AD-A087 555 STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB F/G 12/1
METHODS FOR LARGE-SCALE NONLINEAR OPTIMIZATION.(U)
MAY 80 P E GILL, W MURRAY, M A SAUNDERS DAAG29-79-C-0110
UNCLASSIFIED SOL-80-8 ARO-16470.13-M NL

END
DATE FILMED
9-80
DTIC

ARO 16470.13-M

Systems
Optimization
Laboratory

LEVEL II

Department of Operations Research
Stanford University
Stanford, CA 94305

(12)

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

1473 in back

METHODS FOR LARGE-SCALE NONLINEAR OPTIMIZATION

by

Philip E. Gill, Walter Murray,
Michael A. Saunders, and Margaret H. Wright

TECHNICAL REPORT SOL 80-8
May 1980

Research and reproduction of this report were supported by the Department of Energy Contract DE-AC03-76-SF00326, PA No. DE-AT-0376ER72018; the National Science Foundation Grants MCS-7926009 and ENG77-06761; the Office of Naval Research Contract N00014-75-C-0267; and the U.S. Army Research Office Contract DAAG29-79-C-0110.

Methods for Large-scale Nonlinear Optimization†

Philip E. Gill, Walter Murray, Michael A. Saunders and Margaret H. Wright
Systems Optimization Laboratory
Department of Operations Research
Stanford University
Stanford, CA 94305

ABSTRACT

The application of optimization to electrical power technology often requires the numerical solution of systems that are very large and possibly non-differentiable. A brief survey of the state of the art of numerical optimization is presented in which those methods that are directly applicable to power system problems will be highlighted. The areas of current research that are most likely to yield direct benefit to practical computation are identified. The paper concludes with a survey of available software.

† Presented at the SIAM conference: "Electric Power Problems: the Mathematical Challenge," Seattle, March 1980.

1. Introduction

The application of optimization to the electrical power industry has produced problems in which the matrices of constraint gradients and second derivatives are large and sparse. There are two approaches to solving large-scale problems: methods for small dense problems can be adapted to solve larger problems by using sparse matrix techniques; or special-purpose methods can be used to reduce the problem to a related sequence of simpler problems. In this paper we shall review methods that follow the first approach. In the first three sections we define a number of "prototype" methods that give the best rate of convergence when applied to small problems, but are easily adapted to cater for sparsity. It is demonstrated how sparse matrix techniques can be used to extend the prototype algorithms to solve several important types of large-scale problems — particularly problems in which a small number of the variables appear nonlinearly.

Unfortunately many electrical power problems occur with a large number of nonlinear constraints (see Biggs and Laughton, 1976). This is precisely the class of problems which are the most difficult to solve numerically. In Section 5 we shall discuss current research that shows promise of allowing such difficult problems to be solved.

The mathematical description of the problem of concern is:

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}} && F(x) \\ &\text{subject to} && c_i(x) \geq 0, \quad i = 1, 2, \ldots, m. \end{aligned}$$

where $F(x)$ (the objective function) and $\{c_i(x)\}$ (the constraint functions) are twice continuously differentiable.

In the most difficult problems, each $c_i(x)$ is a nonlinear function. Algorithms will also be described for problems in which the set $\{c_i(x)\}$ is empty — the *unconstrained optimization problem*; and the constraints are all linear functions of x — the *linearly constrained problem*.

The n-dimensional column vector $g(x)$ denotes the gradient of $F(x)$, and $G(x)$ denotes the Hessian matrix of $F(x)$. The vector $\overset{*}{x}$ refers to a solution of the optimization problem.

It is impossible for a short survey to give an adequate description of all the recent developments in a field of study that has grown substantially in complexity and range during the last few years. It is intended, however, that the references cited in the text will enable the reader to become familiar with the more substantial and detailed accounts of the subject.

2. Methods for dense problems

All the algorithms discussed in this paper generate a sequence of estimates $\{x_k\}$

of the optimal point $\overset{*}{x}$. Many do so by generating a search direction p_k and a step length α_k such that

$$x_{k+1} = x_k + \alpha_k p_k;$$

those that have the additional property that $F(x_{k+1}) < F(x_k)$ are known as *descent methods*. A complete description of the methods used to compute α_k is beyond the scope of this paper. However, it must be emphasized that the robustness of an optimization method will depend upon the method used to compute α_k (see Ortega and Rheinboldt, 1970; Gill *et al.*, 1979).

2.1 Methods for unconstrained minimization.

A useful technique in the optimization of differentiable functions is the utilization of a local *quadratic* model of the objective function. Methods based on such approximations should be expected to work efficiently for quadratic functions. One obvious candidate for a quadratic model of the objective function is the function obtained by taking the first three terms of the Taylor-series expansion about the current point, i.e.

$$F(x_k + p) \approx F_k + g_k^T p + \frac{1}{2} p^T G_k p,$$

where G_k denotes the Hessian matrix evaluated at the point x_k. The idea underlying many methods for constrained minimization is that this approximation is a reasonable model of the objective function, in terms of finding a local minimum. To make this analogy, it is helpful to consider the quadratic model in terms of p rather than x. Consider the quadratic function

$$Q(p) = g_k^T p + \frac{1}{2} p^T G_k p,$$

for a fixed vector g_k and symmetric matrix G_k. If G_k is positive definite, $Q(p)$ has a unique minimum at the stationary point p_k that satisfies $G_k p_k + g_k = 0$, or equivalently

$$G_k p_k = -g_k. \tag{1}$$

This choice of search direction is not suitable for general application since any possible indefiniteness in G_k may result in p_k not being a descent direction, i.e. there may not exist an α such that $F(x_k + \alpha p_k) < F_k$. The class of *modified Newton methods* is characterized by the definition of a positive-definite matrix $\overline{G}_k$ which is equal to to G_k on those occasions where G_k is positive definite, but is equal to a related positive-definite matrix otherwise (see Gill and Murray, 1974a; Moré and Sorenson, 1979).

If it is not possible to provide a subroutine to evaluate the second derivatives, an algorithm with almost identical performance to that of Newton's method can be obtained by finding the columns of the Hessian using finite differences of the gradient vector. For details see Gill and Murray (1974b).

A fundamental advantage of Newton-type methods is that under certain mild restrictions on the objective function, they can be shown to be convergent to a *local minimum* of F. However, this advantage is obtained at the cost of evaluating either the second derivatives or $n+1$ gradients at each iteration. Moreover, it is necessary to solve the set of linear equations (1) from scratch at each iteration. The class of *quasi-Newton methods* was designed to avoid both of these problems (see Dennis and Moré, 1977; Brodlie, 1978). Here, a positive-definite approximate Hessian B_k is known at the k-th iteration and the direction of search is given by the solution of the "quadratic program":

$$\underset{p\in\mathbb{R}^n}{\text{minimise}}\ \frac{1}{2}p^TB_kp+g_k^Tp.$$

On completion of the k-th iteration, a matrix B_{k+1} is computed such that $B_{k+1}-B_k$ is a matrix of low rank — usually of rank one or two, and B_{k+1} satisfies the *quasi-Newton condition*

$$B_{k+1}(x_{k+1}-x_k)=g_{k+1}-g_k.$$

This condition is approximately satisfied by the true Hessian in the neighbourhood of the solution. Although there are an infinite number of possible definitions of B_{k+1} it is generally accepted that the *BFGS formula*

$$B_{k+1}=B_k+\frac{1}{g_k^Tp_k}g_kg_k^T+\frac{1}{\alpha_ky_k^Tp_k}y_ky_k^T,$$

where $y_k=g_{k+1}-g_k$, consistently gives good results. Since the approximate Hessian alters only by a matrix of rank two at each iteration it is possible to update an invertible form of B_k. This form may be either a triangular factorization or an explicit inverse. For numerical computation it is preferable to recur the triangular factorisation of B_k (see Gill and Murray, 1972, 1978, for more details).

All the methods discussed so far require the storage of an $n\times n$ approximate Hessian matrix. *Conjugate-gradient methods* require the storage of only a few n-vectors. The following algorithm was suggested by Fletcher and Reeves (1964). During the first iteration, p_k is just the steepest-descent direction $-g(x_0)$. After the computation of α_k, the direction of search for the next iteration is found from the formula

$$p_{k+1}=-g_{k+1}+\beta_kp_k,$$

where $\beta_k = \|g_{k+1}\|_2^2/\|g_k\|_2^2$. If α_k is computed as the minimum of $F(x_k + \alpha p_k)$ and the algorithm is used to minimize a quadratic function $F(x) = c^T x + \frac{1}{2}x^T Q x$ with Q a symmetric positive-definite matrix and c an n-vector, the directions obtained from this algorithm are identical to those of both the Hestenes and Stiefel conjugate-gradient method for solving the linear equations $Qx = -c$ (Hestenes and Stiefel, 1952) and the BFGS quasi-Newton method. Unfortunately, if the approximate Hessian matrix can be stored, the Fletcher-Reeves algorithm is generally inferior to the BFGS quasi-Newton algorithm in terms of the number of times that F is evaluated. However, recent research on conjugate-gradient-type methods has considerably increased their robustness and efficiency. This work has mainly consisted of deriving methods in which p_{k+1} is computed from the gradients of the previous r iterations by solving equations of the form

$$\left(D_{k+1} + \sum_{i=k-r+1}^{k} (\gamma_i g_i g_i^T - \sigma_i y_i y_i^T)\right) p_{k+1} = -g_{k+1},$$

where D_{k+1} is a diagonal matrix. It can be shown that the Fletcher-Reeves algorithm is the member of this class of methods obtained by using an exact linear search with $r = 1$ and D_{k+1} equal to the identity matrix. For more details see Shanno (1978); Nazareth and Nocedal (1979); Gill and Murray (1979); and Nocedal (1979).

Nonlinear conjugate-gradient methods behave in a similar way to conjugate-gradient methods for the solution of systems of linear equations. These methods work best on problems whose Hessian matrices have sets of clustered eigenvalues. On more general problems, however, even the best method may require a prohibitively large number of iterations. For general problems (i.e. problems without clustered eigenvalues) conjugate-gradient-type methods require significantly more function evaluations than quasi-Newton methods.

2.2 Methods for linearly constrained minimization.

It is common for optimisation problems to include linear constraints of the form:

(1) *simple bounds* on the variables,

$$l_i \leq x_i \leq u_i \qquad i = 1, \ldots, n;$$

(2) *linear constraints* (equality or inequality),

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n \left\{\begin{matrix} \geq \\ = \\ \leq \end{matrix}\right\} \beta.$$

The most successful algorithms for linearly constrained minimisation are members of the class of "active set" methods. At each iteration, a particular subset of the constraints are treated as "active". The search direction is then computed so as to stay "on" these constraints, adding new ones to the active set as they are encountered. The search direction on the set of active constraints is found by solving a linearly constrained problem with the active constraints treated as equalities. Suppose that there are t linearly independent active constraints and that their coefficients comprise the rows of the matrix $\hat{A}$. We then need to solve the equality-constraint problem

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad F(x) \\ &\text{subject to} \quad \hat{A}x = \hat{b}. \end{aligned} \tag{2}$$

Given any feasible point x_k we require a vector p_k such that $x_k + p_k$ is feasible and close to a minimum of (2). This gives the following quadratic program for p_k:

$$\begin{aligned} &\underset{p\in\mathbb{R}^n}{\text{minimize}} \quad \tfrac{1}{2}p^T B_k p + g_k^T p \\ &\text{subject to} \quad \hat{A}p = 0. \end{aligned} \tag{3}$$

Here, as in the unconstrained case, B_k denotes some approximation to the Hessian matrix of F. The t equality constraints have the effect of reducing the dimensionality of the optimization problem to $n - t$, as follows. Any vector satisfying the constraints of (3) can be written in the form $p = Zp_Z$, where Z is an $n \times (n - t)$ matrix whose columns form a basis for the set of vectors orthogonal to $\{\hat{a}_i\}$. Written in terms of p_Z, the quadratic program (3) becomes an *unconstrained* problem with optimum given by the solution of the linear equations

$$Z^T B_k Z p_Z = -Z^T g_k.$$

The solution p_k of (3) is recovered as $p_k = Zp_Z$. When second derivatives are known, $G(x_k)$ can often be used as B_k. In this case the matrix $Z^T G_k Z$ is defined as the *projected Hessian matrix* and is denoted by G_Z. Similarly, the $(n - t)$-vector $Z^T g$ is defined as the *projected gradient* and is denoted by g_Z (see Gill and Murray, 1974c).

The terms "gradient projection" and "reduced gradient" are often used to refer to this problem transformation, which effectively "projects" F into a "reduced" subspace. Alternatively, one can consider that these methods generate iterates that remain "on" the constraints while moving to decrease the objective function.

Any vector u in $\mathbb{R}^n$ can be written in the form $u = \hat{A}^T u_1 + Zu_2$. At a solution of the equality-constraint problem (2) the projected gradient will be zero, which implies that the gradient vector has no component in the space spanned

by Z. Thus there must exist t scalars λ_j such that

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

The t-vector λ is known as the vector of *Lagrange multipliers*. Note that λ is the solution of a set of over-determined linear equations.

When a minimum on a particular set of active constraints has been found, a decision must be taken as to which constraint (if any) should be deleted from the active set. It can be shown that a Lagrange multiplier gives a first-order estimate to the change in F for a unit perturbation in the constraint. A positive Lagrange multiplier λ_j indicates that, to first order, the objective function cannot be reduced by moving off the j-th constraint. Conversely, if the search direction is computed so that a constraint with a negative multiplier will be inactive at the next point, there must exist a scalar α such that $F(x_k + \alpha p_k) < F(x_k)$.

It is usually more efficient to avoid a complete minimization on a subspace and delete a constraint earlier when it seems appropriate. A way of investigating whether a constraint should be deleted at points other than at constrained stationary points is to compute *estimates* of the Lagrange multipliers. For example, the estimate λ_L defined by the least-squares problem

$$\min_{\lambda} \|\hat{A}^T\lambda - g_k\|_2$$

is frequently used. The reader is referred to Gill and Murray (1979) for a detailed discussion of Lagrange-multiplier estimates.

The apparent differences among algorithms of the reduced-gradient type arise from the various ways of representing Z, in order to generate search directions that remain in the proper subspace. For small dense problems, the best method for computing Z is based on the QR factorization of $\hat{A}^T$ (see Stewart, 1973). Let Q be an $n \times n$ orthogonal matrix such that:

$$Q\hat{A}^T = R = \begin{pmatrix} \overline{R} \\ 0 \end{pmatrix},$$

where $\overline{R}$ is a $t \times t$ nonsingular upper-triangular matrix. The last $n - t$ rows of Q can be taken as the columns of the matrix Z, since they are linearly independent and orthogonal to the rows of $\hat{A}$.

A basic feature of linearly constrained minimization is that $\hat{A}$ and Z can be *updated* since $\hat{A}$ changes by only a single row at each iteration. Consequently, an important consideration in the choice of Z should be the accuracy of the computed quantities after a long sequence of updates. The orthogonal factorization is by far the most accurate method for computing Z. It is the only choice

of Z that gives a constrained problem whose degree of difficulty is never worse than that of the original problem. For other methods for computing Z and a discussion of the relative merits of alternative techniques, see Gill and Murray (1974b).

If the linear constraints comprise only simple bounds, it is not necessary to compute a matrix Z since columns of the identity matrix naturally define a suitable basis for the null space of the constraints. In this case an active set strategy is equivalent to partitioning the variables into two sets: the set of *fixed* variables which are at their upper or lower bounds, and the set of *free* variables which are currently being optimized. An unconstrained minimization is performed with respect to the free variables. This unconstrained problem is altered occasionally if a free variable violates a bound or a fixed variable is allowed to become free. Clearly, algorithms for bound-constrained minimization are conceptually closer to algorithms for unconstrained minimization than linearly constrained minimization. However, since the number of free variables may be much smaller than n, the difficulty of the minimization may be significantly reduced. Intuitively, one would expect that it would be possible to construct a more efficient algorithm by providing more information about the region in which the solution is expected to lie.

2.3 Methods for nonlinear constraints.

The level of difficulty increases significantly when there are nonlinear constraints. One of the added complications is that, given a direction of search p_k and a point x_k that satisfies a nonlinear constraint, there may be no step length α_k such that $x_k + \alpha_k p_k$ also satisfies the constraint. Moreover, methods that attempt to follow the constraint boundary are no longer suitable. Suppose that $c_i(x)$ is a constraint that is almost satisfied exactly at x_k. The Taylor-series expansion for $c_i(x_k + p)$ gives

$$c_i(x_k + p) = c_i(x_k) + a_i(x_k)^T p + O(\|p\|^2),$$

where $a_i(x)$ denotes the gradient of the constraint function $c_i(x)$. As x_k approaches the solution, all changes in x lie in the null space of constraints and $a_i(x_k)^T p \approx 0$. Clearly a first-order expansion of the constraints does not accurately give $c_i(x_k + p)$ as a function of p.

At a solution $\overset{*}{x}$, suppose that t of the constraints are satisfied exactly and that these constraints are denoted by $\{\hat{c}_i(x)\}$. If $\hat{A}(x)$ denotes the matrix whose i-th row is $\hat{a}_i(x)^T$, there exist t non-negative scalars $\hat{\lambda}_i$ such that

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

(For the precise second-order Kuhn-Tucker conditions for optimality see Fiacco and McCormick, 1964; Powell, 1974.)

An important tool for the understanding and solution of constrained problems is the *Lagrangian function*

$$L(\overset{*}{x}, \hat{\lambda}) \equiv F(\overset{*}{x}) - \sum_{i=1}^{t} \hat{\lambda}_i c_i(\overset{*}{x}).$$

The point $\overset{*}{x}$ is a stationary point of the Lagrangian with optimal multipliers since

$$\nabla_x L(\overset{*}{x}, \hat{\lambda}) = g(\overset{*}{x}) - \hat{A}(\overset{*}{x})^T \hat{\lambda} = 0.$$

Unfortunately, $\overset{*}{x}$ is not usually a local minimum of the Lagrangian function (instead it may be a saddle point), but it can be shown that $\overset{*}{x}$ is the minimum of $L(x, \hat{\lambda})$ when x is restricted to lie in the linear subspace

$$\hat{A}(x - \overset{*}{x}) = 0. \tag{4}$$

To illustrate this, consider the problem

$$\begin{aligned} &\text{minimize} && F(x) = x_1 x_2^2 \\ &\text{subject to} && x_1^2 + x_2^2 - 2 = 0. \end{aligned}$$

This problem has the optimal Lagrange multiplier $\lambda \approx .816497$. Some lines of equal function value for the Lagrangian function are shown in Figure 1 together with the set of x defined by (4). Note that the Lagrangian function has a saddle point at $\overset{*}{x}$ and positive curvature along the linearized constraint.

Suppose that x_k is an estimate of $\overset{*}{x}$. The Taylor-series expansion for $c_i(x_k + p)$ gives

$$c_i(x_k + p) = c_i(x_k) + a_i(x_k)^T p + O(\|p\|^2).$$

This expansion can be used to impose the requirement that $x_k + p$ satisfies all the constraints to first order, i.e.

$$A(x_k)p \geq -c(x_k).$$

Since the optimal Lagrange multipliers are usually unknown, the Lagrangian function must be defined using estimates, $\bar{\lambda}$. This analysis leads to the following linearly constrained sub-problem

$$\begin{aligned} &\underset{x \in \mathbb{R}^n}{\text{minimize}} && L(x, \bar{\lambda}) \\ &\text{subject to} && A(x_k)(x - x_k) \geq -c(x_k). \end{aligned} \tag{5}$$

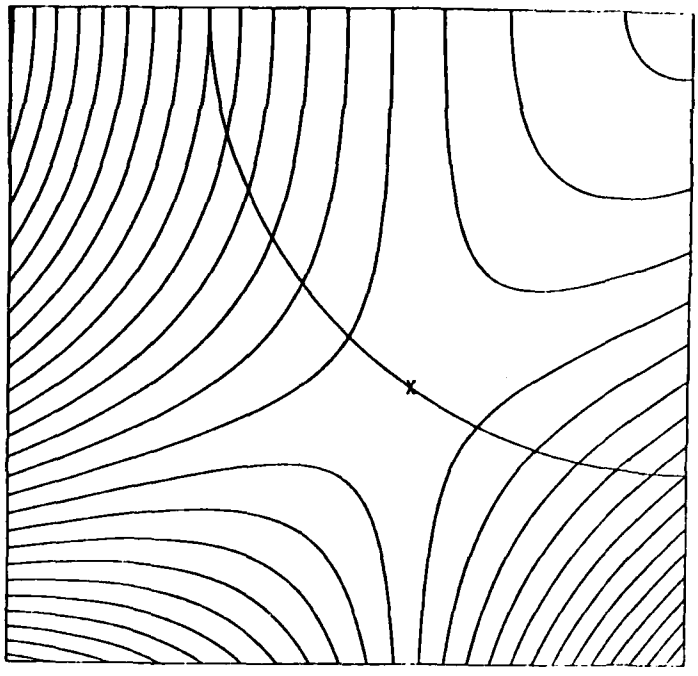

Figure 1.

It can be shown that any term of the form $\sigma^T A(x_k)v$ can be added to the Lagrangian function of (5) without altering the solution of the sub-problem. By adding the term $\bar{\lambda}^T \hat{A}(x_k)x$ to the Lagrangian function we ensure that, as x_k approaches $\overset{*}{x}$, the optimal Lagrange multipliers of the linearly constrained sub-problems are equal to those of the nonlinearly constrained problem. This convergence result also implies that the Lagrange multiplier vector from the previous linearly constrained sub-problem may be used for $\bar{\lambda}$. Note that the set of active constraints is implicitly defined by the solution of the sub-problem. For more theoretical details of this method see Robinson (1972).

Rather than solve a complete linearly constrained problem at each iteration we can form a quadratic approximation to the Lagrangian function and solve the resulting inequality quadratic program:

$$\begin{aligned} &\underset{p\in\mathbb{R}^n}{\text{minimize}} && \tfrac{1}{2}p^T B_k p + g_k^T p \\ &\text{subject to} && A(x_k)p \geq -c(x_k), \end{aligned}$$

where B_k now denotes an approximation to the Hessian of the Lagrangian function.

The conditions on p_k are derived by assuming that $\|p_k\|$ is small. To allow for large values of $\|p_k\|$ it might seem appropriate to find a scalar α such that

$\|c(x_k + \alpha p_k)\| < \|c(x_k)\|$. However, such a step may result in the objective function being increased from its previous value. The solution to this dilemma is to define a "merit" function which allows α to be adjusted so that the conflicting aims of achieving feasibility and reducing the objective function are balanced. Among the possible choices for the merit function are the following

$$P(x, \rho) = F(x) + \frac{\rho}{2} \sum_{i=1}^{t} c_i(x)^2.$$

$$\bar{P}(x, \rho) = F(x) + \rho \sum_{i=1}^{t} |c_i(x)|.$$

These functions add a penalty to the objective function that depends on the constraint violation. Note that $\bar{P}$ is not differentiable.

In our description we have ignored some of the difficulties associated with solving inequality quadratic programs or linearly constrained sub-problems. For example, the sub-problem may have no feasible point or an unbounded solution. Moreover, the quadratic program described here is only one of many alternatives that can be posed. An alternative strategy is to define an explicit set of active constraints and solve an *equality-constraint* quadratic program. This strategy has some advantages over the scheme described here. However, we have been concerned here with a unified description that will serve to introduce methods for large-scale problems. For a selection of QP-based techniques the reader is referred to Wilson (1963), Murray (1969), Biggs (1972), Garcia and Mangasarian (1976), Han (1976, 1977), Powell (1977) and Murray and Wright (1978). A discussion of the relative merits of the alternative approaches is given by Murray and Wright (1980).

2.4 Methods for non-differentiable functions.

Although non-differentiable problems are in general more difficult to solve, a distinction must be made between a problem with random discontinuities in functions or derivatives, and one in which a great deal of information is known about the nature of any discontinuities. In the former case, the only algorithms available are of the *polytope type* (see Nelder and Mead, 1962). In the latter case, algorithms can take advantage of the special structure.

In some well known instances, the problem functions themselves are not smooth, but rather are composites of smooth functions. For example, the following non-differentiable function occurs frequently, and is constructed in a particular way from the set of smooth functions $\{f_i\}$:

$$\underset{x \in \mathbb{R}}{\text{minimize}} \max\{f_1(x), f_2(x), \ldots, f_m(x)\}.$$

There has been much research concerning effective methods for these and related problems, and it is therefore advisable to use a specialized algorithm (see Wolfe, 1974; Murray and Overton, 1979). However, if such an algorithm is not available, this composite non-differentiable problem can be transformed into a smooth, but more complex, problem by introducing a new variable x_{n+1}, which is an upper bound on all the functions $\{f_i(x)\}$. Then the problem is given by

$$\begin{aligned} &\underset{x \in \mathbb{R}^{n+1}}{\text{minimize}} && x_{n+1} \\ &\text{subject to} && f_i(x) \leq x_{n+1}, \quad i = 1, \ldots, m. \end{aligned}$$

Note that the original unconstrained problem has been transformed into a nonlinearly constrained problem. In fact, all transformations of non-differentiable composite functions lead to a similar increase in complexity.

3. Methods for large sparse problems

Wherever possible, methods for large-scale optimization will be identical to those for the dense case, except that sparse matrix techniques will be utilized to minimize the storage and number of operations required. However, we shall often find that a method that is best for dense problems must be substantially altered if it is to be extended to the sparse case. In many cases this alteration compromises the theoretical features of the algorithm to the extent that it is no longer practical.

In many ways, methods for large-scale nonlinear optimization have similar properties to methods for linear equations. It is now generally accepted that the most stable methods for dense linear equations are not the most efficient for the sparse case. Unfortunately, this implies that any method for sparse problems may sometimes involve numerical processes that are not always as numerically stable as we would like. The same situation applies to algorithms for general nonlinearly constrained optimization.

3.1 Methods for sparse unconstrained minimization.

For unconstrained minimisation it is no longer clear that quasi-Newton methods are the most effective techniques. Toint (1979) derived a sparse quasi-Newton update of the form

$$B_{k+1} = B_k + U_k,$$

where U_k is a matrix of rank n that has the same sparsity pattern as B_k. In order to compute the correction U_k it is necessary to solve a system of equations with a coefficient matrix having the same structure as B_k. Unlike the dense case, it is not possible to guarantee that B_k will be positive definite, and consequently quasi-Newton methods are identical to modified-Newton methods in the

requirement for a related positive-definite matrix to define p_k. Note that the three advantages of dense quasi-Newton methods over finite-difference methods have been lost — the maintenance of positive definiteness, the ability to recur B_k in invertible form and the lower overhead per iteration. This reversal in performance is made more striking if finite-differences of the gradient vector are taken along certain linear combinations of the co-ordinate directions. This enables the approximate Hessian to be found in significantly fewer than $n+1$ gradient evaluations (see Curtis *et al.*, 1971; Gill and Murray, 1973; Powell and Toint, 1979)

If there are a large number of variables and the Hessian matrix is not sparse or structured, conjugate-gradient methods, instead of being the lowest ranked algorithms, become the *only* algorithms that can be applied with any chance of success. Problems whose Hessian matrices at the solution contain sets of clustered eigenvalues may be minimized in significantly fewer than n iterations. Problems without this property may require anything from between n and $5n$ iterations, with approximately $2n$ iterations or fewer being a common figure for moderately difficult problems. *Clearly, for a very large unconstrained problem without structure, of the order of several thousand variables, there is currently no algorithm that can be applied and be expected to work efficiently.*

3.2 Methods for sparse linear constraints.

In this section we shall discuss methods for large-scale linearly constrained minimization that exploit the sparsity in A or G.

When A is large and sparse it is not practical to update the orthogonal factorization of the matrix of active constraints as constraints enter and leave the basis. Instead, the problem is formulated in a slightly different way so that the extensive results on updating the sparse factorizations occurring in linear programming may be utilized.

Consider the problem

$$\begin{aligned} &\underset{x\in\mathbb{R}^n}{\text{minimize}} \quad F(x) \\ &\text{subject to} \quad Ax = b, \quad l_i \leq x_i \leq u_i. \end{aligned}$$

The matrix A is now $m \times n$ with $m < n$. If the original problem does not conform to this standard form it can be made to do so by the introduction of slack variables. In this event the resulting matrix A will have a special structure, involving columns that are columns of the identity matrix. This does not seriously increase the storage requirements since the matrix A will be stored in packed form.

A typical set of t active constraints must comprise the m rows of $Ax = b$ and $t - m$ simple bounds. It is not possible in the large sparse case to compute

the columns of Z explicitly. If we assume without loss of generality that the last $n - t$ variables are on their bounds, the active constraint matrix $\hat{A}$ for a typical iteration can be partitioned so that

$$\hat{A} = \begin{pmatrix} B & S & N \\ & & I \end{pmatrix},$$

where B is an $m \times m$ "basis" matrix. This partition effectively divides the variables into three classes — dependent, independent and temporarily fixed. The following matrix Z is orthogonal to the rows of $\hat{A}$:

$$Z = \begin{pmatrix} -B^{-1}S \\ I \\ 0 \end{pmatrix}.$$

In this case the columns of Z are not computed explicitly, since the quantities actually needed are products of the form Zp and Z^Tg, and these can be obtained by solving a system of equations involving B or B^T.

If the matrix Z^TGZ is very large at any trial solution, a conjugate-gradient method must be used to compute p_Z, since Z^TGZ will, in general, be a dense matrix – regardless of any possible sparsity of G and Z. Given the current inefficiency of conjugate-gradient methods on problems without clustered eigenvalues, this seriously limits our ability to solve general large-scale optimisation problems.

Fortunately, there is an important class of problems for which it is known a *priori* that the projected Hessian matrix can never be greater than a manageable size. These problems have a small number of variables that appear nonlinearly in the objective function, i.e. $F(x)$ is of the form

$$F(x) = f(\tilde{x}) + c^T x,$$

where $\tilde{x} = (x_1, x_2, \ldots, x_q)^T$, $x = (x_{q+1}, \ldots, x_n)^T$ and $f(\tilde{x})$ denotes any differentiable nonlinear function. In this case the dimension of the projected Hessian matrix at the solution is bounded by q. If q is small enough, a quasi-Newton approximation to G_Z can be updated in the usual way. For the complete details of this algorithm see Murtagh and Saunders (1978).

Any linearly constrained algorithm based on an active set strategy requires an initial feasible point x_0. This point is best found by linear programming (the typical "Phase 1" of the simplex method). Thus when minimising linear functions, an algorithm for large-scale linearly constrained optimisation should be competitive with large-scale linear programming codes.

3.3 Methods for sparse nonlinear constraints.

The results of the previous section lead us to write large-scale nonlinearly constrained problems in the form

$$\begin{aligned} \text{minimize} \quad & f(\tilde{x}) + c^T x \\ \text{subject to} \quad & c(\tilde{x}) + A_1 x = b_1 \\ & A_2 \tilde{x} + A_3 x = b_2 \\ & l_i \leq x_i \leq u_i. \end{aligned}$$

The linearly constrained sub-problem analogous to (5) for this problem is given by

$$\begin{aligned} \text{minimize } & f(\tilde{x}) + c^T x - \bar{\lambda}^T \big(c(\tilde{x}) - c(\tilde{x}_k) - \hat{A}(\tilde{x}_k)(\tilde{x} - \tilde{x}_k)\big) \\ \text{subject to } & c(\tilde{x}_k) + \hat{A}(\tilde{x}_k)(\tilde{x} - \tilde{x}_k) + A_1 x = b_1 \\ & A_2 \tilde{x} + A_3 x = b_2 \\ & l_i \leq x_i \leq u_i, \end{aligned}$$

where $\hat{A}$ denotes the matrix of gradients of $c(\tilde{x})$. The essential feature of this problem is that the dimension of the projected Hessian matrix of the Lagrangian function is bounded by q. This implies that each linearly constrained sub-problem can be solved using the techniques mentioned in the last section.

Murtagh and Saunders (1980) have described a technique based on solving a similar set of linearly constrained problems. They add a penalty term of the form

$$\frac{\rho}{2}\big(c(\tilde{x}) - c(\tilde{x}_k) - \hat{A}(\tilde{x}_k)(\tilde{x} - \tilde{x}_k)\big)^T \big(c(\tilde{x}) - c(\tilde{x}_k) - \hat{A}(\tilde{x}_k)(\tilde{x} - \tilde{x}_k)\big)$$

to the Lagrangian function so that the sub-problem is more likely to have a bounded solution.

As in the linear constraint case, we must rely on conjugate-gradient methods to solve the sub-problem if the projected Hessian matrix cannot be stored in the machine.

4. Numerical software

As the theoretical basis of algorithms becomes more complex and the scope of optimisation widens, it becomes more difficult for those actively engaged in practical problem solving to write their own software. Software libraries and software distribution centers partially solve this problem by making a variety of techniques immediately available to the non-specialist.

Those interested in obtaining software should be aware of the different ways in which mathematical software is disseminated. Often a new optimization method is published in a research journal and the method is accompanied by a short program implementing the new techniques discussed in the paper. Alternatively, a program will be obtainable from the author on application. Usually this program is written simply to test out the new theoretical developments and will only work on a few selected examples. In general, little attention is given to the standard of the coding or the documentation. A much better way of obtaining mathematical software is from a *software library.* This term does not mean simply a collection of programs from varied sources, written in isolation, that are grouped together in one place with some uniform documentation. Rather, a program library is a set of routines that are conceived and written within a unified framework, to be available to a general community of users.

At this time there are only three organizations that distribute library quality software for optimization:

The Numerical Algorithms Group (NAG) Library, Banbury Road, Oxford, England.

The NPL Numerical Optimization Software Library, The Division of Numerical Analysis and Computer Science, National Physical Laboratory, Teddington, Middlesex, England.

The MINPACK project, Applied Mathematics Division, Argonne National Laboratory, Illinois, USA.

Before acquiring a piece of software from a source other than a genuine software library, a prospective user should obtain answers to the following questions:

(i) *How quickly are errors in the code corrected?* If the routine is the result of some piece of research, the author may no longer be employed by the distributing organization or may no longer be concerned with its maintenance.

(ii) *Is the routine written in a high-level language that is available at the prospective users computer installation?* Even the most commonly used languages have dialects which are only available at certain installations.

(iii) *What changes will need to be made to the routine in order to run the software at the installation?* The most easily transported routines will be written in a *portable* high-level language such as ANSI-Fortran. It is inevitable that some changes will need to be made since certain constants, such as the relative machine precision, must be tailored to the host machine (for this reason, users should be wary of routines that require *no* alteration!).

(iv) *Is there adequate documentation?* We have shown that there are many different problem categories in optimisation. One feature of the documentation should be an adequate description of the problem category for which it is in-

tended. Also the documentation should include the results of the application of the routine to a simple, but non-trivial, example problem.

For other sources of optimization software see Gill and Murray (1974b, pp 242–243) and Nazareth (1978).

There are far fewer codes available for large-scale nonlinear optimization than codes for small problems. Several factors contribute to this situation. Historically, computational research on large-scale optimization has occurred in the commercial field, with numerical work on nonlinear problems being more the province of "academic" research. This has occurred partly because the academic community tends to measure the quality of research by the *quantity* of theoretical papers that are published in scientific journals. This has tended to discourage the considerable investment of time needed to complete a large-scale optimization code. (The emphasis on short theoretical results also goes some way to explain the large number of published papers on unconstrained optimization and nonlinear least-squares problems.)

As a result of this imbalance, methods for large-scale nonlinear optimization have tended to be developed as "add on" features of linear programming codes when a better approach may be to consider linear programming as a special case of nonlinear optimization.

The shortage of codes for large-scale nonlinear optimization is further aggravated by difficulties in disseminating these codes to the user community. Methods for large problems must include sophisticated input and output routines in order to handle the large quantities of data associated with the linear constraints. This requirement limits both the degree of portability that can be achieved and the size of the host machine.

In spite of the difficulties, some portable (or near portable) codes have been developed for large-scale constrained optimization. For linear programs, MINOS (Murtagh and Saunders, 1978) and XMP (Marsten, 1978) are appropriate. The system MINOS is also designed to handle a nonlinear objective function. It normally uses a quasi-Newton approximation to the projected Hessian Z^TGZ, in the form R^TR, where R is upper triangular. For problems where R would be extremely large, the software described by Marsten and Shanno (1979) may be preferable. However, at this time we cannot expect *any* general-purpose code to be efficient on problems in this class.

For problems with nonlinear constraints, the algorithm discussed in Section 3.3 has been implemented in the nonlinear programming system MINOS/AUGMENTED (Murtagh and Saunders, 1980a, b). The matrix of constraint gradients may be sparse, and there may be a large set of purely linear constraints. Both objective and constraint gradients must be computable. (This is usually no restriction with electrical power problems.)

5. Current research

It is a feature of optimal electrical power flow problems that the number of constraints active at the solution is unlikely to be large. This is because many constraints are present to prevent solutions that have no physical significance. A model would be considered to produce a "bad design" if a significant number of these constraints were satisfied at the solution. This feature presents a serious impediment to the routine solution of power flow problems by the methods discussed in Sections 3.2 and 3.3 since it is unlikely that the $(n-t) \times (n-t)$ set of equations

$$Z^T G Z p_Z = -Z^T g_k \tag{6}$$

can be solved in core. However, if the Hessian matrix and constraint gradients are sparse, matrix-vector products of the form $Z^T G Z v$ can be computed relatively cheaply by forming, in turn, $v_1 = Zv$, $v_2 = Gv_1$ and $v_3 = Z^T v_2$. This property can be utilized fully if p_Z is found using the *linear* conjugate-gradient method (see Hestenes and Stiefel, 1952). The linear conjugate-gradient algorithm is usually derived as a direct method, in the sense that, theoretically, the exact solution is found after $n-t$ iterations. However, in practice the algorithm behaves more like an iterative method since it has the potential of converging in fewer than, or more than, $n-t$ iterations.

Recently, Dembo (1979) has suggested a "truncated Newton method", in which the linear conjugate-gradient method is terminated even though a solution to (6) may not have been determined. The last iterate of the linear conjugate-gradient algorithm is then used as the direction of search. If a single linear iteration is used, it can be shown that p_Z will be the steepest-descent direction in the subspace spanned by Z. If a full $n-t$ iterations are used, p_Z will be the Newton direction defined by the solution of (6). Thus the truncated Newton algorithm computes a vector that interpolates between the steepest-descent direction and Newton direction.

The search direction defined by (6) is satisfactory only if $Z^T G Z$ is positive definite. An indefinite matrix $Z^T G Z$ allows the possibility that p_Z is not a descent direction and this may result in convergence to a non-optimal point.

An important feature of the class of modified Newton algorithms described in Section 2 is the ability to detect that $Z^T G Z$ is not sufficiently positive definite and to compute a satisfactory descent direction regardless. A straightforward application of a linear conjugate-gradient algorithm would not have this property. Moreover, the linear conjugate-gradient algorithm is numerically unstable when applied to an indefinite system.

Another unsatisfactory feature of the truncated Newton method is the use of the direction of steepest descent. It is well known that the method of steepest descent is very inefficient. Consequently, unless a large number of iterations of the linear conjugate-gradient method are used, the resulting search direction

may have more in common with the steepest-descent direction than the Newton direction.

Recent research has been concerned with deriving truncated Newton methods that do not require F to have a uniformly positive-definite projected Hessian. In addition, they generate a set of linear conjugate directions that are conjugate to a vector other than the steepest-descent direction. This latter feature gives a truncated Newton method in which p_Z interpolates the Newton direction and the direction obtained from a *nonlinear* conjugate-gradient algorithm.

References

Biggs, M.C. and Laughton M.A.(1977). Optimal Electric Power Scheduling: a Large Nonlinear Test Problem Solved by Recursive Quadratic Programming, *Math. Prog.*, 13, pp. 167–182.

Brodlie, K. W. (1978). Unconstrained optimization, in: *Modern Numerical Analysis* (D.J. Jacobs, ed.), pp. 229–268, Academic Press, London and New York.

Curtis, A. R., Powell, M. J. D. and Reid, J. K. (1974). On the estimation of sparse Jacobian matrices, *J. Inst. Maths Applics.*, 13, pp. 117–119.

Dembo, R. S.(1979). Recent advances in geometric programming, presented at the TIMS XXIV International Meeting, Honolulu, June 1979.

Dennis, J. E. and Moré J. J. (1977). Quasi-Newton methods, motivation and theory, *SIAM Review*, 19, pp. 46–89.

Fletcher, R. and Reeves, C. M. (1964). Function minimization by conjugate gradients, *Comput. J.*, 7, pp. 149–154.

Garcia-Palomares, U. M. and Mangasarian, O. L. (1976). Superlinearly convergent quasi-Newton algorithms for nonlinearly constrained optimization problems, *Math. Prog.* 11, pp. 1–13.

Gill, P. E. and Murray, W. (1972a). Quasi-Newton methods for unconstrained optimisation, *J. Inst. Maths Applics.*, 9, pp. 91–108.

Gill, P. E. and Murray, W. (1973). The numerical solution of a problem in the calculus of variations, in: *Recent Mathematical Developments in Control* (D. J. Bell, ed.), pp. 97–122, Academic Press, London and New York.

Gill, P. E. and Murray, W. (1974a). Newton-type methods for unconstrained and linearly constrained optimisation, *Math. Prog.* 28, pp. 311–350.

Gill, P. E. and Murray, W. (eds.) (1974b). *Numerical Methods for Constrained Optimisation*, Academic Press, London and New York.

Gill, P. E. and Murray, W. (1974c). "Newton-type methods for linearly constrained optimization", in *Numerical Methods for Constrained Optimisation* (P. E. Gill and W. Murray, eds.), pp. 29–66, Academic Press, London and New York.

Gill, P. E. and Murray, W. (1976). Minimization subject to bounds on the variables. Report NAC 71, National Physical Laboratory, England.

Gill, P. E. and Murray, W. (1978). The design and implementation of software for unconstrained optimization, in: *Design and Implementation of Optimization Software* (H. Greenberg, ed.), Sijthoff and Noordhoff.

Gill, P. E. and Murray, W. (1979). The computation of Lagrange multiplier estimates for constrained minimization, *Math. Prog.* 17, pp. 32–60.

Gill, P. E. and Murray, W. (1979). Conjugate-gradient methods for large-scale nonlinear optimization, Report SOL 79-15, Department of Operations Research, Stanford University.

Gill, P. E., Murray, W., Picken, S. M. and Wright, M. H. (1979). The design and structure of a Fortran program library for optimization, *ACM Trans. on Mathematical Software* 5, pp. 259–283.

Gill, P. E., Murray, W., Saunders, M. A. and Wright, M. H. (1979). Two step-length algorithms for numerical optimization, Report SOL 79-25, Department of Operations Research, Stanford University.

Han, S.-P. (1976). Superlinearly convergent variable metric algorithms for general nonlinear programming problems, *Math. Prog.* 11, pp. 263–282.

Han, S.-P. (1977). A globally convergent method for nonlinear programming, *J. Opt. Th. Appl.* 22, pp. 297–310.

Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Standards*, 49, pp. 409–436.

Marsten R. E. (1978). XMP: A structured library of subroutines for experimental mathematical programming, Report 351, Department of Management Information Systems, University of Arizona, Tucson, Arizona.

Marsten, R. E. and Shanno, D. F. (1979). Conjugate gradient methods for linearly constrained nonlinear programming, Report 79-13, Department of Management Information Systems, University of Arizona, Tucson, Arizona.

Moré J. J. and Sorenson D. C. (1979). On the use of directions of negative curvature in a modified Newton method, *Math. Prog.* 15, pp. 1–20.

Murray, W. (1969). "An algorithm for constrained minimization", in *Optimization* (R. Fletcher, ed.), pp. 247–258, Academic Press, London and New York.

Murray, W. and Overton, M. L. (1979). A projected Lagrangian algorithm for nonlinear minimax optimization, Report SOL 79–21, Department of Operations Research, Stanford University.

Murray, W. and Wright, M. H. (1978). Projected Lagrangian methods based on the trajectories of penalty and barrier functions, Report SOL 78-23, Department of Operations Research, Stanford University.

Murray, W. and Wright, M. H. (1980). Computation of the search direction in constrained optimization algorithms, Report SOL 80-2, Department of Operations Research, Stanford University.

Murtagh, B. A. and Saunders, M. A. (1978). Large scale linearly constrained optimization, *Math. Prog.* 14, pp. 41–22.

Murtagh, B. A. and Saunders, M. A. (1980a). The implementation of a Lagrangian-based algorithm for sparse nonlinear constraints, Report SOL 80-1, Department of Operations Research, Stanford University.

Murtagh, B. A. and Saunders, M. A. (1980b). MINOS/AUGMENTED User's Manual, SOL Report (to appear), Department of Operations Research, Stanford University.

Nazareth, L. and Nocedal, J. (1978). A study of conjugate gradient methods, Report SOL 78-29, Department of Operations Research, Stanford University.

Nelder, J.A. and Mead, R. (1965). A simplex method for function minimization, *Computer Journal* 7, pp. 308–313.

Nocedal, J. (1979). Updating quasi-Newton matrices with limited storage, IIMAS, Universidad Nacional Autonoma de Mexico (manuscript).

Ortega, J. M. and Rheinbolt, W. C. (1970). *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, London and New York.

Powell, M. J. D. (1969). "A method for nonlinear constraints in minimization problems", in *Optimisation* (R. Fletcher, ed.), pp. 283–298, Academic Press, London and New York.

Powell, M. J. D. (1974). "Introduction to constrained optimisation", in *Numerical Methods for Constrained Optimization* (P. E. Gill and W. Murray, eds.), pp. 1–28, Academic Press, London and New York.

Powell, M. J. D. (1977). A fast algorithm for nonlinearly constrained optimization calculations, Report DAMTP 77/NA 2, University of Cambridge, England.

Powell, M. J. D. (1978). "The convergence of variable metric methods for nonlinearly constrained optimization calculations", in *Nonlinear Programming 3* (O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds.), pp. 27–63, Academic Press, London and New York.

Robinson, S. M. (1972). A quadratically convergent algorithm for general nonlinear programming problems, *Math. Prog.* 3, pp. 145–156.

Rosen, J. B. and Kreuser, J. (1972). "A gradient projection algorithm for nonlinear constraints", in *Numerical Methods for Non-Linear Optimization* (F. A. Lootsma, ed.), pp. 297–300, Academic Press, London and New York.

Rosen, J. B. (1978). "Two-phase algorithm for nonlinear constraint problems", in *Nonlinear Programming 3* (O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, eds.), pp. 97–124, Academic Press, London and New York.

Shanno, D. F. (1978a). Conjugate gradient methods with inexact searches, *Math. of Oper. Res.* 3, pp. 244–256.

Stewart, G. W. (1973). *Introduction to Matrix Computations*, Academic Press, London and New York.

Toint, Ph. L. (1977). On sparse and symmetric matrix updating subject to a linear equation, *Math. Comp.* 31, pp. 954–961.

Wilson, R. B. (1963). A Simplicial Algorithm for Concave Programming, Ph. D. Thesis, Harvard University.

Wolfe, Ph. (1974). A note on the method of conjugate gradients for minimising nondifferentiable functions, *Math. Prog.* 7, pp. 380–383.

(18) (11)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (14) SOL-80-8	2. GOVT ACCESSION NO. AD-A087	3. RECIPIENT'S CATALOG NUMBER 555
4. TITLE (and Subtitle) METHODS FOR LARGE-SCALE NONLINEAR OPTIMIZATION,		5. TYPE OF REPORT & PERIOD COVERED (9) TECHNICAL REPORT,
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) (7) Philip E. Gill, Walter Murray, Michael A. Saunders, Margaret H. Wright		8. CONTRACT OR GRANT NUMBER(s) (15) DAAG29-79-C-0110, N00014-75-C-0267
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Operations Research - SOL Stanford University Stanford, CA 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (12)
11. CONTROLLING OFFICE NAME AND ADDRESS (11) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709		12. REPORT DATE May 80
		13. NUMBER OF PAGES 21
Office of Naval Research - Dept. of the Navy 800 N. Quincy Street Arlington, VA 22217		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

THE VIEW, OPINIONS, AND/OR FINDINGS CONTAINED IN THIS REPORT ARE THOSE OF THE AUTHOR(S) AND SHOULD NOT BE CONSTRUED AS AN OFFICIAL DEPARTMENT OF THE ARMY POSITION, POLICY, OR DECISION, UNLESS SO DESIGNATED BY OTHER DOCUMENTATION.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

large-scale optimization
unconstrained optimization
linearly constrained optimization
nonlinearly constrained optimization
Lagrangian methods
nondifferentiable problems
sparse Hessian
sparse linear and nonlinear constraints
numerical software

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The application of optimization to electrical power technology often requires the numerical solution of systems that are very large and possibly non-differentiable. A brief survey of the state of the art of numerical optimization is presented in which those methods that are directly applicable to power system problems will be highlighted. The areas of current research that are most likely to yield direct benefit to practical computation are identified. The paper concludes with a survey of available software.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE S/N 0102-014-6601

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)