JOURNAL OF OPTIMIZATION THEORY AND APPLICATIONS: Vol. 16, Nos. 1/2, 1975

An Improved Penalty Function Method for Solving Constrained Parameter Optimization Problems[1,2]

J. T. BETTS[3]

Communicated by H. Y. Huang

Abstract. An effective algorithm is described for solving the general constrained parameter optimization problem. The method is quasi-second-order and requires only function and gradient information. An exterior point penalty function method is used to transform the constrained problem into a sequence of unconstrained problems. The penalty weight r is chosen as a function of the point x such that the sequence of optimization problems is computationally easy. A rank-one optimization algorithm is developed that takes advantage of the special properties of the augmented performance index. The optimization algorithm accounts for the usual difficulties associated with discontinuous second derivatives of the augmented index. Finite convergence is exhibited for a quadratic performance index with linear constraints; accelerated convergence is demonstrated for nonquadratic indices and nonlinear constraints. A computer program has been written to implement the algorithm and its performance is illustrated in fourteen test problems.

Key Words. Penalty-function methods, mathematical programming, nonlinear programming, pseudo Newton–Raphson methods, parameter optimization.

---

[1] This paper was prepared under AF Contract No. F04701-72-C-0073.

[2] The author expresses his appreciation to Dr. H. E. Pickett for his continuing support in the theoretical and practical development of the BEST computer program. In addition, the recursive method for updating the Hessian of the penalty function was developed jointly with Drs. Pickett and J. L. Searcy and is included here with their permission. Finally, the author would like to acknowledge the contribution made by the stimulating environment of an optimal control seminar held at The Aerospace Corporation since 1970. Principal members of the seminar have been Drs. Pickett, Searcy, R. W. Reid, and the author.

[3] Member of the Technical Staff, Guidance and Control Division, The Aerospace Corporation, El Segundo, California.

1. Introduction

Fiacco and McCormick (Ref. 1) have developed the penalty-function method for solving constrained parameter optimization problems. In this approach, a penalty term reflecting the constraint violations multiplied by a scalar weight is augmented to the actual performance index. If the augmented performance index is minimized for a sequence of increasing penalty weights, the solutions of the successive unconstrained problems approach the constrained solution.

While the algorithm appears straightforward, the basic sequential unconstrained minimization technique has three undesirable properties. First, one must be able to solve each of the unconstrained parameter optimization problems posed by the algorithm. Unfortunately, for large values of the penalty weight, the derivatives of the augmented index become large, and the resulting unconstrained performance index is difficult to optimize. Second, the second partial derivatives of the augmented index are discontinuous; consequently, any of the accelerated optimization techniques that attempt to estimate second-order information recursively do not perform well. Third, no criterion is given for choosing the sequence of penalty weights.

Nevertheless, the penalty function method is useful in the fields of regression analysis and is especially valuable for solving optimal control problems using the Balakrishnan epsilon method (Refs. 2–5). Therefore, improving the basic approach seems worthwhile.

Powell has proposed a method (Ref. 6) which deals with the first undesirable property, namely the numerical problems associated with large values of the penalty weight. Miele, Coggins, and Levy (Ref. 7) suggest an approach for choosing the sequence of penalty weights.

A method is presented in this paper for accelerating the convergence of the penalty function method and eliminating the usual difficulties with large penalty weights. The method establishes a criterion for choosing the sequence of penalty weights so that successive optimization problems are computationally easy. A rank-one recursive unconstrained optimization method is also proposed. The optimization algorithm is unique in that it overcomes the usual difficulties associated with discontinuous second derivatives. The approach is also applicable to singular or nearly singular problems, since the search steps are determined using a least-squares process, rather than direct matrix inversion.

2. Background

The problem of interest in this paper is to find the n-vector x that minimizes the scalar function

$$f(x) = f(x_1,..., x_n), \tag{1}$$

called the performance index, subject to the equality constraints

$$c_i(x) = 0, \qquad i = 1,..., m_1, \tag{2}$$

and the inequality constraints

$$c_i(x) \geqslant 0, \qquad i = m_1 + 1,..., m. \tag{3}$$

The functions $f(x)$ and $c_i(x)$ are assumed continuously differentiable to second order.

A principal approach for finding a solution to the nonlinear programming problem is to solve a sequence of unconstrained problems whose solutions approach the constrained solution in the limit. In particular, consider the augmented performance index

$$J(x, r) = f(x) + rP(x), \tag{4}$$

where r is a scalar, referred to as the penalty weight, and

$$P(x) = \sum_{i=1}^{m} I[c_i(x)]\, c_i^2(x), \tag{5}$$

with

$$I[c_i(x)] = \begin{cases} 1, & 1 \leqslant i \leqslant m_1, \\ 1, & m_1 < i \leqslant m \quad \text{and} \quad c_i(x) < 0, \\ 0, & m_1 < i \leqslant m \quad \text{and} \quad c_i(x) \geqslant 0. \end{cases} \tag{6}$$

Beginning with some estimate of the optimum point x^0 and with some $r_0 \geqslant 0$, we can minimize the function $J(x, r_0)$ to obtain $x^*(r_0)$. If we use $x^*(r_0)$ as an initial estimate, the function $J(x, r_1)$ is minimized for some $r_1 > r_0$. Fiacco and McCormick show that, if this process is repeated for a strictly increasing unbounded nonnegative sequence $\{r_k\}$, the solutions $x^*(r_k)$ approach the constrained solution x^*.

3. Easy-r Determination

First, let us consider how the penalty weight r can be chosen. A necessary condition for the existence of an optimum x^* is that the

gradient at x^* be zero, i.e., $\nabla J(x^*) = 0$. Therefore, if x is some arbitrary point, it is reasonable to choose the scalar r to minimize

$$e = \| \nabla J \|^2 = [\nabla f(x) + r\nabla P(x)]^T [\nabla f(x) + r\nabla P(x)], \tag{7}$$

subject to the condition

$$r \geqslant 0. \tag{8}$$

If we set $\partial e/\partial r = 0$, the value of r which minimizes Eq. (7) subject to to Eq. (8) is

$$r_e(x) = \begin{cases} \max \left[0, \dfrac{-\nabla f(x)^T \nabla P(x)}{\nabla P(x)^T \nabla P(x)}\right], & \nabla P(x) \neq 0, \\ \infty, & \nabla P(x) = 0. \end{cases} \tag{9}$$

The subscript e has been added, since this value of r is referred to as the easy-r. Since r_e minimizes the norm of the gradient evaluated at x, heuristically the problem should be easy to solve.

The situation can be illustrated for a simple problem. Suppose that we must choose $x = (x_1, x_2)$ to minimize

$$f(x) = x_1^2 + x_2^2,$$

subject to the linear inequality constraint

$$c_1(x_1, x_2) = x_1 + x_2 - 1 \geqslant 0.$$

The performance index $f(x_1, x_2)$ has an unconstrained minimum at the origin, and contours of equal value are concentric circles centered about the origin (see Fig. 1). The constraint c_1 divides the x_1x_2-space into a feasible region (shaded) and an infeasible region. Furthermore, the value of the easy-r, r_e, is positive in the region between the constraint c_1 and the dashed line parallel to c_1. To the left of the dashed line, $r_e = 0$; in the feasible region, r_e is defined to be infinity, since $\nabla P = 0$.

The constrained optimum is $x^* = (1/2, 1/2)$. The optimal solution for any fixed value of r lies on the line between the origin and x^*; in particular, $x^*(0)$, the solution for $r = 0$, is the unconstrained solution point, namely the origin. As the value of r_k increases, the points $x^*(r_k)$, which minimize J, approach the solution point x^*, as illustrated.

When r is set equal to r_e, the gradient ∇J is orthogonal to ∇P. Consequently, the direction of search is locally parallel to the constraint surface $c_1(x) = 0$. In fact, if we plot contours of r_e as a function of x, the contours for this problem are straight lines parallel to c_1. The line through the origin corresponds to the contour $r_e(x) = 0$, while c_1 corresponds to the contour $r_e(x) = \infty$.

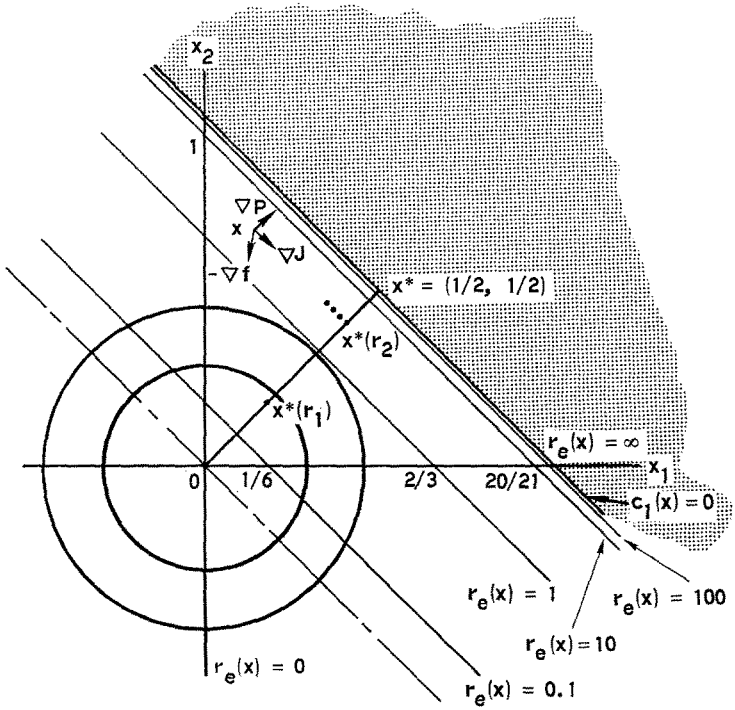

Fig. 1. Sample problem.

To recapitulate, a criterion has been established for defining a value for the parameter r, depending on the point x and the value of the gradient at x. The penalty extrapolation process described next defines an approach for judiciously choosing the estimates of x^*.

4. Penalty Function Extrapolation

Fiacco and McCormick demonstrate that there is a trajectory of optimal points varying continuously as a function of r. They suggest a Taylor series approximation about the solution point, given here as

$$\hat{x}^*(r) = \sum_{i=0}^{N} a(i)\, r^{-i}, \tag{10}$$

where $a(i) = \text{col}[a_1(i),..., a_n(i)]$. If $N + 1$ optimization problems have been solved, then $N + 1$ solution vectors have been obtained, namely

$x^*(r_0)$, $x^*(r_1)$,..., $x^*(r_N)$. Let us insist that the solution vectors $x^*(r_k)$ equal the approximation [Eq. (10)] at the points r_k, or

$$x^*(r_k) = \sum_{i=0}^{N} a(i)\, r_k^{-i}, \qquad k = 0, 1, \ldots, N. \tag{11}$$

The system of $N + 1$ equations is linear in the coefficients $a(i)$, and the matrix of r values is nonsingular, since it is a Vandermonde matrix. Consequently, the system [Eq. (11)] can be solved for the coefficients $a(i)$, completely defining the approximation [Eq. (10)].

Let us now examine the behavior of the approximation [Eq. (10)] as r becomes large. Clearly, we can see that

$$\lim_{r\to\infty} \hat{x}^*(r) = a(0). \tag{12}$$

Thus, a reasonable estimate for the constrained solution point x^*, based on the extrapolation of $N + 1$ unconstrained solutions $x^*(r_0)$,..., $x^*(r_N)$, is simply

$$x^* = a(0) = \hat{x}(\infty). \tag{13}$$

Fiacco and McCormick also observe that each additional solution point $x^*(r_k)$ improves the approximation [Eq. (10)].

5. Easy-r Algorithm

The easy-r algorithm essentially combines the penalty extrapolation procedure with the easy-r determination criterion. The basis of the easy-r algorithm is the extrapolation of a sequence of solution points $x^*(r_0) \cdots x^*(r_k)$ to obtain an asymptotic estimate $a(0)$. The easy-r corresponding to the asymptote is then computed; and, after checking the convergence of the approximation, the next optimization problem is started with the asymptote as an initial guess using the corresponding r. In essence, then, the extrapolation process determines the initial point for the $(k + 1)$th optimization problem, $x^0(r_{k+1})$. The value of r_{k+1} is determined using the easy-r criterion.

Several special considerations must be pointed out. First, if the easy-r is not larger than some minimum acceptable increment, it is not used. In particular,

$$r_{k+1} = \begin{cases} r_e[\hat{x}(\infty)] & \text{if } r_e[\hat{x}(\infty)] > \kappa \cdot r_k, \\ \kappa \cdot r_k & \text{if } r_e[\hat{x}(\infty)] \leqslant \kappa \cdot r_k, \end{cases} \tag{14}$$

where κ is a constant greater than one. This precaution ensures that the sequence of r_k is monotonically increasing; it is necessary when the points

$x^*(r_k)$ are far from the constrained solution x^*. Second, the violated constraint set must be the same over all points used in the extrapolation formula [Eq. (10)]. If the constraint sets differ, the extrapolation is restarted at the latest solution point. This procedure is necessary because, when the constraint set changes, the derivatives in the Taylor series expansion [Eq. (10)] become discontinuous, and the approximation cannot be used. Therefore, when the function is evaluated at the asymptote, the penalty term is evaluated with the same set of violated constraints as those at the other points in the sequence. Third, the asymptotic estimate may actually be in the feasible region, which is inconsistent with the behavior of an exterior point algorithm. In this case, a new estimate of the asymptote must be made. For example, an average of the old asymptote and the latest solution point $x^*(r_k)$ may be used.

From a theoretical standpoint, the choice of r_0 is arbitrary; from a practical standpoint, it is not. The mode of operation preset in the computer program is to first scale the problem and to then choose $r_0 = 1$. This choice gives equal weight to both f and P and has worked in all cases tested to date. The scaling process used is outlined in Appendix A. A second possibility that has been investigated is to choose r_0 using the easy-r criterion. Based on the test results, this approach was generally less effective.

6. Preliminary Developments

In the preceding sections, a criterion was established for choosing the parameter r. The sections that follow develop an optimization technique which is independent of r, is effective for singular and/or discontinuous Hessian matrices, and requires only function and gradient information.

Let us begin the development by writing the explicit expressions for the first and second derivatives. From Eqs. (4) and (5), we obtain

$$\nabla J = \nabla f + r\nabla P = \nabla f + r \sum_{i=1}^{m} 2I[c_i(x)]\, c_i(x)\, \nabla c_i(x). \tag{15}$$

The Hessian matrix is given by

$$\begin{aligned} \nabla^2 J &= \nabla^2 f + r\nabla^2 P \\ &= \nabla^2 f + r\left[\sum_{i=1}^{m} 2I[c_i(x)]\, c_i(x)\, \nabla^2 c_i(x) + \sum_{i=1}^{m} 2I[c_i(x)]\, \nabla c_i(x)\, \nabla c_i(x)^T\right]. \end{aligned} \tag{16}$$

The Hessian matrices $\nabla^2 J$, $\nabla^2 f$, $\nabla^2 P$, and $\nabla^2 c_i$ are all symmetric because of the continuity assumptions.

It is convenient to define the following matrices:

$$H = \nabla^2 J, \qquad T = \nabla^2 f,$$
$$U = \sum_{i=1}^{m} 2I[c_i(x)]\, c_i(x)\, \nabla^2 c_i(x), \qquad V = \sum_{i=1}^{m} 2I[c_i(x)]\, \nabla c_i(x)\, \nabla c_i(x)^T. \tag{17}$$

In terms of these matrices, the expression for the Hessian matrix of J can be written as

$$H = T + r[U + V]. \tag{18}$$

7. Determination of Search Step

Consider the Taylor series expansion of J about some arbitrary point x^k

$$J(x) = J(x^k) + \nabla J(x^k)^T (x - x^k) + (1/2)(x - x^k)^T \nabla^2 J(x^k)(x - x^k) + \cdots. \tag{19}$$

Neglecting terms of order higher than second, we obtain a quadratic approximation to the actual function. The gradient of this approximate function is

$$\nabla J(x) = \nabla J(x^k) + H^k(x - x^k), \tag{20}$$

where H^k denotes the Hessian matrix $\nabla^2 J(x^k)$, evaluated at the point x^k.

Suppose that an optimization iteration is to begin at the point x^k and proceed in a search direction defined by the vector s^k, to the point x^{k+1}, where

$$x^{k+1} = x^k - \rho^k s^k. \tag{21}$$

The positive scalar parameter ρ^k can be used to modify the magnitude of the step s^k. Let us temporarily assume $\rho^k = 1$ and examine how s^k can be determined such that x^{k+1} is, in some sense, a better estimate of the minimum point. One reasonable criterion would be to choose s^k such that at x^{k+1} the gradient is as small as possible. Formally, let us define s^k to be the vector that minimizes

$$\| \nabla J(x^{k+1}) \| = \| \nabla J(x^k) - H^k s^k \|, \tag{22}$$

where Eqs. (20) and (21) have been used. When H^k is singular, we impose the additional condition that

$$\| s^k \| \tag{23}$$

be a minimum, in order to define a unique solution. The problem of minimizing Eq. (22) is a standard numerical problem. A number of routines for solving it are presented in Ref. 8, one of which is used in the computer program.

Several important points should be noted about this process. First, if the matrix H^k can be inverted, then the search step is simply $s^k = (H^k)^{-1} \nabla J(x^k)$. This is the Newton–Raphson search step. One defect of the Newton–Raphson approach, however, is that H^k may not be invertible. In contrast, the least-square method of determining s^k works even if H^k is singular. While a truly singular problem may be poorly posed to begin with, nearly singular (or numerically singular) problems are common.

Second, if H^k is not positive definite, the search direction defined by s^k may not be a downhill direction. To ensure that each search direction is downhill, we multiply the value of s^k obtained from the least-squares process (for example, $\tilde{s}^k$) by the appropriate sign. Thus, the actual search step used in Eq. (21) is

$$s^k = \text{sign}[\nabla J(x^k)^T \tilde{s}^k] \tilde{s}^k. \tag{24}$$

Third, the determination of the search direction using a least-square procedure may be more time consuming than working with an estimate of the inverse (or pseudo-inverse) Hessian matrix directly. However, for many applications, e.g., trajectory optimization problems, the computational effort of the optimization algorithm is insignificant with respect to the computational effort involved in evaluating the function and gradient.

In summary, the basic optimization iteration begins at the point x^k and estimates a point x^{k+1}. The search step s^k is determined such that the gradient of the quadratic approximation to the performance index is as small as possible at the new point x^{k+1}. The information required for the computation of s^k includes the gradient vector and the Hessian matrix, both evaluated at x^k, specifically, $\nabla J(x^k)$ and H^k. When the performance index J is quadratic, the value of ρ^k can be set to one; however, for more general functions, modification of ρ^k may be desirable. Within the computer program, ρ^k is computed using a cubic search technique until one improving step is made, i.e., until $J(x^{k+1}) < J(x^k)$.

8. Construction of *T*-Matrix

The method for determining the search step vector s, described in Section 7, requires an estimate of the Hessian matrix H. Referring to

Eq. (18), we notice that knowledge of the matrices T, U, and V provides this information. The construction of T is outlined in this section; U and V are treated in Section 9. The basic requirement is that only function and gradient information may be used.

Let us first state a rank-one recursion formula. Suppose that Q^k is the value of some square, symmetric matrix at the kth iteration. Further, let us presume that, at every iteration, the Q-matrix must satisfy the equation

$$y^j = Q^k p^j, \qquad 1 \leqslant j \leqslant k, \tag{25}$$

where y^j and p^j are known vectors. If Q^{k-1} is known and $j = k$, then

$$Q^k = Q^{k-1} + (y^k - Q^{k-1}p^k)(y^k - Q^{k-1}p^k)^T/(p^k)^T\,(y^k - Q^{k-1}p^k). \tag{26}$$

This iterative process has been reported by a number of authors including Davidon (Ref. 9), Broyden (Ref. 10), and Murtagh and Sargent (Ref. 11). It is referred to as a rank-one method, since at every iteration a matrix of rank one is used as a correction to Q^{k-1}. Murtagh and Sargent demonstrate that, if the matrix Q is actually constant and Q^k represents the kth estimate of it, then the nth estimate will be exact. Equivalently, we could evaluate the vectors y^k and p^k n different times and then solve the resulting linear system for Q by direct inversion. Either the direct or the recursive approach fails if the set of p^k vectors are not linearly independent. To ensure that the iterative process does not break down when this occurs, we must verify that the scalar product in the denominator of Eq. (26) is nonzero.

The performance index $f(x)$ is assumed to be continuous and differentiable in x. Furthermore, it seems reasonable to approximate f by a quadratic function, as has been done for J [Eq. (19)]. Consequently, we can write an expression, analogous to Eq. (20), for the function f

$$\nabla f(x^{k+1}) = \nabla f(x^k) + T^k(x^{k+1} - x^k). \tag{27}$$

If we define

$$y^k = \nabla f(x^{k+1}) - \nabla f(x^k) \tag{28}$$

and

$$p^k = x^{k+1} - x^k, \tag{29}$$

then Eq. (27) can be rewritten as

$$y^k = T^k p^k. \tag{30}$$

Comparing Eq. (30) with Eq. (25), we realize that Eq. (26) becomes

$$T^k = T^{k-1} + (y^k - T^{k-1}p^k)(y^k - T^{k-1}p^k)^T/(p^k)^T\,(y^k - T^{k-1}p^k). \tag{31}$$

Clearly, the estimate of T can be updated using only the gradient differences [Eq. (28)] and the step [Eq. (29)]. The iterative process can be initiated with any symmetric matrix. The identity matrix is a reasonable choice.

9. Construction of U and V Matrices

The construction of the V-matrix follows directly from the defining equation (17). It contains only the gradients of the constraint functions and, consequently, does not necessitate explicit evaluation of second derivatives. V changes discontinuously whenever the constraint set changes, because of the presence of the step function $I[c_i(x)]$.

In contrast, the U matrix changes continuously when the constraint set changes, because of the presence of the product $I[c_i(x)]\, c_i(x)$. Clearly, this product approaches zero from either side of a constraint boundary. Since it is continuous, it seems reasonable to apply a recursive formula to U. For linear constraints, we find that $\nabla^2 c_i = 0$, in which case $U \equiv 0$. Furthermore, as the solution is approached, $c_i(x) \to 0$, and again $U \to 0$. In the classical Gauss method, the contribution of U is neglected, and $\nabla^2 P = V$. The Gauss approximation is quite good if the constraints are linear or nearly zero. For nonlinear constraints and points where the $c_i(x)$ are large, the contribution of U becomes significant.

Consider the boundary point b at which the set of violated constraints changes:

$$b = x^k + \alpha(x^{k+1} - x^k) = x^k + \alpha p^k, \tag{32}$$

where p^k is defined in Eq. (29) and α is a scalar, $0 \leqslant \alpha \leqslant 1$. It is assumed that the set of constraints changes, at most, once between x^{k+1} and x^k. If the constraint set does not change, by definition, set $\alpha = 1$.

Let P^+ be the function consisting of the sum of squares of constraints violated on the x^{k+1} side of the boundary point b. Representing the function P^+ by a Taylor series expansion about b, we obtain

$$P^+(x^{k+1}) = P^+(b) + \nabla P^+(b)^T (x^{k+1} - b) + 1/2(x^{k+1} - b)^T \nabla^2 P^+(b)(x^{k+1} - b). \tag{33}$$

A similar expansion for the function P^- consisting of the sum of squares of constraints violated on the other side of the boundary gives

$$P^-(x^k) = P^-(b) + \nabla P^-(b)^T (x^k - b) + 1/2(x^k - b)^T \nabla^2 P^-(b)(x^k - b). \tag{34}$$

Differentiating Eqs. (33) and (34) yields

$$\nabla P^+(x^{k+1}) = \nabla P^+(b) + \nabla^2 P^+(b)(x^{k+1} - b), \tag{35}$$

$$\nabla P^-(x^k) = \nabla P^-(b) + \nabla^2 P^-(b)(x^k - b). \tag{36}$$

But the derivatives of P are continuous across b, so let us set $\nabla P^+(b) = \nabla P^-(b) = \nabla P(b)$ and subtract Eq. (36) from Eq. (35) to give

$$\nabla P^+(x^{k+1}) - \nabla P^-(x^k) = \nabla^2 P^+(b)(x^{k+1} - b) - \nabla^2 P^-(b)(x^k - b). \tag{37}$$

From the definition of $\nabla^2 P$ and the fact that $U^+(b) = U^-(b) = U(b)$, it follows that

$$\begin{aligned} \nabla^2 P^+(b) &= U(b) + V^+(b) = U + V^+, \\ \nabla^2 P^-(b) &= U(b) + V^-(b) = U + V^-. \end{aligned} \tag{38}$$

Using the notation from Eq. (38), and observing that $\nabla P^+(x^{k+1}) = \nabla P(x^{k+1})$, $\nabla P^-(x^k) = \nabla P(x^k)$, Eq. (37) can be rewritten as

$$\begin{aligned} \nabla P(x^{k+1}) - \nabla P(x^k) &= U(x^{k+1} - b) + V^+(x^{k+1} - b) - U(x^k - b) - V^-(x^k - b) \\ &= U(x^{k+1} - x^k) + V^+(x^{k+1} - b) - V^-(x^k - b). \end{aligned} \tag{39}$$

Using the definition of b [Eq. (32)], we can simplify Eq. (39):

$$\begin{aligned} U(x^{k+1} - x^k) &= \nabla P(x^{k+1}) - \nabla P(x^k) - V^+(x^{k+1} - x^k - \alpha p^k) \\ &\quad + V^-(x^k - x^k - \alpha p^k), \\ Up^k &= \nabla P(x^{k+1}) - \nabla P(x^k) - (1 - \alpha)\, V^+ p^k - \alpha V^- p^k, \end{aligned} \tag{40}$$

where the definition of p^k has been applied. Let us define

$$q^k = \nabla P(x^{k+1}) - \nabla P(x^k) - (1 - \alpha)\, V^+ p^k - \alpha V^- p^k. \tag{41}$$

When we use Eq. (41), Eq. (40) becomes

$$q^k = U^k p^k, \tag{42}$$

where U^k is the kth estimate of U.

Now, let us consider the computation of q^k. First, if x^k and x^{k+1} are close to the boundary, the quadratic approximations for P will be accurate, and, within this region, the Hessian matrices are nearly constant. Therefore, it is reasonable to set

$$V^+ = V(x^{k+1}), \qquad V^- = V(x^k), \tag{43}$$

where $V(x^{k+1})$ and $V(x^k)$ can be computed directly from the defining equation (17). Second, the scalar α, locating the boundary point b, can be computed by utilizing the continuity of the penalty function P across b. That is, $P^+(b) = P^-(b) = P(b)$. Using this fact and the defining equations, we can solve Eqs. (33) and (34), for

$$\alpha = \left\{ \frac{2[P(x^{k+1}) - P(x^k) - \nabla P(x^k)^T p^k] + (p^k)^T (U^{k-1} + V^+) p^k}{(p^k)^T (V^+ - V^-) p^k} \right\}^{1/2}. \quad (44)$$

Using the rank-one recursion formula [Eq. (26)] again, where now y^k is replaced by q^k and Q^k is replaced by U^k, we obtain the recursive formula for

$$U^k = U^{k-1} + (q^k - U^{k-1}p^k)(q^k - U^{k-1}p^k)^T/(p^k)^T (q^k - U^{k-1}p^k). \quad (45)$$

Since we expect U to approach zero, a reasonable value to use for beginning the iterative process is $U^0 = 0$.

To recapitulate, the V-matrix is computed at every iteration using the definition [Eq. (17)]. If the set of violated constraints is different at x^{k+1} than at x^k, α is computed using Eq. (44); otherwise, $\alpha = 1$. The vectors q^k and p^k are then computed, and finally the matrix U is updated using Eq. (45). None of the expressions require second-derivative information. In particular, we need store only the values of the penalty P, its gradient ∇P, and the Gauss estimate of the Hessian matrix V, all evaluated at the last two successive points x^{k+1} and x^k.

At this point, procedures for constructing the matrices T, U, and V have been described. Using the definition [Eq. (18)], we obtain directly the complete Hessian matrix H. The least-square process described in Section 7 can then be used to determine the search step s.

10. Optimization Algorithm

Let us describe the kth iteration of the optimization algorithm. It is assumed that an estimate of the optimum x^k is available. In addition, the function values $f(x^k)$ and $P(x^k)$, as well as the gradients $\nabla f(x^k)$ and $\nabla P(x^k)$, have been computed. Finally, let us assume that values for T^k, U^k, and V^k have been computed (or initialized). The algorithm proceeds as follows.

Step (1). Define r.

Step (2). Compute $J(x^k)$ using Eq. (4).

Step (3). Compute $H(x^k) = H^k$ using Eq. (18).

Step (4). Compute $\nabla J(x^k)$ and $\| \nabla J(x^k) \|$ using Eq. (15).

Step (5). Perform convergence test: if $\| \nabla J(x^k) \| \leqslant \delta_1$, terminate algorithm.

Step (6). Using least-square procedure, determine $\tilde{s}^k$ to minimize Eq. (22), subject to Eq. (23).

Step (7). Define downhill direction from $\tilde{s}^k$ and $\nabla J(x^k)$ using Eq. (24).

Step (8). Determine ρ^k such that $J(x^{k+1}) \leqslant J(x^k)$, using cubic search on line [Eq. (21)]. Set improved point to x^{k+1}, and store function and gradient there; i.e., store $J(x^{k+1}), f(x^{k+1}), P(x^{k+1}), \nabla f(x^{k+1}), \nabla P(x^{k+1})$, and $\nabla J(x^{k+1})$.

Step (9). Compute V^{k+1} using definition [Eq. (17)].

Step (10). Update T matrix as follows:

(a) compute y^k and p^k using Eqs. (28) and (29);

(b) perform convergence test: if $\| p^k \| / \| x^k \| \leqslant \delta_2$, terminate algorithm;

(c) compute T^{k+1} using Eq. (31).

Step (11). Update U matrix as follows:

(a) if the problem is unconstrained, proceed to Step (12);

(b) test set of constraints violated at x^{k+1}: if unchanged from x^k, set $\alpha = 1$; if different than at x^k, compute α using Eq. (44);

(c) compute q^k using (41);

(d) using p^k from Step (10), compute U^{k+1} using Eq. (45).

Step (12). Set $k = k + 1$. Update point information; i.e., store x^{k+1} in x^k, $\nabla f(x^{k+1})$ in $\nabla f(x^k)$, *etc.* Return to Step (1).

11. Performance of the Algorithm

The algorithm described has been implemented in a computer program called BEST (Ref. 12). It was tested on 14 test problems of varying complexity, which are presented in Appendix B for reference.

The first five problems are unconstrained and consequently illustrate the effectiveness of the T-matrix recursion process in conjunction with the least-square procedure.

Table 1. Unconstrained results for Problems 14.1 through 14.5.

Problem	NFE	$f(x^0)$	$f(x^*)$	x^*
14.1	4	8.0	2.007×10^{-30}	$(1.9567 \times 10^{-15}, 1.1934 \times 10^{-15})$
14.2	51	24.2	5.003×10^{-14}	(0.99999990, 0.99999981)
14.3	37	250.0	9.617×10^{-17}	$(1.0, 3.104 \times 10^{-9}, 5.713 \times 10^{-9})$
14.4	39	215.0	7.053×10^{-9}	$(7.0701985 \times 10^{-3}, -7.0701985 \times 10^{-4}, 4.0958387 \times 10^{-3}, 4.0958387 \times 10^{-3})$
14.5	51	24.2	5.003×10^{-14}	(0.99999990, 0.99999981, 1.0)

The unconstrained results are presented in Table 1. The first column defines the problem number. The second column gives the total number of function evaluations (NFE) and associated gradient evaluations required to solve the problem. The third column gives the initial value of the performance index; the fourth, the optimum value; and the fifth, the optimum point.

Problem 14.1 represents the ideal performance of the algorithm, since the performance index is quadratic. The gradient differences were computed between the first and second points, and then the second and third points. Since the Hessian matrix was exact after the update at the third point, the final function evaluation (NFE = 4) was made at the exact optimum.

Problem 14.2 is the standard nonlinear programming test problem. Typical computer programs implementing the Davidon–Fletcher–Powell algorithm (Ref. 13) report solutions to this problem after 134 to 200 function evaluations, although some newer versions of the original algorithm report solutions in 60 evaluations. One reason BEST performs well on this problem is that it does not require accurate one-dimensional searches.

Problems 14.3 and 14.4 are both nonquadratic functions used as test problems for the original Davidon–Fletcher–Powell method. Problem 14.4 is especially significant, since the Hessian matrix becomes singular near the optimum point. In fact, after 15 function evaluations, the performance index had been reduced to 3.79×10^{-2} from 215, although another 24 function evaluations were necessary to satisfy the gradient and resolution accuracy tests.

Problem 14.5 is essentially the same as Problem 14.2, except that it is a singular problem since any value of x_3 will minimize f. The solution obtained for x_3 was unchanged from the initial guess. This problem demonstrates the effectiveness of the least-square approach for deter-

mining the search step. Since the Hessian is singular, direct inversion is impossible for this problem. Also, in terms of function evaluations, the singular Problem 14.5 was no more difficult than the nonsingular Problem 14.2.

Problem 14.6 illustrates the method used for solving constraints, that is, solving a set of nonlinear equations. In addition, it shows how more information about the problem can improve performance. The penalty term for this problem is simply the Rosenbrock function of Problem 14.2. The difference is that, in this case, the performance index is known to be of least-square form. Thus, while the function being minimized is identical, Problem 14.6 was solved in 37 function evaluations, as opposed to the 51 required for Problem 14.2.

Problems 14.7 and 14.8 were drawn from typical physical applications. Both problems, as posed, are poorly scaled, and both involve performance indices that are not easily approximated. Tables 2 and 3 indicate how the easy-r algorithm worked in conjunction with the optimization method. In these tables, the operation code is as follows: OP = unconstrained optimization problem; AE = asymptotic extrapolation of $x(r)$; DR = determination of r_{k+1}.

Table 2 summarizes the performance of the algorithm on Problem 14.7. The problem began with $r = 1$, and the first problem was solved in 12 function evaluations. A new value of r was determined using the minimum increment $\kappa = 1.1$. The second problem posed began at the solution of the first problem with $r = 1.1$ and was solved with two additional function evaluations. The solutions obtained for $r = 1$ and $r = 1.1$ were used to made an estimate of the asymptote. After evaluating the function at the asymptote, the easy-r was computed at the asymptote

Table 2. Easy-r algorithm performance on Problem 14.7.

Iteration, k	Operation	Function evaluations per operation	Cumulative number of function evaluations	r at end of operation
1	OP	12	12	1.000
	DR	0	12	1.100
	OP	2	14	1.100
2	AE	1	15	1.100
	DR	0	15	36.655
3	OP	1	16	36.655
	AE	1	17	36.655

Table 3. Easy-r algorithm performance on Problem 14.8.

Iteration, k	Operation	Function evaluations per operation	Cumulative number of function evaluations	r at end of operation
1	OP	16	16	1.000
	DR	0	16	1.100
	OP	2	18	1.100
2	AE	1	19	1.100
	DR	0	19	7.329
	OP	1	20	7.329
3	AE	1	21	7.329
	DR	0	21	331.660
4	OP	0	21	331.660
	AE	1	22	331.660

and found to be $r = 36.655$. A new optimization problem was then solved with $r = 36.655$, beginning at the asymptote. This problem required one function evaluation. The solutions for $r = 1$, $r = 1.1$, and $r = 36.655$ when extrapolated to the asymptote, produced the solution point, at the seventeenth function evaluation.

Table 3 summarizes the performance of the easy-r algorithm on Problem 14.8. Four unconstrained optimization problems were solved, with the last problem ($r = 331.66$) requiring no additional function evaluations; the asymptotic point was both the starting point and the solution point for the problem.

The performance of the easy-r algorithm on these examples is significant in the following ways. The easy-r algorithm produced a sequence of penalty weights that increased rapidly: in the first case, $\{r_k\} = \{1, 1.1, 36.655\}$; in the second case, $\{r_k\} = \{1, 1.1, 7.329, 331.66\}$. Furthermore, the unconstrained optimization problems posed were solved easily, although the value of r became large. In fact, after the first problem was solved, solutions to the remaining problems were obtained with only one or two additional function evaluations per problem.

The results of the algorithm on the remaining problems are summarized in Table 4. The first column defines the problem number; the second column, the total number of function evaluations necessary to solve the problem. The third column gives the total number of unconstrained problems that had to be solved, that is, the total number of

Table 4. Results of easy-r algorithm on Problems 14.9 through 14.14.

Problem	NFE	Number of optimization problems	Number of extrapolations	Number of restarts	Max r
14.9	7	3	2	0	112.1
14.10	25	3	2	0	2.2
14.11	55	7	4	2	3773.0
14.12	70	13	12	0	2437.0
14.13	101	10	7	2	72710.0
14.14	126	17	10	6	224900.0

different values of r in the sequence $\{r_k\}$. The fourth column defines the total number of asymptotic extrapolations made. The total number of times that the extrapolation process had to be restarted, because of a change in the constraint set, is given in the fifth column. The sixth column gives the maximum value of r, for which an unconstrained optimization problem was solved.

Problem 14.9 demonstrates the ideal performance of the algorithm, since the constraints are linear and the performance index is quadratic. Problems 14.10 through 14.12 all have nonlinear constraints and, consequently, benefit from the U matrix updating process. Problem 14.10 dramatically illustrates the effectiveness of the U-matrix recursive process for nonlinear constraints; without the U-matrix recursion procedure, this problem required 126 function evaluations—more than five times as many as the 25 reported.

Problems 14.13 and 14.14 are, perhaps, the most difficult of all the test problems. Problem 14.13 has a quadratic performance index in five variables, with 16 nonlinear inequality constraints; Problem 14.14 has a quartic performance index in 16 variables with 40 linear inequality constraints. Furthermore, a number of the constraints lie very close together at the optimum point, causing the easy-r algorithm to restart the extrapolation process repeatedly. According to Colville's report (Ref. 15), the interior point penalty function method of Fiacco and McCormick solved Problem 14.13 in 200 function evaluations. A penalty function method used by Davies solved Problem 14.13 in 124 evaluations and Problem 14.14 in 221 evaluations. While BEST compares favorably with the penalty function methods on these problems, other techniques reported by Colville solve these problems in 23 and 10 function evaluations, respectively.

12. Summary and Conclusions

The paper develops an algorithm for effectively solving the general constrained parameter optimization problem. Constraints are dealt with using an accelerated exterior point penalty function method. The easy-r algorithm poses a sequence of unconstrained optimization problems that are easily solved, because each problem begins with a good estimate of the solution $x^*(r_k)$ and the value of r_k is chosen to make the problem computationally easy.

The unconstrained optimization algorithm developed uses a rank-one recursive estimate of the Hessian matrix. Accurate one-dimensional searches are unnecessary. The search direction is determined using a least-square process, and, consequently, the algorithm is effective for singular or nearly-singular problems. The recursive process applied to the Hessian matrix of the penalty function eliminates the usual difficulties with discontinuous second derivatives and accelerates convergence for nonlinear constraints. Finite convergence is exhibited on a quadratic performance index with linear constraints; accelerated convergence is demonstrated on nonquadratic performance indices and nonlinear constraints. The algorithm is illustrated with a number of computational examples.

13. Appendix A: Scaling the Problem

In principle, the optimization algorithm, when used in conjunction with the easy-r algorithm, can be applied directly to the problem posed in Section 2. In practice, however, the behavior of the method can be substantially improved if some precautions are taken to ensure that the problem is appropriately scaled and normalized.

In particular, it is desirable to have the range of all the variables of roughly the same order of magnitude. Suppose that the problem is posed in terms of the variables x, which lie in the region

$$a \leqslant x \leqslant b.$$

For numerical purposes, it is desirable to work with a different set of variables x', which lie in the region

$$0 \leqslant x' \leqslant 1.$$

This can be accomplished by making the simple linear transformation

$$x_i' = (x_i - a_i)/r_i\,, \tag{46}$$

where the range of the ith variable is $r_i = b_i - a_i$. The gradients of J, in terms of x_i, must be multiplied by r_i to obtain the gradients with respect to x_i'. This range normalization process is done automatically within the computer program.

In general, a second type of scaling is also desirable. Again, for numerical purposes, it is usually better to work with a weighted performance index, for example, J', rather than J. In particular, let us construct

$$J' = w_0 f + r \sum_{i=1}^{m} I[c_i]\{(\sqrt{w_i})\, c_i\}^2. \tag{47}$$

Let us choose

$$w_0 = 1/|f|. \tag{48}$$

Furthermore, let us choose w_i such that

$$\sum_{i=1}^{m} w_i c_i^2 = 1, \tag{49}$$

and also

$$w_i \| \nabla c_i \| = w_m \| \nabla c_m \|, \qquad i = 1,..., m-1. \tag{50}$$

Essentially, Eq. (49) normalizes the total penalty term, and Eq. (50) chooses weights to equalize the gradients of all the constraints. Let us rewrite Eq. (49) as

$$\sum_{i=1}^{m-1} w_i c_i^2 + w_m c_m^2 = 1. \tag{51}$$

Solving Eq. (50) for w_i gives

$$w_i = w_m \| \nabla c_m \| / \| \nabla c_i \|, \qquad i = 1,..., m-1. \tag{52}$$

Combining Eqs. (51) and (52) and solving for w_m yields

$$w_m = \left[c_m^2 + \| \nabla c_m \| \cdot \sum_{i=1}^{m-1} (c_i^2 / \| \nabla c_i \|) \right]^{-1}. \tag{53}$$

Once w_m is determined from Eq. (53), all of the w_i can be computed from Eq. (52). Within BEST, this penalty function normalization is performed at the very first function evaluation and thereafter is not changed. Certainly, if the initial point x^0 is *bad* in some sense, the scaling process may prove useless. In general, however, the described scaling has proved quite adequate. As with the range normalization process, the gradients of J must be appropriately modified.

14. Appendix B: Test Problems

The following representative test problems were used to assess the capability of BEST.

Problem 14.1. Quadratic function of two variables:

$$f(x_1, x_2) = x_1^2 - 2x_1x_2 + 2x_2^2.$$

Minimize f. Start at (4, 2).

Problem 14.2. Rosenbrock's valley:

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

Minimize f. Start at (−1.2, 1).

Problem 14.3. Helical valley in three dimensions (see Fletcher and Powell, Ref. 13):

$$f(x_1, x_2, x_3) = 100\{[x_3 - 10\theta(x_1, x_2)]^2 + [r(x_1, x_2) - 1]^2\} + x_3^2,$$

where

$$2\pi\theta(x_1, x_2) = \begin{cases} \arctan(x_2/x_1), & x_1 > 0, \\ \pi + \arctan(x_2/x_1), & x_1 < 0, \end{cases}$$

and

$$r(x_1, x_2) = (x_1^2 + x_2^2)^{1/2}.$$

Minimize f. Start at (−1, 0, 0).

Problem 14.4. Fourth-degree polynomial (Powell, Ref. 13):

$$f(x_1, x_2, x_3, x_4) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4.$$

Minimize f. Start at (3, −1, 0, 0).

Problem 14.5. Singular problem—Rosenbrock's valley in three dimensions:

$$f(x_1, x_2, x_3) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

Minimize f. Start at (−1.2, 1, 1).

Problem 14.6. Constraint solving:

$$f(x_1, x_2) = 0,$$

$$c_1(x_1, x_2) = 10(x_2 - x_1^2) = 0, \qquad c_2(x_1, x_2) = 1 - x_1 = 0.$$

Minimize f, subject to constraints. Start at $(-1.2, 1)$.

Problem 14.7. Three-stage specific thrust equations with one linear equality constraint (Ref. 14):

$$\begin{aligned} f(x_1, x_2, x_3) = {} & GI_1 \log[(x_1 + x_2 + x_3 + W)/(x_1G_1 + x_2 + x_3 + W)] \\ & + GI_2 \log[(x_2 + x_3 + W)/(x_2G_2 + x_3 + W)] \\ & + GI_3 \log[(x_3 + W)/(x_3G_3 + W)], \end{aligned}$$

$$c_1(x_1, x_2, x_3) = x_1 + x_2 + x_3 - 1 = 0,$$

where

$$G = 32.174, \qquad G_1 = 0.09, \qquad G_2 = 0.07, \qquad G_3 = 0.13,$$

$$W = 0.03, \qquad I_1 = 255, \qquad I_2 = 280, \qquad I_3 = 290.$$

Maximize f, subject to constraint. Start at (0.7, 0.2, 0.1).

Problem 14.8. Six-dimensional exponential with one linear equality constraint (Ref. 14):

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = \exp(-Q/2),$$

where

$$\begin{aligned} Q = {} & (1 - \rho^2)^{-1} [(x_1 - \mu_1)^2/\sigma_1^2 + 2\rho(x_1 - \mu_1)(x_2 - \mu_2)/\sigma_1\sigma_2 + (x_2 - \mu_2)^2/\sigma_2^2] \\ & + (x_3 - \mu_3)^2/\sigma_3^2 + (x_4 - \mu_4)^2/\sigma_4^2 + (x_5 - \mu_5)^2/\sigma_5^2 + (x_6 - \mu_6)^2/\sigma_6^2, \end{aligned}$$

$$c_1(x_1, x_2, x_3, x_4, x_5, x_6) = (x_1 - \mu_1) - 0.2\sigma_1 + 4000(x_2 - \mu_2) - 2000\sigma_2$$

with $\rho = 0.2$ and

$$\mu_1 = 10000, \qquad \mu_2 = 1, \qquad \mu_3 = 2.0 \times 10^6,$$

$$\mu_4 = 10, \qquad \mu_5 = 0.001, \qquad \mu_6 = 1.0 \times 10^8,$$

$$\sigma_1 = 8000, \qquad \sigma_2 = 1, \qquad \sigma_3 = 7.0 \times 10^6,$$

$$\sigma_4 = 50, \qquad \sigma_5 = 0.05, \qquad \sigma_6 = 5.0 \times 10^8.$$

Maximize f, subject to constraint. Start at $(6000, 1.5, 4.0 \times 10^6, 2, 0.003, 5.0 \times 10^7)$.

Problem 14.9. Two-dimensional quadratic function with two linear inequality constraints:

$$f(x_1, x_2) = 100 - (0.01x_1^2 + x_2^2),$$

$$c_1(x_1, x_2) = x_1 - 2 \geqslant 0, \qquad c_2(x_1, x_2) = 10x_1 - x_2 - 10 \geqslant 0.$$

Maximize f, subject to constraints. Start at $(-1, -1)$.

Problem 14.10. Two-dimensional quadratic function with three nonlinear inequality constraints:

$$f(x_1, x_2) = 0.01x_1^2 + x_2^2,$$

$$c_1(x_1, x_2) = x_1x_2 - 25 \geqslant 0, \qquad c_2(x_1, x_2) = x_1^2 + x_2^2 - 25 \geqslant 0,$$

$$c_3(x_1, x_2) = x_1 - 2 \geqslant 0.$$

Minimize f, subject to constraints. Start at $(2, 2)$.

Problem 14.11. Two-dimensional quadratic function with five nonlinear inequality constraints:

$$f(x_1, x_2) = x_1^2 + x_2^2,$$

$$c_1(x_1, x_2) = x_1^2 + x_2^2 - 1 \geqslant 0, \qquad c_2(x_1, x_2) = 9x_1^2 + x_2^2 - 9 \geqslant 0,$$

$$c_3(x_1, x_2) = x_1 + x_2 - 1 \geqslant 0, \qquad c_4(x_1, x_2) = x_1^2 - x_2 \geqslant 0,$$

$$c_5(x_1, x_2) = x_2^2 - x_1 \geqslant 0.$$

Minimize f, subject to constraints. Start at $(3, 1)$.

Problem 14.12. Rosenbrock's valley with three nonlinear inequality constraints:

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2,$$

$$c_1(x_1, x_2) = x_1x_2 - 1 \geqslant 0, \qquad c_2(x_1, x_2) = x_2^2 + x_1 \geqslant 0,$$

$$c_3(x_1, x_2) = -x_1 + 1/2 \geqslant 0.$$

Minimize f, subject to constraints. Start at $(-2, 1)$.

Problem 14.13. Colville test problem No. 3 (Ref. 15).

Problem 14.14. Colville test problem No. 7 (Ref. 15).

References

1. FIACCO, A. V., and MCCORMICK, G. P., *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, New York, New York, 1968.
2. BETTS, J. T., and CITRON, S. J., *Approximate Optimal Control of Distributed Parameter Systems*, AIAA Journal, Vol. 10, No. 1, 1972.
3. BETTS, J. T., *An Approximation Technique for Determining the Optimal Control of a Distributed Parameter System*, Purdue University, Department of Aeronautics, Astronautics, and Engineering Science, Ph.D. Thesis, 1970.
4. JOHNSON, F. T., *Approximate Finite-Thrust Trajectory Optimization*, AIAA Journal, Vol. 7, No. 6, 1969.
5. TAYLOR, L. W., JR., SMITH, H. J., and ILIFF, K. W., *Experience Using Balakrishnan's Epsilon Technique to Compute Optimum Flight Profiles*, Journal of Aircraft, Vol. 7, No. 2, 1970.
6. POWELL, M. J. D., *A Method for Nonlinear Constraints in Minimization Problems*, Optimization, Edited by R. Fletcher, Academic Press, New York, New York, 1969.
7. MIELE, A., COGGINS, G. M., and LEVY, A. V., *Updating Rules for the Penalty Constant Used in the Penalty Function Method for Mathematical Programming Problems*, Ricerche di Automatica, Vol. 3, No. 2, 1972.
8. HANSON, R. J., and LAWSON, C. L., *Extensions and Applications of the Householder Algorithm for Solving Linear Least Squares Problems*, Mathematics of Computation, Vol. 23, No. 108, 1969.
9. DAVIDON, W. C., *Variance Algorithm for Minimization*, The Computer Journal, Vol. 10, No. 4, 1968.
10. BROYDEN, C. G., *The Convergence of Single-Rank Quasi-Newton Methods*, Mathematics of Computation, Vol. 24, No. 110, 1970.
11. MURTAGH, B. A., and SARGENT, R. W. H., *A Constrained Minimization Method with Quadratic Convergence*, Optimization, Edited by R. Fletcher, Academic Press, New York, New York, 1969.
12. BETTS, J. T., *An Effective Method for Solving Constrained Parameter Optimization Problems*, Aerospace Corporation, Report No. TR-0073(3450-10), El Segundo, California, 1972.
13. FLETCHER, R., and POWELL, M. J. D., *A Rapidly Convergent Descent Method for Minimization*, The Computer Journal, Vol. 6, No. 2, 1963.
14. PICKETT, H. E., *A Contribution to the Thaumaturgy of Nonlinear Programming*, Aerospace Corporation, Report No. ATR-71(S9990)-1, San Bernardino, California, 1970.
15. COLVILLE, A. R., *A Comparative Study on Nonlinear Programming Codes*, IBM, New York Scientific Center, Report No. 320-2949, 1968.