Chapter 9
Optimization

1 Scope of the Chapter

This chapter provides procedures for the numerical solution of optimization problems.

An optimization problem involves minimizing a function (called the *objective function*) of several variables, possibly subject to restrictions on the values of the variables defined by a set of *constraints*. Unless stated otherwise, NAG *fl*90 procedures are concerned with *minimization* only, since the problem of maximizing a given objective function $F(x)$ is equivalent to minimizing $-F(x)$.

2 Available Modules

Module 9.1: `nag_qp` — **Linear and Quadratic Programming**

Provides a procedure for computing

- a constrained minimum of a linear or quadratic objective function subject to a set of general linear constraints and/or bounds on the variables. It treats all matrices as dense and hence is not intended for large sparse problems.

Module 9.2: `nag_nlin_lsq` — **Unconstrained Nonlinear Least-squares**

Provides procedures for computing

- an unconstrained minimum (x^*) of a sum of squares of m nonlinear functions in n variables (where $m \geq n$),
- estimates of elements of the variance–covariance matrix at x^*.

Module 9.3: `nag_nlp` — **Nonlinear Programming**

Provides a procedure for computing

- a constrained minimum of an arbitrary smooth objective function subject to a set of general nonlinear constraints and/or linear constraints and/or bounds on the variables. It treats all matrices as dense and hence is not intended for large sparse problems.

Module 9.4: `nag_con_nlin_lsq` — **Constrained Nonlinear Least-squares**

Provides procedures for computing

- a constrained minimum of a smooth (nonlinear) sum of squares function subject to a set of general nonlinear constraints and/or linear constraints and/or bounds on the variables. It treats all matrices as dense and hence is not intended for large sparse problems.

Module 9.5: `nag_uv_min` — **Univariate Minimization**

Provides a procedure for computing

- a minimum of a continuous function of a single variable in a given finite interval using safeguarded polynomial approximation.

Module 9.6: `nag_nlp_sparse` — **Sparse Nonlinear Programming**

Provides a procedure for computing

- a constrained minimum (or maximum) of an arbitrary smooth objective function subject to a set of general nonlinear constraints and/or linear constraints and/or bounds on the variables.

3 Background

3.1 Types of Optimization Problems

The solution of optimization problems by a single, all-purpose, method is cumbersome and inefficient. Optimization problems are therefore classified into particular categories, where each category is defined by the properties of the objective function and the constraints, as illustrated by the following examples:

Properties of Objective Function	*Properties of Constraints*
Nonlinear	Nonlinear
Sums of squares of nonlinear functions	Sparse linear
Quadratic	Linear
Sums of squares of linear functions	Bounds
Linear	None

For instance, a specific problem category involves the minimization of a nonlinear objective function subject to bounds on the variables. Not all categories are catered for by specific procedures in the current version of the Library; however, the long-term objective is to provide a comprehensive set of procedures to solve problems in all such categories.

3.2 Geometric Representation and Terminology

To illustrate the nature of optimization problems it is useful to consider the following example:

$$F(x) = e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1).$$

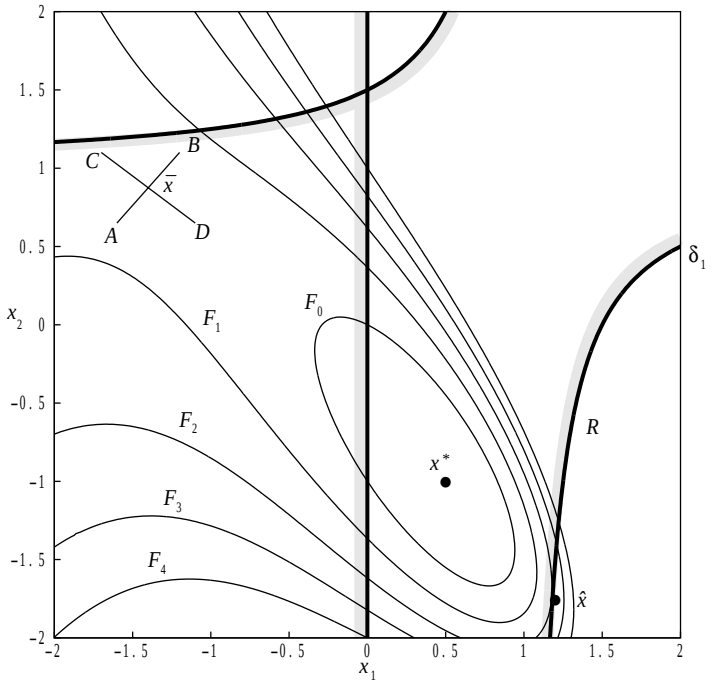

Figure 1

Figure 1 is a contour diagram of $F(x)$. The contours labelled $F_0, F_1, \ldots, F_4$ are isovalue contours, or lines along which $F(x)$ takes specific constant values. The point $x^* = (\frac{1}{2}, -1)^T$ is a *local unconstrained*

minimum, i.e., the value of $F(x^*)$ $(= 0)$ is less than at all the neighbouring points. A function may have several such minima. The lowest of the local minima is termed a *global minimum*. In the problem illustrated in Figure 1, x^* is the only local minimum. A point, such as $\bar{x}$, is said to be a *saddle point* because it is a minimum along the line AB, but a maximum along CD.

If we add the constraint $x_1 \geq 0$ (a simple bound) to the problem of minimizing $F(x)$, the solution remains unaltered. In Figure 1 this constraint is represented by the straight line passing through $x_1 = 0$, and the shading on the line indicates the unacceptable region (i.e., $x_1 < 0$). The region in R^n satisfying the constraints of an optimization problem is termed the *feasible region*. A point satisfying the constraints is defined as a *feasible point*.

If we add the nonlinear constraint $c_1(x) : x_1 + x_2 - x_1 x_2 - \frac{3}{2} \geq 0$, represented by the curved shaded line in Figure 1, then x^* is not a feasible point because $c_1(x^*) < 0$. The solution of the new constrained problem is $\hat{x} \simeq (1.1825, -1.7397)^T$, the feasible point with the smallest function value (where $F(\hat{x}) \simeq 3.0607$).

The vector of first partial derivatives of $F(x)$ is called the *gradient vector*, and is denoted by $g(x)$, i.e.,

$$g(x) = \left(\frac{\partial F(x)}{\partial x_1}, \frac{\partial F(x)}{\partial x_2}, \ldots, \frac{\partial F(x)}{\partial x_n} \right)^T .$$

The gradient vector is of importance in optimization because it must be zero at an unconstrained minimum of any function with continuous first derivatives.

The matrix of second partial derivatives of a function is termed its *Hessian matrix*. The Hessian matrix of $F(x)$ is denoted by $G(x)$, and its (i, j)th element is given by $\partial^2 F(x)/\partial x_i \partial x_j$. If $F(x)$ has continuous second derivatives, then $G(x)$ must be positive semi-definite at any unconstrained minimum of $F(x)$.

The vector of first partial derivatives of the constraint $c_i(x)$ is denoted by

$$a_i(x) = \left(\frac{\partial c_i(x)}{\partial x_1}, \frac{\partial c_i(x)}{\partial x_2}, \ldots, \frac{\partial c_i(x)}{\partial x_n} \right)^T .$$

The matrix whose columns are the vectors $\{a_i\}$ is termed the *matrix of constraint normals*.

At a point $\hat{x}$, the vector $a_i(\hat{x})$ is orthogonal (normal) to the isovalue contour of $c_i(x)$ passing through $\hat{x}$; this relationship is illustrated for a two-dimensional function in Figure 2.

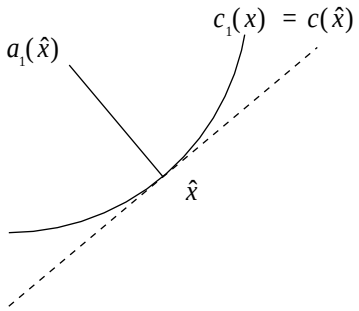

Figure 2

Note that if $c_i(x)$ is a linear constraint involving $a_i^T x$, then its vector of first partial derivatives is simply the vector a_i.

3.3 Sufficient Conditions for a Solution

All nonlinear functions will be assumed to have continuous second derivatives in the neighbourhood of a solution.

Unconstrained minimization

The following conditions are sufficient for the point x^* to be an unconstrained local minimum of $F(x)$:

(a) $\|g(x^*)\| = 0$; and

(b) $G(x^*)$ is positive definite,

where $\|g\|$ denotes the Euclidean length of g.

Linearly constrained minimization

At a solution x^* of a linearly constrained problem, the constraints which hold as equalities are called the *active* or *binding* constraints. Assume that there are t active constraints at the solution x^*, and let $\hat{A}$ denote the matrix whose columns are the columns of A corresponding to the active constraints, with $\hat{b}$ the vector similarly obtained from b; then

$$\hat{A}^T x^* = \hat{b}.$$

The matrix Z is defined as an n by $(n - t)$ matrix satisfying:

$$\hat{A}^T Z = 0;$$

$$Z^T Z = I.$$

The columns of Z form an orthonormal basis for the set of vectors orthogonal to the columns of $\hat{A}$.

Define

$$g_Z(x) = Z^T g(x), \text{ the projected gradient vector of } F(x);$$

$$G_Z(x) = Z^T G(x) Z, \text{ the projected Hessian matrix of } F(x).$$

At the solution of a linearly constrained problem, the projected gradient vector must be zero, which implies that the gradient vector $g(x^*)$ can be written as a linear combination of the columns of $\hat{A}$, i.e.,

$$g(x^*) = \sum_{i=1}^{t} \lambda_i^* \hat{a}_i = \hat{A}\lambda^*.$$

The scalar λ_i^* is defined as the *Lagrange multiplier* corresponding to the ith active constraint. A simple interpretation of the ith Lagrange multiplier is that it gives the gradient of $F(x)$ along the ith active constraint normal; a convenient definition of the Lagrange multiplier vector (although not a recommended method for computation) is:

$$\lambda^* = (\hat{A}^T \hat{A})^{-1} \hat{A}^T g(x^*).$$

Sufficient conditions for x^* to be the solution of a linearly constrained problem are:

(a) x^* is feasible, and $\hat{A}^T x^* = \hat{b}$; and

(b) $\|g_Z(x^*)\| = 0$, or equivalently, $g(x^*) = \hat{A}\lambda^*$; and

(c) $G_Z(x^*)$ is positive definite; and

(d) $\lambda_i^* > 0$ if λ_i^* corresponds to a constraint of the form $\hat{a}_i^T x^* \geq \hat{b}_i$.

$\lambda_i^* < 0$ if λ_i^* corresponds to a constraint of the form $\hat{a}_i^T x^* \leq \hat{b}_i$.

The sign of λ_i^* is immaterial for equality constraints, which by definition are always active.

Nonlinearly constrained minimization

For nonlinearly constrained problems, much of the terminology is defined exactly as in the linearly constrained case. The set of active constraints at x again means the set of constraints that hold as equalities at x, with corresponding definitions of $\hat{c}$ and $\hat{A}$: the vector $\hat{c}(x)$ contains the active constraint functions, and the columns of $\hat{A}(x)$ are the gradient vectors of the active constraints. As before, Z is defined in terms of $\hat{A}(x)$ as a matrix such that:

$$\hat{A}^T Z = 0;$$

$$Z^T Z = I$$

where the dependence on x has been suppressed for compactness.

The projected gradient vector $g_Z(x)$ is the vector $Z^T g(x)$. At the solution x^* of a nonlinearly constrained problem, the projected gradient must be zero, which implies the existence of Lagrange multipliers corresponding to the active constraints, i.e., $g(x^*) = \hat{A}(x^*)\lambda^*$.

The *Lagrangian function* is given by:

$$L(x, \lambda) = F(x) - \lambda^T c(x).$$

We define $g_L(x)$ as the gradient of the Lagrangian function, $G_L(x)$ as its Hessian matrix and $\hat{G}_L(x)$ as its projected Hessian matrix, i.e., $\hat{G}_L = Z^T G_L Z$.

Sufficient conditions for x^* to be the solution of a nonlinearly constrained problem are:

(a) x^* is feasible, and $\hat{c}(x^*) = 0$; and

(b) $\|g_Z(x^*)\| = 0$, or equivalently, $g(x^*) = \hat{A}(x^*)\lambda^*$; and

(c) $\hat{G}_L(x^*)$ is positive definite; and

(d) $\lambda_i^* > 0$ if λ_i^* corresponds to a constraint of the form $\hat{c}_i \geq 0$.

The sign of λ_i^* is immaterial for equality constraints, which by definition are always active.

Note that condition (b) implies that the projected gradient of the Lagrangian function must also be zero at x^*, since the application of Z^T annihilates the matrix $\hat{A}(x^*)$.

3.4 Scaling

Scaling (in a broadly defined sense) often has a significant influence on the performance of optimization methods. Since convergence tolerances and other criteria are necessarily based on an implicit definition of 'small' and 'large', problems with unusual or unbalanced scaling may cause difficulties for some algorithms. Although there are currently no user-callable scaling procedures in the Library, scaling is automatically performed by default in the procedure which solves sparse nonlinear programming problems. The following sections present some general comments on problem scaling.

Transformation of variables

One method of scaling is to transform the variables from their original representation, which may reflect the physical nature of the problem, to variables that have certain desirable properties in terms of optimization. It is generally helpful for the following conditions to be satisfied:

(a) the variables are all of similar magnitude in the region of interest;

(b) a fixed change in any of the variables results in similar changes in $F(x)$. Ideally, a unit change in any variable produces a unit change in $F(x)$;

(c) the variables are transformed so as to avoid cancellation error in the evaluation of $F(x)$.

Normally, users should restrict themselves to linear transformations of variables, although occasionally nonlinear transformations are possible. The most common such transformation (and often the most appropriate) is of the form

$$x_{\text{new}} = Dx_{\text{old}},$$

where D is a diagonal matrix with constant coefficients. Our experience suggests that more use should be made of the transformation

$$x_{\text{new}} = Dx_{\text{old}} + v,$$

where v is a constant vector.

Scaling the objective function

The objective function has already been mentioned in the discussion of scaling the variables. The solution of a given problem is unaltered if $F(x)$ is multiplied by a positive constant, or if a constant value is added to $F(x)$. It is generally preferable for the objective function to be of the order of unity in the region of interest; thus, if in the original formulation $F(x)$ is always of the order of 10^{+5} (say), then the value of $F(x)$ should be multiplied by 10^{-5} when evaluating the function within an optimization procedure. If a constant is added or subtracted in the computation of $F(x)$, usually it should be omitted; i.e., it is better to formulate $F(x)$ as $x_1^2 + x_2^2$ rather than as $x_1^2 + x_2^2 + 1000$ or even $x_1^2 + x_2^2 + 1$. The inclusion of such a constant in the calculation of $F(x)$ can result in a loss of significant figures.

Scaling the constraints

A 'well scaled' set of constraints has two main properties. Firstly, each constraint should be well conditioned with respect to perturbations of the variables. Secondly, the constraints should be *balanced* with respect to each other, i.e., all the constraints should have 'equal weight' in the solution process.

The solution of a linearly or nonlinearly constrained problem is unaltered if the ith constraint is multiplied by a positive weight w_i. At the approximation of the solution determined by a Library procedure, any active linear constraints will (in general) be satisfied 'exactly' (i.e., to within the tolerance defined by `EPSILON(1.0_wp)`) if they have been properly scaled. This is in contrast to any active nonlinear constraints, which will not (in general) be satisfied 'exactly' but will have 'small' values (for example, $\hat{c}_1(x^*) = 10^{-8}$, $\hat{c}_2(x^*) = -10^{-6}$, and so on). In general, this discrepancy will be minimized if the constraints are weighted so that a unit change in x produces a similar change in *each* constraint.

A second reason for introducing weights is related to the effect of the size of the constraints on the Lagrange multiplier estimates and, consequently, on the active set strategy. This means that different sets of weights may cause an algorithm to produce different sequences of iterates. Additional discussion is given in Gill *et al.* [1].

3.5 Analysis of Computed Results

Convergence criteria

The convergence criteria inevitably vary from procedure to procedure, since in some cases more information is available to be checked (for example, is the Hessian matrix positive definite?), and different checks need to be made for different problem categories (for example, in constrained minimization it is necessary to verify whether a trial solution is feasible). Nonetheless, the underlying principles of the various criteria are the same; in non-mathematical terms, they are:

(a) is the sequence $\left\{x^{(k)}\right\}$ converging?

(b) is the sequence $\left\{F^{(k)}\right\}$ converging?

(c) are the necessary and sufficient conditions for the solution satisfied?

The decision as to whether a sequence is converging is necessarily speculative. The criterion used in the present procedures is to assume convergence if the relative change occurring between two successive iterations is less than some prescribed quantity. Criterion (c) is the most reliable but often the conditions cannot be checked fully because not all the required information may be available.

Checking results

Little *a priori* guidance can be given as to the quality of the solution found by a nonlinear optimization algorithm, since no guarantees can be given that the methods will not fail. Therefore, you should always check the computed solution even if the procedure reports success. Frequently a 'solution' may have been found even when the procedure does not report a success. The reason for this apparent contradiction is that the procedure needs to assess the accuracy of the solution. This assessment is not an exact process and consequently may be unduly pessimistic. Any 'solution' is in general only an approximation to the exact solution, and it is possible that the requirements for 'success' are too stringent.

Further confirmation can be sought by trying to check whether or not convergence tests are almost satisfied, or whether or not some of the sufficient conditions are nearly satisfied. When it is thought that a procedure has returned a non-zero value of `error%level` only because the requirements for 'success' were too stringent it may be worth restarting with increased convergence tolerances.

Confidence in a solution may be increased by re-solving the problem with a different initial approximation to the solution. See Section 8.3 of Gill *et al.* [1] for further information.

Monitoring progress

Some of the procedures in this chapter have facilities to allow you to monitor the progress of the minimization process, and you are encouraged to make use of these facilities. Monitoring information can be a great aid in assessing whether or not a satisfactory solution has been obtained, and in indicating difficulties in the minimization problem or in the ability of the procedure to cope with the problem.

The behaviour of the function, the estimated solution and first derivatives can help in deciding whether a solution is acceptable and what to do in the event of a procedure returning with `error%level` > 0.

3.6 Function Evaluations at Infeasible Points

All the procedures for constrained problems will ensure that any evaluations of the objective function occur at points which *approximately* satisfy any *simple bounds* or *linear constraints*. Satisfaction of such constraints is only approximate because procedures which estimate derivatives by finite differences may require function evaluations at points which just violate such constraints even though the current iteration just satisfies them.

No attempt is made to ensure that the current iteration satisfies any nonlinear constraints. In order to prevent the objective function being evaluated outside some known region (where it may be undefined or not practically computable), you may try to confine the iteration within this region by imposing suitable simple bounds or linear constraints (but beware as this may create new local minima where these constraints are active).

3.7 Related Problems

Module `nag_lin_lsq` (6.4) contains a procedure which solves the *linear least-squares* problem

$$\text{minimize } \sum_{i=1}^{m} r_i(x)^2, \quad \text{where } \; r_i(x) = b_i - \sum_{j=1}^{n} a_{ij}x_j.$$

4 References

[1] Gill P E, Murray W and Wright M H (1981) *Practical Optimization* Academic Press