CURRENT TECHNIQUES FOR NON-LINEAR PROGRAMMING

SUSAN J. BYRNE
MASSEY UNIVERSITY, PALMERSTON NORTH, NEW ZEALAND

IAN D. COOPE
UNIVERSITY OF CANTERBURY, CHRISTCHURCH, NEW ZEALAND

SUMMARY

Non-linear programming has become increasingly important as the inadequacies of linear models have become more apparent. We commence this paper with a look at a wide range of problems solved using non-linear programming. The general form of such problems is outlined.

Several approaches currently used for solving these problems are briefly described with the characteristics of each being simply explained. Methods developed specifically for large scale problems are considered.

We conclude by discussing the computer codes available for solving non-linear programming problems and their suitability for particular types of problems.

1. INTRODUCTION

The study and development of non-linear optimization methods have become increasingly important in recent years as the inadequacies of approximate linear programming models for planning have become more obvious. We commence with a brief look at the range of problems tackled using non-linear optimization. The general form of such problems is outlined, along with some necessary conditions for a solution.

Several approaches to solving these problems in practice are developed, with the characteristics of each approach being clearly indicated. The methods considered include penalty and barrier function methods, augmented Lagrangian functions, reduced gradient methods, and sequential quadratic programming. Methods specifically developed for large-scale, usually sparse, problems are also discussed. Computer codes have been written to implement these methods, so we conclude by indicating the availability of such codes and the problem types for which each is more suited.

Manuscript submitted June 1982, revised July 1982
2. APPLICATIONS

A study of recent abstracts reported in International Abstracts in Operations Research has uncovered a wide range of problems which have been analysed using the various techniques of non-linear optimization, ranging from health, engine design, economic models, optical filter design, to agriculture and education planning models. Some of these areas are now described in more detail.

Within the general social area, most work is concentrated on health problems. Boldy (1976) has a comprehensive review of the problems tackled, including health admission systems, menu planning, hospital location, as well as the strategic problems of planning the prevention and control of disease. Kolesar (1980) explores further a model to select testing intervals for patients with glaucoma. Rousseau and Laporte (1977) devised a practical optimization technique for scheduling outpatient appointments at a clinic, so as to minimize total patient queueing time. Another problem considered (Huckfeldt, 1976) was planning post-secondary education provision in several states in America.

Scheduling problems tackled include work on determining the best repair times in a machine shop. Time spent in repair and maintenance is to be minimized, but breakdowns from lack of care are to be avoided. Lie (1977) and Goheen (1977) have both solved this problem, assuming particular distributions for expected failure rates and corrective and preventive maintenance times. The familiar problem of bus scheduling and fleet selection is also considered, as in Sengupta and Gupta (1980).

Interest in optimal transportation design problems varies from engines to road networks. Zarotti and Nervegna (1979) in their review discuss selected applications of non-linear programming to such areas of engine design as torque converters and hydrodynamic conversion. Road networks and traffic assignment problems are considered by Robinson (1977), LeBlanc et al (1979) and Ho (1980). Lasdon and Waren (1980) survey further applications. Robinson treated problems with up to 90 variables and 400 inequality constraints, while LeBlanc had 1860 variables and 550 constraints in his largest problem.

Water supply and distribution problems have recently been considered. Himmelblau (1977) showed that a water resource system could be managed in an optimal way to fulfil both quality and quantity demands at least cost. Careful modelling of the problem and thoughtful formulation was required to obtain useful solutions. Kao and Meyer (1981) discuss the solution of a large water supply problem with more than 900 variables and 600 constraints.

Non-linear optimization techniques have been used in the oil industry. Lasdon and Waren (1980) discuss numerous problems considered, including modelling of oil field development and selection of optimal investment policies (Ali et al, 1978), the optimization of refinery operation to ensure maximum profits (Griffith and Stewart, 1961), and the selection of system
parameters for heat exchangers, furnaces, and coolers for a given process and temperature configuration (Bryant and Gutterman, 1979). Optimization techniques are used extensively in systems for computer-aided design of chemical engineering plant, such as that outlined by Sargent (1967). Work on particular elements of plant design has included the selection of plate sizes for distillation columns to obtain the desired separation at minimum cost (Sargent and Gaminibandara, 1976).

Assistance in budget allocation and investment decisions has been provided by non-linear programming. Gay (1978) describes the use of these techniques in helping to reduce production losses in a pottery and make decisions regarding the acquisition of new plant. Investment decisions in banking (Mavrides, 1978), selection of ship fleet size (Murotsu and Taguchi, 1976), and plant expansion (Philip and Liittschwager, 1979), are other areas in which non-linear programming has assisted decision-makers. Many equilibrium economic models are formulated as non-linear problems and one such is described by Abadie and Robert (1978).

Most realistic models of energy production and demand involve significant non-linearities. Manne (1976) describes a large system built to examine the energy options available to the United States as it moves from a dependence on oil and gas to more diversified energy sources. It was essential to examine the effects of different technologies on the total system, and the non-linear programming model makes this task straightforward. Biggs and Laughton (1977) discuss the optimal scheduling of an electric power system to meet specified demands at minimum cost. Bosser (1978) considers the problem of introducing time-of-day marginal cost pricing for electricity tariffs, and examines the design and application of such tariffs. Laugham and Frail (1979) have examined the optimal design of wind power systems and showed that design parameters can be selected by the solution of a non-linear programming problem, given the wind regime at the site under consideration.

Finally, other applications have included decisions on deployment of customs officers (Lenormand and Walliser, 1979), aero engine design studies (Brown, 1976) and minimal cost design of optical filters (McKeown and Nag, 1976).

Problem sizes ranged from the small - perhaps 10 variables and 5 constraints - to the very large - 900 variables and 600 constraints. Larger problems are solvable, but usually require a detailed consideration of the structure of the constraints to simplify processing. The traffic assignment model of LeBlanc (1979) is an example of this type.

3. PROBLEM FORMULATION

Mathematically, the most general form of the non-linear problem is
minimize \( f(x) \)
\[
\text{subject to } c_j(x) = 0, \ j \in E
\]
\[
c_j(x) > 0, \ j \in I
\]

where the objective function \( f(x) \) and the constraint functions \( c_j(x) \) are general non-linear functions of the \( n \) real variables, \( x_j \), and \( E \) and \( I \) are index sets (possibly empty) of integers. It is normally assumed that all the functions involved are at least continuous with continuous first derivatives, and preferably continuous second derivatives. Methods of handling problems with discontinuous functions are being developed, but no approach has yet been found entirely acceptable. The required solution of the problem is the vector \( x^* \).

Solution methods are often designed to handle particular subclasses of the general problem (3.1). A common subdivision is to assume that either the set \( I \) or the set \( E \) is empty, so that the problem is subject to either equality or inequality constraints only. Of course, inequality constraints can be converted into equality constraints by the addition of suitable slack variables, for example, \( c_j(x) > 0 \) can be replaced by \( c_j(x) - y_j = 0 \). Another common distinction is between linear and non-linear constraints. For linear constraints the function \( c_j(x) \) is given by \( a_j^T x - b_j \), as in linear programming. Such constraints are considerably easier to handle, at least in part because their gradient vector is constant for any \( x \), unlike non-linear constraints. If simple bounds on the variable values are required, some algorithms require them to be explicitly stated as inequality constraints and treated as such. This contrasts with linear programming where non-negativity of the variables is generally assumed. Many of the newer algorithms do however process simple lower and upper bounds on variables directly, and such bounds should be tightly stated if they are known, particularly if it is not clear that only one possible minimum exists for the problem being investigated. Finally, a not uncommon subclass of non-linear programming is quadratic programming where the objective function is quadratic and all the constraints are linear. A general survey of solution methods for this class of problems is found in Byrne (1980).

4. SOME SIMPLE BEGINNINGS

Optimization of a function in the absence of any constraints is a much simpler problem than the non-linear programming problem (3.1). Even so, it is only recently that good methods for solving the unconstrained optimization problem have become well-established, with sufficient theoretical and practical evidence to ensure they work satisfactorily. Brodlie (1977) provides a good survey of this area, with discussion of the development of the various methods. It has been tempting, therefore, to attempt to convert constrained problems, particularly those with just bound constraints of the form \( l_i \leq x_i \leq u_i \) on the variables, into unconstrained problems, and use the good unconstrained optimization
procedures to solve the resultant transformed problem. Some familiar transformations are now discussed, as also are the difficulties and pitfalls inherent in this approach.

In the 1960's several transformations were suggested to convert a constrained problem into an unconstrained one. Box (1966) presents a good survey of this area. Some possibilities include:

(i) \( x_i \geq 0 \): \( x_i \) replaced by \( x_i = w_i^2 \)
(ii) \( 1_i \leq x_i \leq u_i \): \( x_i \) replaced by \( x_i = 1_i + (u_i - 1_i)\sin^2 y_i \)
(iii) \( 0 < x_i < 1 \): \( x_i \) replaced by \( x_i = e^{y_i}/(e^{y_i} + e^{-y_i}) \)
(iv) \( \sum x_i^2 = 1 \): \( x_i \) replaced by
\[
\begin{align*}
x_1 &= \sin y_1 \sin y_2 \ldots \sin y_{n-1} \\
x_i &= \cos y_{i-1} \sin y_i \sin y_{i+1} \ldots \sin y_{n-1}, \quad i = 2, \ldots, n-1 \\
x_n &= \cos y_{n-1}
\end{align*}
\]
(v) \( \sum x_i^2 = r^2 \): \( x_i \) replaced by
\[
\begin{align*}
x_i &= v_i/a, \quad i = 1, \ldots, n-1, \quad x_n = 1/a
\end{align*}
\]
\[
\text{where } a = \frac{1}{r} \sqrt{1 + \sum_{i=1}^{n-1} v_i^2}
\]
(vi) \( 0 \leq x_1 \leq x_2 \leq x_3 \): replaced by \( x_1 = y_1^2, \quad x_2 = y_2^2, \quad x_3 = y_1^2 + y_2^2 + y_3^2 \)

The characteristics of a desirable transformation for the case of bounded variables have been recently studied by Sisser (1981). These include that the transformation should lie in the desired region for all values of the introduced variable, and that as the introduced variable tends to \( \pm \infty \), the transformation should tend to the bounds of the desired region. These conditions are necessary to ensure that the transformation neither destroys existing local optima nor introduces any extraneous and possibly unbounded optima. However, even with these criteria, it is difficult to find acceptable transformations. Often the transformation creates a more difficult function to be optimized, for example, it may be highly non-linear, whereas the original function was only mildly so, or its derivatives may be much harder to evaluate. In addition, the new function may now be infinite for certain values of the variables or suffer other singularities. If trigonometric transformations are used periodicity may cause problems unless the function is carefully coded. Often extra stationary points are introduced which can be disastrous with some unconstrained optimization methods. Another serious problem is that often the Hessian matrix (the matrix of second partial derivatives of the function) becomes singular either at isolated
points, possibly including the desired optimum, or over some major and important part of the region. Even if the Hessian remains non-singular, it may become very poorly conditioned, which tends to cause difficulties in the optimization.

Although some of these transformations were useful in the absence of better methods, that is now no longer the case as good codes for non-linear optimization problems have become available. The simplest of these are for the case when the only constraints are simple upper and lower bounds on the variables. Suitable Fortran codes are available, for example, in the NAG Library (routines E04JAF, E04KAF). Even for the solution of unconstrained problems it is often more efficient to apply some bounds, derived from the physical problem, on the variable values and use one of these codes, rather than an unconstrained routine. In fact, the codes now available in the NPL library for unconstrained optimization do exactly that.

5. PENALTY AND BARRIER METHODS

Another approach which requires only unconstrained optimization software is to augment the objective function by terms which severely penalize constraint violations. The simplest penalty function has the form

$$P(x,a) = f(x) + a \sum_{j \in E} \{c_j(x)\}^2 + \sum_{j \in I} \{\min(0, c_j(x))\}^2$$

(5.1)

Thus violated constraints increase the value of the objective function by an amount that is proportional to the penalty parameter $a$. Initially a suitable value of $a$ is chosen and the unconstrained minimizer of $P(x,a)$ is determined. If the constraint violations are not acceptable $a$ is increased, typically by a factor of 10, and the process is repeated. In the limit as $a \to \infty$ the solution of the constrained problem is obtained. The simplicity of the approach is appealing but high accuracy is difficult to obtain since numerical problems arise through ill-conditioning when $a$ is large. Because the unconstrained minimizers provide infeasible but "better than optimal" solutions the method is clearly most useful when the constraints can be loosely interpreted.

If feasibility is essential and the constraints are all inequalities then the barrier function approach is more suitable. In this method an initial feasible point is required and this may be a non-trivial task for problems of moderate size. As with the penalty method a sequence of unconstrained minimizations is performed for increasing values of the parameter $a$. The simplest barrier function is

$$B(x,a) = f(x) + \frac{1}{a} \sum_{j=1}^{m} \frac{1}{c_j(x)}$$

(5.2)

The effect here is rather different from that of the penalty function. The addition of the barrier term causes the unconstrained minimizers to remain feasible but sub-optimal. As $a$ is increased convergence to the solution of the constrained problem is again assured in theory.
In practice the success of both penalty and barrier function methods is severely limited; convergence is usually slow and accuracy rather poor. Performance can be improved by extrapolation techniques (Fiacco and McCormick, 1968) but a better approach is discussed in Section 7. The major advantage is that they are easily programmed, especially in the case of penalty functions. A good discussion on the relative strengths and weaknesses of penalty and barrier methods is given by Ryan (1974), and a simple penalty method is available in the NAG library (routine E04HAF).

6. LAGRANGE MULTIPLIERS AND KUHN-TUCKER CONDITIONS

Except in pathological cases (which rarely occur in practical problems) necessary conditions for a constrained stationary point of problem (3.1) can be expressed in terms of a vector of Lagrange multipliers, $\lambda$. Introducing the Lagrangian function

$$L(x,\lambda) = f(x) - \sum_j c_j(x)$$

where the summation is taken over all constraints (IUE), then some of the most successful methods for solving problem (3.1) proceed by attempting to determine values of $x$, $\lambda$ satisfying the following Kuhn-Tucker conditions:

$$\nabla f(x) = \sum_j \lambda_j \nabla c_j(x)$$

$$c_j(x) = 0, \quad j \in E$$

$$c_j(x) \geq 0, \quad \lambda_j c_j(x) = 0, \quad \lambda_j \geq 0, \quad j \in I$$

(Note, $\nabla$ denotes the vector of partial derivatives with respect to $x$). When only equality constraints are present these conditions reduce to finding a solution $(x^*,\lambda^*)$ of the non-linear system of equations in $(x,\lambda)$:

$$\nabla L(x,\lambda) = 0$$

$$c(x) = 0,$$

which is the method of Lagrange multipliers. If a good initial approximation, $(x(0),\lambda(0))$, is available then Newton's method may be applied to solve this non-linear system and rapid convergence can be expected. Unfortunately this is rarely the case. A further disadvantage is that equations (6.3) also hold at constrained maxima and saddle points and no bias is given towards minimization. Therefore the above approach is not suitable in general. However, in Section 9 it is shown how Newton's method can be extended to handle inequality constraints. A further modification removes the necessity for good starting approximations and leads to some very effective algorithms for solving constrained optimization problems.

The next section combines the use of Lagrange multipliers with the simple penalty function (5.1).
7. AUGMENTED LAGRANGIAN METHODS

The disadvantage of the simple penalty function method can be removed if, instead of penalising the objective function, the Lagrangian function (6.1) is penalised. In the case when only equality constraints are present this leads to the augmented Lagrangian function

$$\psi(x, \lambda, \sigma) = f(x) - \sum_{j \in \mathcal{E}} \lambda_j c_j(x) + \sigma \sum_{j \in \mathcal{E}} [c_j(x)]^2$$  \hspace{1cm} (7.1)

Because $x^*$ is a stationary point of the Lagrangian function, $L(x, \lambda^*)$, it is also a stationary point of $\psi(x, \lambda^*, \sigma)$ for any value of $\sigma$, since $c(x^*) = 0$. The advantage of including the penalty term is that by choosing $\sigma (> 0)$ sufficiently large $x^*$ becomes a minimizer of $\psi(x, \lambda^*, \sigma)$ whereas usually $x^*$ is a saddle point for the Lagrangian $L(x, \lambda^*)$. Therefore, if values of the optimal Lagrangian multipliers are known, it is sufficient to perform a single unconstrained minimization of $\psi(x, \lambda^*, \sigma)$ with a suitably large value of $\sigma$ in order to determine $x^*$. Of course, in practice, $\lambda^*$ is not known and again a sequence of unconstrained minimizations must be performed. However, the aim now is to keep the penalty parameter, $\sigma$, constant to avoid the tendency towards increasingly ill-conditioned sub-problems. Instead a sequence of estimates of $\lambda^*$, $\{\lambda^{(k)}\}$, is used. If $x^{(k)}$ denotes the unconstrained minimizer of $\psi(x, \lambda^{(k)}, \sigma)$, then provided $\lambda^{(k)} \rightarrow \lambda^*$, it can be shown under reasonably mild assumptions that $x^{(k)} \rightarrow x^*$.

Initially $\lambda^{(0)}$ is usually taken as the zero vector so that the first iteration is the same as that for a simple penalty function. After each unconstrained minimization the estimates of $\lambda^*$ are updated. A simple scheme suggested by Hestenes (1969) which gives rise to a linear rate of convergence is

$$\lambda^{(k+1)} = \lambda^{(k)} - \sigma c(x^{(k)})$$  \hspace{1cm} (7.2)

The simplicity of this scheme is very attractive since it requires no derivative information nor any further demands on computer storage. The linear rate of convergence can be increased by increasing $\sigma$ and a suitable strategy is to multiply $\sigma$ by a factor of 10 if $\|c^{(k)}\| > \frac{1}{3} \|c^{(k-1)}\|$, (Powell, 1969). This ultimately ensures linear convergence at a rate of 0.25 or better. Alternatively, more sophisticated update schemes than (7.2) may be used which make use of derivative information and enable superlinear rates of convergence to be obtained. Further information, including the generalization to handle inequality constraints, is given in Fletcher (1981). FORTRAN codes are available, for example, in the NAG library (routines E04UAF, E04VAF, E04VBF, E04WAF) and in the Harwell library (routine VF01A).

Augmented Lagrangian methods, when carefully implemented, are extremely robust and have enjoyed considerable success over the last ten years. They do not require the user to provide initial feasible points and can be implemented either with or without making use of derivatives. Unfortunately they suffer two primary
disadvantages. One is that they do not handle linear constraints in an efficient manner. Therefore when only a small proportion of constraints is non-linear it is unlikely that they will provide the best approach. The second disadvantage is that several unconstrained minimizations have to be performed and this may prove expensive. Because of this latter deficiency some recent work has concentrated on utilizing low accuracy in the unconstrained minimizations in order to update the λ-parameters more frequently. Approaches of this kind are reviewed by Bertsekas (1976) and Coope and Fletcher (1980).

3. REDUCED GRADIENT AND GRADIENT PROJECTION METHODS

In contrast to the methods previously considered, methods in this category handle linear constraints very efficiently. Indeed reduced gradient and gradient projection methods are based on extending methods for linear constraints to the non-linear case. The underlying philosophy of methods of this type is to satisfy exactly a subset of the constraints so that the search for a minimum is confined to a subspace. The subset of constraints, called an active set, includes all equality constraints but only some of the inequality constraints, ideally those which are binding at the solution. Because the correct active set is not known initially it must be revised periodically. Whenever a constraint that is not in the active set is violated it is automatically added. However, it is also necessary to be able to delete constraints that have been incorrectly added. Decisions of this kind are based on estimates of Lagrange multipliers. The basic approach is best illustrated by considering the minimization of a non-linear function subject only to linear inequality constraints:

\[
\text{Min } f(x) \text{ subject to } a_j^T x - b_j \geq 0, \quad j \in I.
\]

If \( x^{(k)} \) is a feasible solution then the active set \( I_A^{(k)} \) is defined by

\[
I_A^{(k)} = \{ j : a_j^T x^{(k)} - b_j = 0 \}
\]

(8.1)

In order to find a better approximation to \( x^* \) a feasible descent direction \( p \) is calculated; that is, \( p \) satisfies

\[
p^T a_j = 0, \quad j \in I_A^{(k)} \quad \text{and} \quad p^T f(x^{(k)}) < 0
\]

(8.2)

These conditions imply that for sufficiently small \( \alpha (> 0) \), \( x^{(k)} + \alpha p \) remains feasible and gives a reduction in the value of the objective function. In practice \( \alpha \) is determined by a line search to ensure that \( x^{(k+1)} = x^{(k)} + \alpha p \) is a good approximation to the best feasible point in the descent direction \( p \). Then the active set at \( x^{(k+1)} \) is redefined if necessary before the next iteration is begun.

Of course the conditions (8.2) rarely define \( p \) uniquely. One choice is to let \( p \) be the vector

\[
p = -(I - A^T A)^{-1} A^T f(x^{(k)})
\]

(8.3)
where $A$ is the matrix composed from the vectors $a_j$, $j \in I^{(k)}$. This gives the analogue of the steepest descent algorithm for unconstrained minimization since $p$ is the projection of the steepest descent vector, $-\nabla f$, onto the linear manifold defined by the active constraints. Alternative (and better) choices for $p$ are obtained from the projected forms of conjugate gradient or quasi-Newton descent directions.

A basic difficulty with reduced gradient type methods is the requirement to maintain feasibility. This is straightforward when all constraints are linear but in the case of non-linear constraints an inner iteration is required to project infeasible estimates back into the feasible region. Thus when the constraints are highly curved the requirements of reducing the objective function and maintaining feasibility result in a sequence of very small steps. In such situations reduced gradient type methods can be extremely inefficient. When the constraints are linear or only mildly non-linear then this inefficiency is not present and good accuracy can usually be achieved in very few iterations.

The generalized reduced gradient (GRG) method of Abadie and Carpentier (1969) for non-linear constraints is the earliest of methods of this kind. Now there are several carefully implemented GRG codes available (Abadie, 1978; Lasdon and Waren, 1978; Gabriele and Ragsdell, 1977; Helte and Liittschwager, 1975) for solving small to medium sized problems. Gabriele (1980) and Lasdon et al (1978) have implemented versions for large sparse systems. A good discussion of the available algorithms and their advantages and disadvantages is given by Lasdon (1982). He comments that only sequential quadratic programming algorithms have been competitive or superior in comparative tests.

9. SEQUENTIAL QUADRATIC PROGRAMMING

Methods in this category solve quadratic programming subproblems at each iteration and are motivated by an extension of Newton's method, for solving the non-linear equations (9.1), to incorporate inequality constraints. Specifically, if $(x^{(k)}, \lambda^{(k)})$ approximate $(x^*, \lambda^*)$ then the next iterates are defined by first solving the subproblem

$$
\begin{array}{l}
\text{Min } \frac{1}{2} p^T B^{(k)} p + p^T \nabla f^{(k)} \\
\text{subject to } p^T v_j = 0, \quad \left\{ \begin{array}{c}
\sum_j c_j = 0, \\
 j \in I \\
\lambda_j \geq 0, \\
 j \in I
\end{array} \right.
\end{array}
\tag{9.1}
$$

This is a quadratic programming problem in $p$. If $B^{(k)}$ is the Hessian matrix of the Lagrangian function $L$, so that

$$
B^{(k)} = \nabla^2 f - \sum_j \lambda_j \nabla^2 c_j
\tag{9.2}
$$

evaluated at $(x^{(k)}, \lambda^{(k)})$, then setting $x^{(k+1)} = x^{(k)} + \Delta x$ and $\lambda^{(k+1)}$ to be the Lagrange multipliers of the QP subproblem (9.1), the required extension
to Newton's method for solving the non-linear equations (6.3) is obtained. The attraction of this approach, first proposed by Wilson (1963), is that convergence, if it occurs, is usually at a second order rate. Moreover, because the constraints of sub-problem (9.1) are the linear terms of the Taylor series expansion of $c_i(x^{(k)} + p^i)$ about $x^{(k)}$, it follows that linear constraints are satisfied automatically by this approach. Unlike the reduced gradient/gradient projection methods, constraint violations are permitted when non-linearities are present and thus there is no essential requirement to follow highly curved constraints in the early iterations.

Unfortunately there are two serious disadvantages with the approach as described above. One is that, although large errors can be tolerated in the initial approximation $x^{(0)}$, it is usually essential for the initial approximation $x^{(0)}$ to be close to the required solution, $x^*$, for convergence to occur. This is highly unsatisfactory. The second disadvantage is that users of optimization software are rarely prepared to provide second derivative information relevant to their objective and constraint functions! Because of these disadvantages much recent research has been directed towards improving the sequential quadratic programming approach in, for example, Han (1977), Powell (1978, 1981), Bartholomew-Biggs (1982), and Sargent (1982). The dependence on second derivative matrices is removed by using quasi-Newton approximations for the matrix $B(k)$ in the subproblem (9.1) so that only first derivatives are required. Also, convergence from poor starting values $x^{(0)}$ is forced by sometimes insisting that smaller corrections are made when defining $x^{(k+1)}$. This is achieved by setting $x^{(k+1)} = x^{(k)} + \alpha^{(k)} p$, where $0 < \alpha^{(k)} \leq 1$ is chosen to obtain a compromise between reducing $f(x)$ and keeping constraint violations small. At present there is no general consensus of opinion as to the best method of achieving this compromise and alternative strategies are presented in the previously cited papers.

Sequential quadratic programming methods are sometimes referred to as recursive quadratic programming methods. Examples of FORTRAN codes are available from the Harwell library (routine VF02AD) and the Numerical Optimization Centre (routine XROP). They have received very favourable reports in recent comparative studies; for example, the extensive numerical comparisons in Schittkowski (1980).

The computational cost of solving a quadratic programming problem at each iteration can be quite high but usually very few iterations are required, especially when there are many constraints active at the solution. Consequently, sequential quadratic programming methods are best suited to non-linear problems of small to moderate size (less than 50 variables, 100 constraints). Large scale non-linear problems are better handled by the special techniques discussed in the next section.
The methods so far discussed are acceptable for problems of moderate size. Large-scale problems, for example the optimal investment policy problem of Ali (1978) which included 2000 constraints and 4000 variables, present more difficulties, particularly in respect of storage and computation time. To obtain a solution efficiently for these large, usually sparse systems, it is critical to carefully consider both the problem structure and details of implementation. Much time in solving these problems is spent in linear algebraic procedures and data structure operations. A common feature of many large-scale problems is that relatively few of the constraints are non-linear and these constraints include only a subset of the variables, and usually only a small number of variables are included in the non-linear portion of the objective function. It seems appropriate therefore to consider two approaches, either to extend the linear programming techniques developed for large problems to handle nonlinearities in the objective function and constraints, or to specialize the non-linear programming methods previously discussed in this paper by using these well-developed techniques for large linear problems. The first approach is simpler and is considered initially. It is however not as useful as the latter approach, of which a particularly good example is discussed.

A useful technique when solving large linear programming systems is to apply some form of decomposition or partitioning to the problem so as to be able to solve it via the solution of a sequence of smaller subproblems. Lasdon (1970) provides a comprehensive discussion of this approach which relies on the Dantzig-Wolfe (1961) decomposition principle. The problem is represented as a master problem and a set of subproblems, whose objective functions include some variable parameters, termed prices. These are set by the master problem, each subproblem is then solved, and the solutions are returned to the master problem. The master problem combines these solutions with the previous solutions in some optimal way, recomputes the prices and commences the cycle again. This process continues until an optimal solution is attained. In essence, the master problem acts as a co-ordinator of the actions of the subproblems by setting prices on the resources used in each subproblem. When nonlinear terms are involved in the objective function, they are linearized and the approximate linear program is solved, as in Wolfe (1967). The disadvantage is that the problem size is often vastly increased if the nonlinear terms are accurately represented. If however the nonlinear terms of the objective function are functions of just one variable, that is, the objective function \( f(x) \) is separable into the sum of \( n \) functions of just one variable, \( f_i(x_i) \), the process is simpler. The linearization is performed on each of these one variable functions and the subproblems resulting from the decomposition become problems in just one variable, providing the constraints are also separable. Facilities to handle such problems, even if only the objective function is separable, are common in up-to-date linear programming systems (SCICONIC/VM, 1981; TEMPO, 1974).
Another approach is to use a partitioning of the variables into two subsets, changing the values in one subset first, then the other, and iterating between them so as to eventually obtain an optimal solution. Rosen (1963) considers the problem

\[
\begin{align*}
\text{minimize} & \quad c(y)^T x + c_0(y) \\
\text{subject to} & \quad A(y)x - b(y) \geq 0, \text{ where } A(y) \text{ is a square diagonal matrix.}
\end{align*}
\]

It is obvious that for \( y \) fixed, this problem reduces to a linear program, suggesting the choice of subsets to be the variables \( x \) in one, and the variables \( y \), the coupling variables, in the other. For \( y \) fixed, the problem reduces to a set of one variable sub-problems, and a co-ordinating program with a non-linear objective function in the variables \( y \). Benders (1962) discusses another partitioning form for problems of the type

\[
\begin{align*}
\text{minimize} & \quad c^T x + f(y) \\
\text{subject to} & \quad A(x) + F(y) - b > 0 \\
& \quad x \geq 0, \ y \in S, \text{ with } S \text{ typically being the non-negative integers.}
\end{align*}
\]

The problem is solved by varying \( y \), finding the optimal \( x \) for each \( y \), and repeating the process. It is related to the Dantzig-Wolfe decomposition; in fact, they are duals of each other. It is easier to handle a wider range of nonlinearities using Bender's partitioning.

A different approach that uses solely a linear programming system is the method of approximation programming, MAP, developed by Griffith and Stewart (1961), now often referred to as sequential linear programming. It is designed to solve problems of the form

\[
\begin{align*}
\text{minimize} & \quad f(x_N^*) + c^T x_L \\
\text{subject to} & \quad F(x_N^*) + A^* x_L = b \\
& \quad L \leq x_N^* \leq U, \ x_L \geq 0
\end{align*}
\]

A first order approximation to the non-linear functions \( f(x_N) \) and \( F(x_N) \) is made about each estimate \( x_N^{(k)} \) of the solution using the first terms of a Taylor series expansion. The resultant linear programming problem is then solved, and the process is repeated, with the linearization performed about \( x_N^{(k+1)} \). The procedure terminates when successive solutions \( x_N^{(k)} \) show no appreciable change. The method is acceptable if the functions are nearly linear. The stepsize in each successive linear program is restricted and the selection of these limits is difficult in the presence of significant nonlinearities. Too small a limit results in very slow convergence, whereas too large a limit gives an inaccurate approximation. If the functions are highly non-linear, much fine tuning is required to find acceptable stepsize limits.
The decomposition and partitioning approaches tend to be useful only for restricted types of problems. If the nonlinearities are considerable, they tend to be inefficient and time-consuming ways to solve the problem. It is better therefore to solve the problem by using a good non-linear solution method along with the techniques developed for linear programming such as good basis factorization and storage schemes.

11. MINOS AND MINOS/AUGMENTED

The large scale non-linear programming systems, MINOS and MINOS/AUGMENTED developed by Murtagh and Saunders (1978 and 1982) use the theory of non-linear programming, with careful attention paid in the implementations to using well-developed modern linear programming techniques such as basis factorizations and storage schemes for sparse systems. The MINOS system applies to problems with an objective function consisting of a linear term involving usually most of the variables and a non-linear term involving the remainder, and constraints consisting of a set of linear equalities and upper and lower bounds on the variables, as in a typical linear programming problem. For problems with a large number of variables the constraint matrix is almost certainly sparse. The method is based on an active set strategy, with the search direction at each iteration being defined in terms of a group of so-called "superbasic variables", the $s$ variables other than the $m$ basic variables which are not at either of their bounds. (In linear programming $s$ would be zero.) It is easy to show that $s$ is no larger than the number of non-linear variables in the objective function, and thus the search direction can be obtained by the solution of a (usually small) linear system. Suitable quasi-Newton approximations are made to the projected Hessian matrix and are updated for changes in the active set as required. The calculation of the search directions from the solution of the linear system involves the solution of systems such as $Bx = b$, as in linear programming, where $B$ is equivalent to an $m \times m$ basis matrix, and the techniques used are exactly those in large linear programming systems. Unfortunately, the Lagrange multiplier estimates, on which decisions to drop constraints are based, are not as reliable as in the dense case, because of the different way the search direction is developed. The method is however still very good for large problems, particularly when there are significant nonlinearities in the objective function. It is well known that in such a case, linearization of the objective function can cause major difficulties.

When non-linear constraints are present, the problem is considered in the form

$$\begin{align*}
\text{minimize} & \quad f^0(x) + c^T x + d^T y \\
\text{subject to} & \quad f(x) + A_1 y = b_1 \\
& \quad A_2 x + A_3 y = b_2 \\
& \quad l < \begin{bmatrix} x \\ y \end{bmatrix} < u
\end{align*} \quad (11.1)$$
where usually the dimension of $x$ is much smaller than $y$. The solution process here consists of a sequence of major iterations, each one involving the linearization of the non-linear constraints at some point $x^{(k)}$. The objective function used in these linearized subproblems is a modified augmented Lagrangian function, with a penalty term added to maintain a positive definite Hessian matrix and enhance convergence from a poor starting point. The nonlinearities in this function occur only in the $x$ variables, as in the original objective function. The linearized subproblems are of course solved by the MINOS procedure for linear constraints. The use of Lagrange multiplier estimates improves the speed of solution, relative to simple sequential linearization as in the method of approximation programming.

These two procedures, MINOS and MINOS/AUGMENTED, are probably the best codes currently available for solving large scale non-linear programming problems. Despite the fact that they were developed essentially for problems with mostly linear variables, they also perform surprisingly well on several of the well-known completely non-linear test problems.

12. SUMMARY AND CONCLUDING REMARKS

We have indicated typical applications in which non-linear programming problems may arise, and have discussed various solution methods, including the advantages and disadvantages of each class. We now describe briefly a selection of optimization software likely to be readily available in New Zealand including information on such matters as problem types for which they are useful, ease of use, and quality of the code. We conclude with some helpful remarks and hints for solving non-linear programming problems successfully.

The NAG subroutine library, a comprehensive library of Fortran and Algol subroutines for numerical computation, includes five subroutines specifically designed for non-linear programming problems. Routine E04HAF uses a penalty function method, and is acceptable for small problems with fairly simple objective and constraint functions. The four routines, E04UAF, E04VAF, E04VBF, E04WAF, based on the augmented Lagrangian method, are however likely to solve even such simple problems better and are to be recommended in preference to E04HAF. Four routines are given, to be used according to the amount of information about the problem functions the user is prepared and able to provide. For example, if no derivative information is available, E04UAF should be used. It constructs finite difference approximations as required, and is preferable to E04HAF. These augmented Lagrangian methods are useful for problems of moderate size with perhaps $\frac{1}{3}n$ constraints.
active at the solution. They are also appropriate for larger problems with only a few equality constraints and mostly non-linear constraints. If the constraints are all linear other methods, such as MINOS are likely to be more efficient. The documentation for this library is very good and examples are provided for each routine. They are however not stand-alone codes, but require to be called from a user-supplied Fortran or Algol program.

The two constrained optimization Fortran subroutines, VF02AD and VF01A, available in the Harwell (AERE) library are of a similar type. The documentation provided is less complete for an unskilled user. Routine VF01A is an augmented Lagrangian method for problems with up to 25 variables and 50 equality and inequality constraints. It requires derivative information for both the objective and constraint functions. The method is appropriate for the same types of problems as the four NAG augmented Lagrangian methods. Routine VF02AD is a sequential quadratic programming method, and is useful for small to moderate sized problems with a large number of constraints active at the solution, such as when most of the constraints are equalities, not inequalities.

Some numerical software is available from the Numerical Optimization Centre at Hatfield Polytechnic. The subroutine XROP is a sequential quadratic programming method, programmed as a Fortran subroutine. It again is appropriate for small to medium size problems, but because of the quadratic programming solution method used, is likely to be most efficient when only a small number of constraints are active at the solution.

The GRGA generalized reduced gradient code of Abadie is a good robust routine. It again is a Fortran subroutine, with many parameters given default values, which users need alter only if they require different values. A separate subroutine to evaluate the objective and constraint functions and their derivatives is required, although the user may indicate that the finite difference routine available should be used to approximate the derivatives. On comparative tests (Schittkowski, 1980), taking into account robustness, ease of use, ability to handle difficult problems, this code is usually ranked as being very good. Given the underlying method, it generally works very well on problems with only mildly non-linear constraints.

The two routines, MINOS and MINOS/AUGMENTED, are both stand-alone systems, similar in structure to commercially-available linear programming systems. The user is required to provide an MPS-type file containing information on the linear part of the problem, subroutines for calculating the non-linear functions and derivatives, if a finite difference method is not invoked, and a file containing information on various system parameters, such as problem size, frequency of iteration information, and tolerances. Much of this file can be left at default settings if required. The documentation available is good with a series of well-constructed examples which clearly indicate the setting up of the problem for solution and the possible choices available to the user. For people familiar with linear programming systems, this
non-linear programming system is very easy to use. It is very good for large problems, particularly when only a small proportion of the variables and constraints are non-linear. The method is also acceptable on totally non-linear problems of small to moderate size.

Of course, even with the best solution method, it is difficult to solve a poorly formulated problem. It is important to consider the formulation stage carefully when trying to solve practical problems using the techniques of non-linear programming. The model to be optimized should be built by striking a reasonable balance between improved accuracy of the model and increased ease of optimization. With the availability of robust codes there is no longer the need to approximate highly non-linear functions by linear ones; this flexibility should result in better solutions to such problems than has previously been possible.

The question of scaling of variables and constraints is often important. Scaling is used to convert from units which typically reflect the physical nature of the problem to units that display certain desirable properties during the optimization process. Usually, the variables in the scaled problem should be of a similar magnitude and of order unity in the region of interest. Implicit definitions of "small" and "large" are usually made in optimization systems, so variables of widely different orders of magnitude are likely to create difficulties. Care in the formulation of the problem can usually avoid difficulties with poor scaling. Gill et al (1981) discuss this matter in more detail.

If the user can provide additional information about his problem, the solution process is usually made easier. In particular, if first derivative information is available, it is beneficial to provide it. Most systems now incorporate finite difference routines which are used to check the user-provided derivative routines or to generate derivative approximations as required. This automatic checking is extremely valuable since mistakes in programming derivatives are often the most common sources of failure in using non-linear optimization codes.

All methods described in this paper assume that the objective and constraint functions are sufficiently smooth. Techniques to handle non-differentiable optimization problems are now being developed, but are not yet widely available. Fletcher (1981) discusses this problem in more detail.

Finally, the interested reader can find additional information on both the theory and practice of non-linear programming in the four recent books, Fletcher (1981), Gill et al (1981), Murtagh (1980), and Schittkowski (1980).
APPENDIX: SOFTWARE SOURCES

Codes discussed in the text of this paper are available from the following sources. For a more comprehensive list of available software see Schittkowski (1980).

GRGA :  J. Abadie, University of Paris VI, Institut de Programmation, 4, Place Jussieu, Paris, France.

Harwell :  Computer Science and Systems Division, AERE, Harwell, Oxfordshire, United Kingdom.

MINOS :  Systems Optimization Laboratory, Operations Research Dept., Stanford University, Stanford, CA 94305, U.S.A.

NAG :  The Numerical Algorithms Group Ltd., NAG Central Office, Mayfield House, 265 Banbury Road, Oxford OX2 7DE, United Kingdom

NOC :  Numerical Optimization Centre, The Hatfield Polytechnic, 19 St. Albans Road, Hatfield, Herts, United Kingdom

REFERENCES


Heltne, D.R. and Liittschwager, J.M. (1975), "Users' Guide for GRG73" and Technical Appendices to GRG73", University of Iowa, Iowa City, USA.


Powell, M.J.D. (1981), "Extensions to subroutine VF02AD", Report No. DAMTP 1981/NA8, Department of Applied Mathematics and Theoretical Physics, University of Cambridge.


