work we used $f \le f_i - \gamma h_i$ for the second test, giving a rectangular envelope. However, this allows the possibility that $(h,\, f)$ dominates $(h_i,\, f_i)$ but the envelope of $(h,\, f)$ does *not* dominate the envelope of $(h_i,\, f_i)$, which is undesirable.)

During testing of the filter algorithm, another less obvious disadvantage became apparent. Say the current filter contains an entry $(0,\, f_i)$ where f_i is relatively small. If, subsequently, feasibility restoration is invoked, it may be impossible to find an acceptable point which is not dominated by $(0,\, f_i)$. Most likely, the feasibility restoration phase then converges to a feasible point that is not a KT point. We refer to $(0,\, f_i)$ as a *blocking entry*. We were faced with two possibilities. One is to allow the removal of blocking entries on emerging from feasibility restoration. This we implemented in the first code, reported by Fletcher and Leyffer [18]. To avoid the possibility of cycling, we reduce the upper bound when a blocking entry is removed. We did not attempt to provide a global convergence proof for this code, although it may well be possible to do so. Subsequently it became clear that other heuristics in the code were redundant and further work resulted in a related filter algorithm (Fletcher, Leyffer and Toint [20]) for which a convergence proof can be given. In this algorithm we resolve the difficulty over blocking by not including all accepted points in the filter. This work is described in the next section. However, the earlier code proved very robust, and has seen widespread use. It shows up quite well on the numbers of function and derivative counts required to solve a problem, in comparison say with SNOPT. Actual computing times are less competitive, probably because the QP solver used by SNOPT is more efficient. It has the same disadvantage as SNOPT that it is inefficient for large null space problems. Otherwise, good results were obtained in comparison with LANCELOT and an implementation of the l_1EPF method.

6.2 A Filter Convergence Proof

In this section an outline is given of the way in which a global convergence proof has been developed for the trust region filter SQP method. The theory has two aspects: how to force $h^{(k)} \to 0$, and how to minimize $f(\mathbf{x})$ subject to $h(\mathbf{c}(\mathbf{x})) = 0$.

First we review existing trust region SQP convergence theory for *unconstrained* optimization. At any non-KT point $\mathbf{x}^{(k)}$ and radius ρ in QP$^{(k)}(\rho)$, we define the *predicted reduction*

$$\Delta q = q(\mathbf{0}) - q(\mathbf{d}) = -\mathbf{g}^{(k)T}\mathbf{d} - \tfrac{1}{2}\mathbf{d}^T W^{(k)}\mathbf{d} > 0 \tag{6.2}$$

and the *actual reduction*

$$\Delta f = f^{(k)} - f(\mathbf{x}^{(k)} + \mathbf{d}). \tag{6.3}$$

As $\rho \to 0$, and with suitable assumptions, so $\mathbf{d} \to \mathbf{0}$, $\Delta q \sim \rho\|\mathbf{g}^{(k)}\|$ and $\Delta f/\Delta q \to 1$. It follows for sufficiently small ρ that the inequality

$$\Delta f \geq \sigma \Delta q \tag{6.4}$$

is valid. This is referred to as the *sufficient reduction* condition. In the TR algorithm of Figure 4 (with the feasibility restoration and filter boxes stripped out) we essentially choose $\rho^{(k)}$ as large as possible (within a factor of 2) subject to (6.4) holding. If the gradients $\mathbf{g}^{(k)}$ are accumulating at a value $\mathbf{g}^{\infty} \neq \mathbf{0}$, then $\rho^{(k)}$ is uniformly bounded away from zero, and it follows that $\Delta f^{(k)} \geq \sigma \Delta q^{(k)} \sim \sigma \rho^{(k)}$. Summing over all k shows that $f^{(k)} \to -\infty$ which is a contradiction. Thus the gradients can only accumulate at $\mathbf{0}$ and any accumulation point $\mathbf{x}^{\infty}$ is stationary.

We aim to make use of these ideas in an NLP context. However there is a difficulty when $h^{(k)} > 0$ that $\Delta q < 0$ may be possible. This is illustrated by the left hand diagram of Figure 3. However, with a larger trust region radius, it is possible to have $\Delta q > 0$, as in the right hand diagram. We describe the resulting steps respectively as being either *h-type* or *f-type*, according to whether $\Delta q \leq 0$ or not. Of course, if $h^{(k)} = 0$ the resulting step must be f-type. We construct our TR algorithm such that whilst f-type steps are being taken, we make no new entries into the filter, and rely to a large extent on the above convergence theory. Only h-type steps give rise to a filter entry (we include the use of feasibility restoration as an h-type step). The resulting algorithm is detailed in Figure 4. Note that the current pair $(h^{(k)}, f^{(k)})$ is not in the filter, and is only included subsequently if the algorithm takes an h-type step.

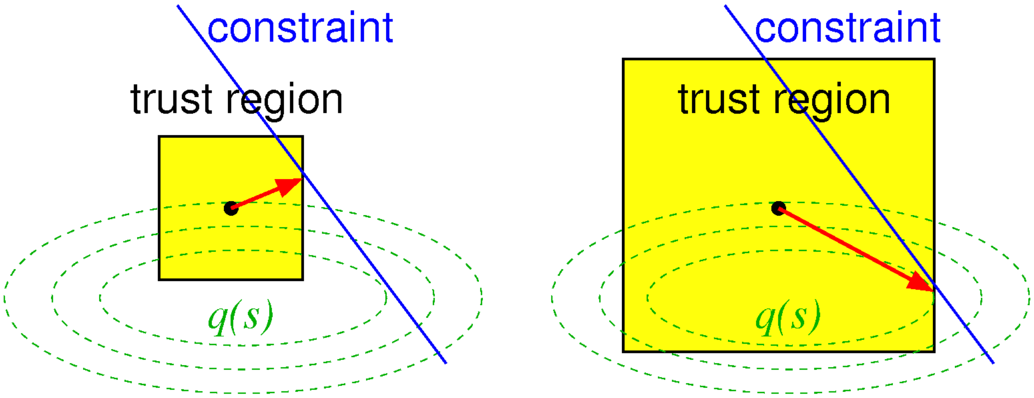

Figure 3: Illustration of h-type and f-type steps

We now turn to examine the way in which the slanting envelope (6.1) operates. If an infinite sequence of entries are made, then it is a consequence that $h^{(k)} \to 0$ (otherwise the condition $f \le f_i - \gamma h$ has the effect of forcing $f \to -\infty$: a contradiction). Thus the convergence proof claims that either

1. The restoration phase converges to a locally infeasible point,
2. The algorithm terminates at a KT point, or
3. There exists a feasible accumulation point that is either a KT point, or the Mangasarian-Fromowitz constraint qualification (MFCQ) fails.

The proof proceed as follows. The first case corresponds to the situation that the local approximation to the constraint set is infeasible, and no further progress can be made. So we only need to examine case 3. If there are an infinite number of h-type iterations, it is possible to find a subsequence on which $h^{(k)} \to 0$ by virtue of (6.1). By taking thinner subsequences if necessary, we examine the behaviour of iterates in the neighbourhood of a feasible non-KT point that satisfies MFCQ (a type of regularity condition). Because MFCQ holds, there exist feasible descent directions and it is shown that the TR algorithm takes f-type steps in the limit, which is a contradiction.

The only other possibility to consider is that there exists some K such that the algorithm takes f-type steps for all $k \ge K$. We can deduce from (6.1) and the fact that $(h^{(k+1)}, f^{(k+1)})$ is always acceptable to $(h^{(k)}, f^{(k)})$, that $h^{(k)} \to 0$. Then an argument similar to that in the unconstrained case contradicts the fact that there are feasible descent directions at

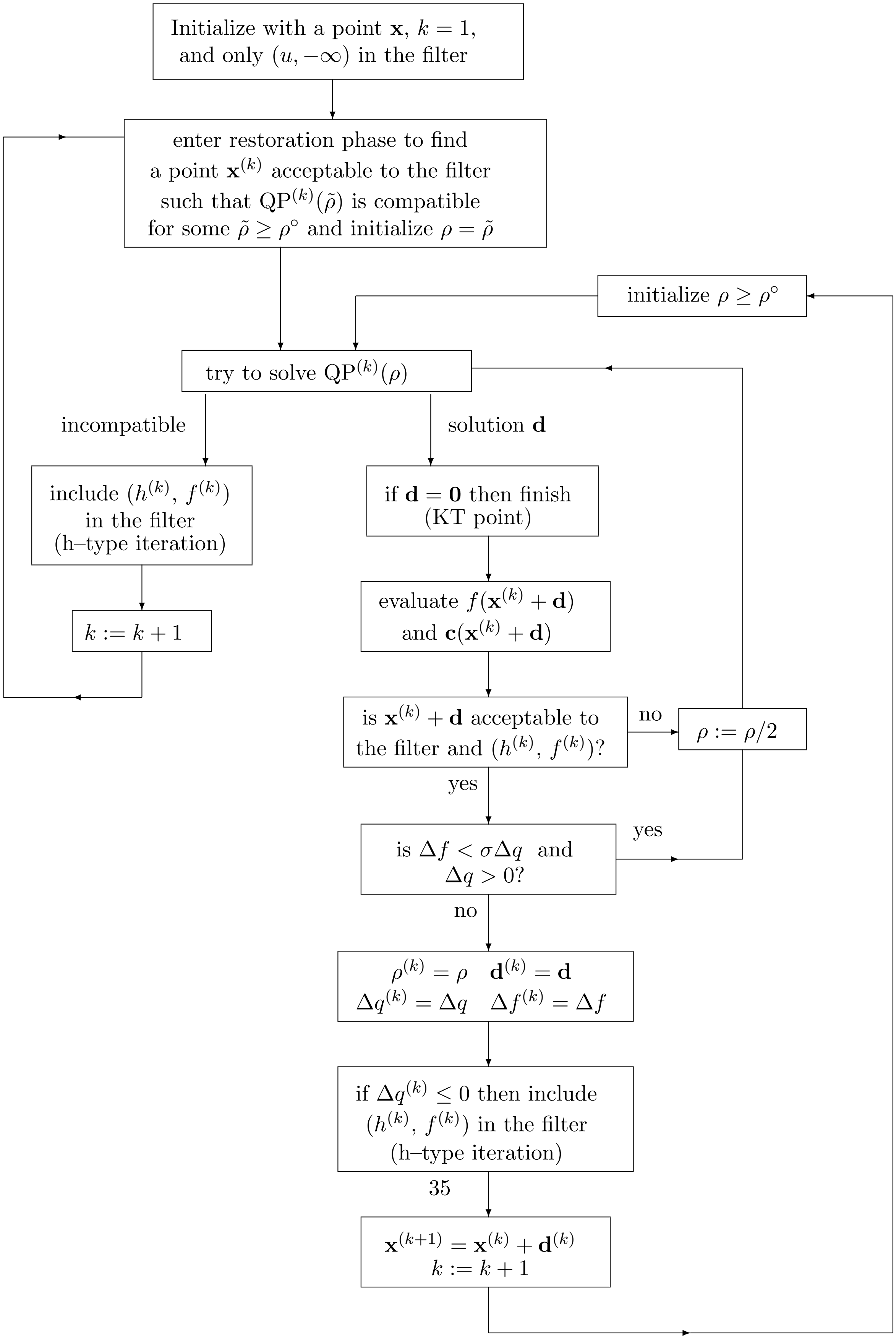
Initialize with a point $\mathbf{x}$, $k=1$, and only $(u,-\infty)$ in the filter
enter restoration phase to find a point $\mathbf{x}^{(k)}$ acceptable to the filter such that $\mathrm{QP}^{(k)}(\tilde{\rho})$ is compatible for some $\tilde{\rho} \geq \rho^\circ$ and initialize $\rho = \tilde{\rho}$
initialize $\rho \geq \rho^\circ$
try to solve $\mathrm{QP}^{(k)}(\rho)$
incompatible
solution $\mathbf{d}$
include $(h^{(k)},\, f^{(k)})$ in the filter (h–type iteration)
$k := k+1$
if $\mathbf{d}=\mathbf{0}$ then finish (KT point)
evaluate $f(\mathbf{x}^{(k)}+\mathbf{d})$ and $\mathbf{c}(\mathbf{x}^{(k)}+\mathbf{d})$
is $\mathbf{x}^{(k)}+\mathbf{d}$ acceptable to the filter and $(h^{(k)},\, f^{(k)})$?
no
$\rho := \rho/2$
yes
is $\Delta f < \sigma \Delta q$ and $\Delta q > 0$?
yes
no
$\rho^{(k)} = \rho \quad \mathbf{d}^{(k)} = \mathbf{d}$
$\Delta q^{(k)} = \Delta q \quad \Delta f^{(k)} = \Delta f$
if $\Delta q^{(k)} \leq 0$ then include $(h^{(k)},\, f^{(k)})$ in the filter (h–type iteration)
$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{d}^{(k)}$
$k := k+1$

Figure 4: A Filter–SQP Algorithm

any accumulation point. Because MFCQ holds, it follows that the accumulation point is a KT point. In passing, note that the Corollary to Lemma 1 in [20] contains an error. A corrected version of the paper can be found on my web site.

6.3 Other filter SQP methods

Another filter SQP algorithm , analysed by Fletcher, Gould, Leyffer, Toint and Wächter [16], decomposes the SQP step into a normal and tangential component. The normal step provides feasibility for the linearized constraints and the tangential step minimizes the quadratic model in the feasible region. Related ideas are discussed by Gonzaga, Karas and Varga [26] and Ribiero, Karas and Gonzaga [41]. Wächter and Biegler describe line search filter methods in [45] and [46]. Chin [5] and Chin and Fletcher [6] consider SLP-EQP trust region filter methods. Gould and Toint [28] present a non-monotone filter SQP which extends the non-monotonic properties of filter SQP type algorithms. A review of other recent developments of filter methods, outwith SQP, but including interior point methods for NLP, appears in the SIAG/OPT activity group newsletter (March 2007) and can be accessed in [21].

7 Modelling Languages and NEOS

Solving complex optimization problems that arise in practice has many difficulties. Writing your own NLP solver is not recommended, as there are many difficulties to be circumvented, even if you have access to a good QP solver. However, access to a fully developed production code, e.g. MINOS, LANCELOT, etc., is not the end of the story. The interface between the requirements of the code, and the features of the problem can be very difficult to set up. It is usually necessary to provide derivatives of any nonlinear functions, which is prone to error. Modelling languages have been designed to allow the user to present the problem to the NLP solver in as friendly a way as possible. The user is able to define constructions in terms of concepts familiar to the problem. Three languages come to mind, AMPL, GAMS and TOMLAB. All are hooked up to a number of well known NLP solvers. TOMLAB is based on MATLAB syntax, the other have their own individual syntax that has to be learned. Unfortunately all are commercial products.

In this review I shall describe AMPL (A Mathematical Programming Language) which I have found very flexible and easy to use. A student edition is freely available which allows problems of up to 300 variables and constraints, and gives access to MINOS and some other solvers. Access to AMPL for larger problems is freely available through the so-called NEOS system, described below in Section 7.5.

7.1 The AMPL Language

AMPL is a high level language for describing optimization problems, submitting them to a solver, and manipulating the results. An AMPL program has three main parts, the `model`, in which the problem constructs are defined, the `data` which is self evident, and `programming` in which instructions for activating a solver, and displaying or manipulating the results are carried out. A list of model, data and programming *commands* are prepared in one or more files and presented to the AMPL system. This processes the information, evaluates any derivatives automatically, and presents the problem to a designated solver. Results are then returned to the user. The AMPL system is due to Fourer, Gay and Kernighan, and is described in the AMPL reference manual [23]. The syntax of AMPL is concisely defined in the Appendix of [23], and provides an invaluable aid to debugging an AMPL model. The user is strongly advised to come to terms with the notation that is used. Other more abridged introductions to AMPL are available, as can be found by surfing the web.

The main features of an NLP problem are the *variables*, presented via the keyword `var`, the *objective*, presented via either `minimize` or `maximize`, and *constraints*, presented via the keyword `subject to`. These constructions are described using entities introduced by the keywords `param` for fixed parameters, and `set` for multivalued set constructions. An example which illustrates all the features is to solve the HS72 problem in CUTE. Here the model is specified in a file `hs72.mod`. Note that upper and lower case letters are different: here we have written user names in upper case, but that is not necessary. All AMPL commands are terminated by a semicolon.

`hs72.mod`

```
set ROWS = {1..2};
set COLUMNS = {1..4};
param A {ROWS, COLUMNS};
param B {ROWS};
var X {COLUMNS} >= 0.001;
minimize OBJFN: 1 + sum {j in COLUMNS} x[j];
subject to
   CONSTR {i in ROWS}:  sum {j in COLUMNS} A[i,j]/X[j] <= B[i];
   UBD {j in COLUMNS}:  X[j] <= (5-j)*1e5;
```

In this model, the sets are just simple ranges, like 1..4 (i.e. 1 up to 4). We could have shortened the program by deleting the set declaration and replacing `ROWS` by `1..2` etc., in the rest of the program. But the use of `ROWS` and `COLUMNS` is more descriptive. The program defines a vector of variables `X`, and the data is the matrix `A` and vector `B` which are parameters. Simple lower bounds on the variables are specified in the `var` statement, and `sum` is a construction which provides summation. The constraint `CONSTR` implements the system of inequalities $\sum_{j=1}^{4} a_{i,j}/x_j \leq b_i$, $i = 1, 2$. Note that indices are given within square brackets in AMPL. Constructions like `j in COLUMNS` are referred to as *indexing* in the AMPL syntax. The constraints in `UBD` define upper bounds on the variables which depend upon `j`. Note that the objective function and each set of constraint functions must be given a name by the user.

The data of the problem is specified in the file `hs72.dat`. Note the use of tabular presentation for elements of `A` and `B`.

hs72.dat

```
param A: 1 2 3 4 :=
    1  4     2.25  1     0.25
    2  0.16  0.36  0.64  0.64;
param B :=
    1  0.0401
    2  0.010085;
```

The next stage is to fire up the AMPL system on the computer. This will result in the user receiving the AMPL prompt `ampl:`. The programming session to solve HS72 would proceed as follows.

An AMPL session

```
ampl: model hs72.mod;
ampl: data hs72.dat;
ampl: let {j in COLUMNS} X[j] := 1;
ampl: solve;
ampl: display OBJFN;
ampl: display X;
```

The `model` and `data` keywords read in the model and data. The keyword `let` allows assignment of initial values to the variables, and the `display` commands initiate output to the terminal. Output from AMPL (not shown here) would be interspersed between the user commands. It is also possible to aggregate data and, if required, programming, into a single `hs72.mod` file. In this case the data must follow the model, and must be preceded by the statement `data;`. One feature to watch out for occurs when revising the model. In this case, repeating the command `model hs72.mod;` will *add* the new text to the database, rather than overwrite it. To remove the previous model from the database, the command `reset;` should first be given.

7.2 Networks in AMPL

AMPL is a most convenient system for modelling networks of diverse kinds (road, gas supply, work flow, ...). We illustrate some of the useful AMPL constructions in this section by reference to an *electrical power network* in which the objective is to minimize the power generation required to meet the demand on a given network. The model also illustrates some other useful AMPL features, notably the use of *dependent variables.* A toy system used by electrical engineers is illustrated in Figure 5. It is described in AMPL by

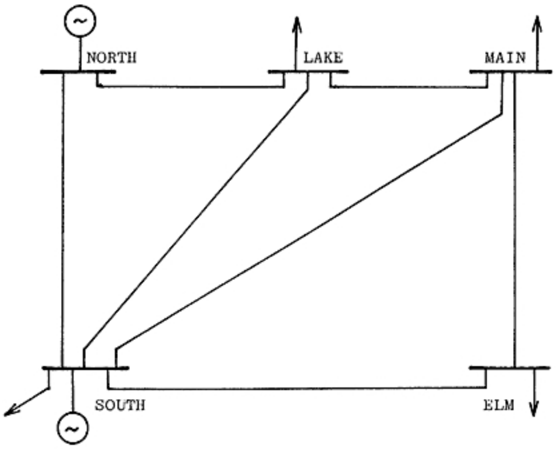

Figure 5: A Simple Power Distribution Network

the constructions shown below. Note the use of the keyword `union` to merge the nodes with and without power generation. Also observe the use of `cross`, which indicates all possible connections between the nodes, and `within` which indicates that the actual network is a subset of these. The user is using the convention that power flows *from* the first node *to* the second node (a negative value of flow is allowed and would indicate flow in the opposite sense). The program also shows the use of dependent parameters and variables. Thus the parameters `ZSQ` depend on `R` and `X`, and `C` and `D` both depend on `R`, `X` and `ZSQ`. It is necessary that the order in which these statements are given reflects these dependencies. The true variables in the problem (as shown here) are `V`, `THETA` and `PG`. Additional variables which depend on these variables, and also the parameters, are `PSI` and `P`, as defined by the expressions which follow the '=' sign. The objective is to minimize the sum of generated power. Constraints include power balance constraints at consumer nodes and generator nodes, the latter including a term for constraint generation. Note the use of `P[j,i]` for power entering node `i` and `P[i,j]` for power exiting node `i`. The program also provides a useful illustration of how to supply data for network problems, and the use of the # sign for including comments. Note also the use of standard functions such as `sin` and `cos` in the expressions. The program shown is only part of the full model, which would include flow of reactive power, and upper and lower bounds on various of the variables.

7.3 Other useful AMPL features

The following problem is due to Bob Vanderbei (who gives many interesting AMPL programs: search for vanderbei princeton on Google and click on LOQO). A rocket starts at p

A Power Generation Problem

```
set CONSUMERS;
set GENERATORS;
set NODES := CONSUMERS union GENERATORS;
set POWERLINES within (NODES cross NODES);
param LOADP {NODES};
param R {POWERLINES};
param X {POWERLINES};
param ZSQ {(i,j) in POWERLINES} = R[i,j]^2+X[i,j]^2;
param C {(i,j) in POWERLINES} = R[i,j]/ZSQ[i,j];
param D {(i,j) in POWERLINES} = X[i,j]/ZSQ[i,j];
...
var V {NODES};          # Voltages
var THETA {NODES};      # Phase angles
var PG {NODES};
...
var PSI {(i,j) in POWERLINES} = THETA[i] - THETA[j];
var P {(i,j) in POWERLINES} = C[i,j]*V[i]^2 +
           V[i]*V[j]*(D[i,j]*sin(PSI[i,j]) - C[i,j]*cos(PSI[i,j]);
...
minimize PGEN: sum {i in GENERATORS} PG[i];
...
subject to EQUALPC {i in CONSUMERS}: sum {(j,i) in POWERLINES} P[j,i]
                  = sum {(i,j) in POWERLINES} P[i,j] + LOADP[i];
EQUALPG {i in GENERATORS}: PG[i,j] + sum {(j,i) in POWERLINES} P[j,i]
         = sum {(i,j) in POWERLINES} P[i,j] + LOADP[i];
...
data;
set CONSUMERS := LAKE MAIN ELM;
set GENERATORS := NORTH SOUTH;
set POWERLINES := (NORTH,SOUTH) (NORTH,LAKE) (SOUTH,LAKE) (SOUTH,MAIN)
              (SOUTH,ELM) (LAKE,MAIN) (MAIN,ELM);
param:       R    X  :=
NORTH SOUTH 0.02 0.06
NORTH LAKE  0.08 0.24
SOUTH LAKE  0.06 0.24
...
MAIN  ELM   0.08 0.24;
param LOADP := NORTH 0; SOUTH 0.2, LAKE 0.45, MAIN 0.4, ELM 0.6;
...
```

time $t = 0$, position $x(0) = 0$ and velocity $v(0) = 0$. It must reach position $x(T) = 100$, also with velocity $v(T) = 0$. We shall divide the total time T into n intervals of length h and use finite difference approximations

$$v_i = \frac{x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}}{h} \qquad \text{and} \qquad a_i = \frac{v_{i+\frac{1}{2}} - v_{i-\frac{1}{2}}}{h}.$$

for velocity and acceleration. The maximum velocity is 5 units and the acceleration must lie within ±1 units. The aim is to minimize the total time $T = nh$. The AMPL program is

A Rocket Problem

```
param n > 2;
set vrange = {0.5..n-0.5 by 1}; set arange = {1..n-1};
var x {0..n}; var v {vrange} <= 5; var a {arange} <= 1, >= -1;
var h;
minimize T: n*h;
subject to
     xdiff {i in vrange}:  x[i+0.5] - x[i-0.5] = h*v[i];
     vdiff {i in arange}:  v[i+0.5] - v[i-0.5] = h*a[i];
     x0: x[0] = 0;   xn: x[n] = 100;
     v0: v[1.5] = 3*v[0.5];        # Implements v0 = 0
     vn: v[n-1.5] = 3*v[n-0.5];    # Implements vn = 0
```

The actual value of `n` must be supplied in the data section. An alternative implementation could be made in which `v` and `a` are expressed as dependent variables. Things to observe include the treatment of ranges and the use of `by`. Also note the use of both upper and lower bounds on the variables, and the beautifully concise form that AMPL permits.

AMPL can also be very descriptive when applied to finite element discretizations. I have used the following constructions in the context of a two dimensional p.d.e.

2-D Finite Element Constructions

```
set NODES;
set DIRICHLET within NODES;
set INTERIOR = NODES diff DIRICHLET;
set TRIANGLES within NODES cross NODES cross NODES;
set EDGES = setof {(i,j,k) in TRIANGLES} (i,j) union
             setof {(i,j,k) in TRIANGLES} (j,k) union
             setof {(i,j,k) in TRIANGLES} (k,i);
set SHAPE_FNS = setof {(i,j,k) in TRIANGLES} (i,j,k) union
             setof {(i,j,k) in TRIANGLES} (j,k,i) union
             setof {(i,j,k) in TRIANGLES} (k,i,j);
```

This illustrates the use of three suffix quantities (`TRIANGLES`), and the selection of all two suffix entities (`EDGES`) using `setof`. Shape functions are elementary functions defined on triangles taking the value 1 at one node and 0 at the others. Note also the use of `diff` for set difference, and the allowed use of underscore within an identifier.

AMPL contains many more useful constructions which we have not space to mention here. Purchasing a copy of the manual is essential! Worthy of mention however is the existence of an `if then else` construction. This can be very useful at the programming stage. It is also allowed within a model but should be used with care, because it usually creates a nonsmooth function which many methods are not designed to handle. The same goes for `abs` and related nonsmooth functions. Another useful feature for creating loops at the programming stage is the `repeat` construction.

7.4 Accessing AMPL

Unfortunately the full AMPL system is a commercial product for which a licence fee must be paid. However there is a student version of AMPL available which is free, but restricted to no more than 300 variables and constraints. In this section we list the steps needed to install the student version of AMPL on a unix operating system, as follows

1. connect to `www.ampl.com`
2. follows the link to **download** the student edition, following the quick start instructions
3. choose the architechture
4. download `ampl.gz` and gunzip it
5. download one or more solvers (e.g. MINOS or SNOPT).

Ideally these files should be stored in a `\usr\bin` area with symbolic links that enable them to be called from other directories.

7.5 NEOS and Kestrel

NEOS (Network Enabled Optimization Server) is a system running at the Argonne National Laboratory that solves optimization problems submitted by anyone, through the medium of the internet. The simplest way to access NEOS is via email. The following template, using the solver SNOPT for example, should be sent to `neos@mcs.anl.gov`, including the model, data, etc., where indicated.

NEOS Template for AMPL

```
<document>
<category>nco</category>
<solver>SNOPT</solver>
<inputMethod>AMPL</inputMethod>

<model><![CDATA[
Insert Model Here
]]></model>

<data><![CDATA[
Insert Data Here
]]></data>

<commands><![CDATA[
Insert Programming Here
]]></commands>

<comments><![CDATA[
Insert Any Comments Here
]]></comments>

</document>
```

An alternative approach is to use the **Kestrel** interface to NEOS. This enables the remote solution of optimization problems within the AMPL and GAMS modelling languages. Quoting from the documentation, problem generation, including the run-time detection of syntax errors, occurs on the local machine using any available modelling language facilities. Solution takes place on a remote machine, with the result returned in the native modelling language format for further processing. To use Kestrel, the Kestrel interface must be downloaded at step 5 above, using the same process as for downloading the solvers. To initiate a solve with say SNOPT using Kestrel, the following protocol must be initiated when using AMPL on the local machine.

Accessing NEOS from the Kestrel interface

```
ampl:  option solver kestrel;
ampl:  option kestrel_options "solver=SNOPT";
```

An advantage of using NEOS from Kestrel (or by email as above) is that the restriction in size no longer applies. A disadvantage is that the response of the NEOS server can be slow at certain times of the day.

References

[1] E. M. L. Beale, *Numerical Methods* In: Nonlinear Programming, J. Abadie, ed., North-Holland, Amsterdam, 1967.

[2] I. Bongartz, A. R. Conn, N. I. M. Gould and Ph. L. Toint, *CUTE: Constrained and Unconstrained Testing Environment* ACM Trans. Math. Software, 21, 1995, pp. 123-160.

[3] R. H. Byrd, N. I. M. Gould, J. Nocedal and R. A. Waltz, *An Active-Set Algorithm for Nonlinear Programming Using Linear Programming and Equality Constrained Subproblems*, Math. Programming B, 100, 2004 pp. 27-48.

[4] R. H. Byrd, J. Nocedal and R. B. Schnabel, *Representations of quasi-Newton matrices and their use in limited memory methods*, Math. Programming, 63, 1994, pp. 129-156.

[5] C. M. Chin, *A new trust region based SLP filter algorithm which uses EQP active set strategy*, PhD thesis, Dept. of Mathematics, Univ. of Dundee, 2001.

[6] C. M. Chin and R. Fletcher, *On the global convergence of an SLP-filter algorithm that takes EQP steps*, Math. Programming, 96, 2003, pp. 161-177.

[7] R. M. Chamberlain, C. Lemarechal, H. C. Pedersen and M. J. D. Powell, *The watchdog technique for forcing convergence in algorithms forconstrained optimization*, In: Algorithms for Constrained Minimization of Smooth Nonlinear Functions, A. G. Buckley and J.-L. Goffin, eds., Math. Programming Studies, 16, 1982, pp. 1-17.

[8] A. R. Conn, N. I. M. Gould and Ph. L. Toint, Trust Region Methods, MPS-SIAM Series on Optimization, SIAM Publications, Philadelphia, 2000.

[9] R. Courant, *Variational methods for the solution of problems of equilibrium and vibration,* Bull. Amer. Math. Soc., 49, 1943, pp. 1-23.

[10] J. E. Dennis and J. J. Moré, *A characterization of superlinear convergence and its application to quasi-Newton methods*, Math. Comp., 28, 1974, pp. 549-560.

[11] R. Fletcher, *A model algorithm for composite nondifferentiable optimization problems*, In: Nondifferential and Variational Techniques in Optimization, D. C. Sorensen and R. J.-B. Wets eds., Math.Programming Studies, 17, 1982, pp. 67-76.

[12] R. Fletcher, *Second order corrections for nondifferentiable optimization*, In: Numerical Analysis, Dundee 1981, G. A. Watson ed., Lecture Notes in Mathematics 912, Springer-Verlag, Berlin, 1982.

[13] R. Fletcher, *Practical Methods of Optimization*, 1987, Wiley, Chichester.

[14] R. Fletcher, *Dense Factors of Sparse Matrices*, In: Approximation Theory and Optimization, M. D. Buhmann and A. Iserles, eds, C.U.P., Cambridge, 1997.

[15] R. Fletcher, *A New Low Rank Quasi-Newton Update Scheme for Nonlinear Programming*, In: System Modelling and Optimization, H. Futura, K. Marti and L. Pandolfi, eds., Springer IFIP series in Computer Science, 199, 2006, pp. 275-293, Springer, Boston.

[16] R. Fletcher, N. I. M. Gould, S. Leyffer, Ph. L. Toint, and A. Wächter, *Global convergence of trust-region SQP-filter algorithms for general nonlinear programming*, SIAM J. Optimization, 13, 2002, pp. 635-659.

[17] R. Fletcher, A. Grothey and S. Leyffer, *Computing sparse Hessian and Jacobian approximations with optimal hereditary properties*, In: Large-Scale Optimization with Applications, Part II: Optimal Design and Control, L. T. Biegler, T. F. Coleman, A. R. Conn and F. N. Santosa, Springer, 1997.

[18] R. Fletcher and S. Leyffer, *Nonlinear programming without a penalty function*, Math. Programming, 91, 2002, pp. 239–270.

[19] R. Fletcher and S. Leyffer, *Filter-type algorithms for solving systems of algebraic equations and inequalities*, In: G. di Pillo and A. Murli, eds, High Performance Algorithms and Software for Nonlinear Optimization, Kluwer, 2003.

[20] R. Fletcher, S. Leyffer, and Ph. L. Toint, *On the global convergence of a filter-SQP algorithm*, SIAM J. Optimization, 13, 2002, pp. 44-59.

[21] R. Fletcher, S. Leyffer, and Ph. L. Toint, *A Brief History of Filter Methods*, Preprint ANL/MCS-P1372-0906, Argonne National Laboratory, Mathematics and Computer Science Division, September 2006.

[22] R. Fletcher and E. Sainz de la Maza, *Nonlinear programming and nonsmooth optimization by successive linear programming*, Math. Programming, 43, 1989, pp. 235-256.

[23] R. Fourer, D. M. Gay and B. W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming*, 2nd Edn., Duxbury Press, 2002.

[24] K. R. Frisch, *The logarithmic potential method of convex programming*, Oslo Univ. Inst. of Economics Memorandum, May 1955.

[25] P. E. Gill, W. Murray and M. A. Saunders, *SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization*, SIAM Review, 47, 2005, pp. 99-131.

[26] C. C. Gonzaga, E. Karas, and M. Vanti, *A globally convergent filter method for nonlinear programming*, SIAM J. Optimization, 14, 2003, pp. 646-669.

[27] L. Grippo, F. Lampariello and S. Lucidi, *A nonmonotone line search technique for Newton's method*, SIAM J. Num. Anal., 23, pp. 707-716.

[28] N. I. M. Gould and Ph. L. Toint, *Global Convergence of a Non-monotone Trust-Region SQP-Filter Algorithm for Nonlinear Programming*, In: Multiscale Optimization Methods and Applications, W. W. Hager, S. J. Huang, P. M. Pardalos and O. A. Prokopyev, eds., Springer Series on Nonconvex Optimization and Its Applications, Vol. 82, Springer Verlag, 2006.

[29] S. P. Han, *A globally convergent method for nonlinear programming*, J. Opt. Theo. Applns., 22, 1976, pp.297-309.

[30] M. R. Hestenes, *Multiplier and gradient methods*, J. Opt. Theo. Applns, 4, 1969, pp. 303-320.

[31] W. Karush, *Minima of functions of several variables with ineqalities as side conditions*, Master's Thesis, Dept. of Mathematics, Univ. of Chicago, 1939.

[32] H. W. Kuhn and A. W. Tucker, *Nonlinear Programming*, In: Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, J. Neyman, ed., University of California Press, 1951.

[33] N. Maratos, *Exact penalty function algorithms for finite dimensional and control optimization problems*, Ph.D. Thesis, Univ. of London, 1978.

[34] B. A. Murtagh and M. A. Saunders, *A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints*, Math. Programming Studies, 16, 1982, pp. 84-117.

[35] J. Nocedal, *Updating quasi-Newton matrices with limited storage*, Math. Comp., 35, 1980, pp. 773-782.

[36] J. Nocedal and M. L. Overton, *Projected Hessian updating algorithms for nonlinearly constrained optimization*, SIAM J. Num. Anal., 22, 1985, pp. 821-850.

[37] T. Pietrzykowski, *An exact potential method for constrained maxima*, SIAM J. Num. Anal., 6, 1969, pp. 217-238.

[38] R. Polyak, *Modified barrier functions (theory and methods)*, Math. Programming, 54, 1992, pp. 177-222.

[39] M. J. D. Powell, *A method for nonlinear constraints in minimization problems*, In: Optimization, R. Fletcher ed., Academic Press, London, 1969.

[40] M. J. D. Powell, *A fast algorithm for nonlinearly constrained optimization calculations*, In: Numerical Analysis, Dundee 1977, G. A. Watson, ed., Lecture Notes in Mathematics 630, Springer Verlag, Berlin, 1978.

[41] A. A. Ribeiro, E. W. Karas, and C. C. Gonzaga, *Global convergence of filter methods for nonlinear programming*, Technical report, Dept .of Mathematics, Federal University of Paraná, Brazil, 2006.

[42] S. M. Robinson, *A quadratically convergent method for general nonlinear programming problems*, Math. Programming, 3, 1972, pp. 145-156.

[43] R. T. Rockafellar, *Augmented Lagrange multiplier functions and duality in non-convex programming*, SIAM J. Control, 12, 1974, pp. 268-285.

[44] Ph. L. Toint, *On sparse and symmetric updating subject to a linear equation*, Math. Comp., 31, 1977, pp. 954-961.

[45] A. Wächter and L. T. Biegler, *Line search filter methods for nonlinear programming: Motivation and global convergence*, SIAM J. Optimization, 16, 2005, pp. 1-31.

[46] A. Wächter and L. T. Biegler, *Line search filter methods for nonlinear programming: Local convergence*, SIAM J. Optimization, 16, 2005, pp. 32-48.

[47] R. B. Wilson, *A simplicial algorithm for concave programming*, Ph.D. dissertation, Harvard Univ. Graduate School of Business Administration, 1960.