CONSTRAINED NONLINEAR OPTIMIZATION BY HEURISTIC PROGRAMMING

D. A. Paviani and D. M. Himmelblau

The University of Texas, Austin, Texas

(Received September 26, 1968)

This paper develops a new algorithm for solving the general nonlinear programming problem that melds the flexible simplex search of NELDER AND MEAD with various additional rules to take care of equality and/or inequality constraints. The set of violated inequalities and equalities is lumped into one inequality constraint loosely satisfied during the early progress of the optimization and more closely satisfied during its final stages. To permit this type of search, the method sets up a special tolerance criterion, a function that does not depend on either the values of the objective function or the values of the constraints. The new algorithm has solved successfully a number of problems that have been proposed in the literature as test problems. Finally, to indicate the algorithm's capabilities, the paper describes an example composed of a linear objective function of twenty-four variables subject to fourteen nonlinear equalities and thirty inequalities.

MANY algorithms have been proposed for solving the general nonlinear programming problem, which encompasses the optimization of an objective function subject, in the most general case, to both equality and inequality constraints. Although algorithms for unconstrained optimization appear to be in a satisfactory state of development, the status of optimization algorithms that can handle both equality and inequality constraints for realistic problems is still unresolved.

This paper describes a new approach to nonlinear programming problems incorporating equality and/or inequality constraints; it represents an extension of the method proposed by NELDER AND MEAD[4] for unconstrained problems.

Nonlinear programming techniques can be roughly divided into two broad categories: (a) direct search methods that depend upon a direct comparison of the value of the objective function, and (b) gradient methods that seek the extremum by using first- and perhaps second-order derivatives.

Gradient methods for nonlinear programming problems are, in general, faster than direct-search methods.[1] Gradient methods are extensively used in practice to give an indication of the best direction to follow in E^n in order to increase or decrease the value of the objective function. However, the gradient is a strictly local property of a function. Furthermore, the need to provide analytical functions or numerical values to evaluate the derivatives is a practical disadvantage of the gradient methods in comparison with the direct-search methods. Also, the amount of effort required by the user to introduce a problem into a direct-search algorithm is relatively low. Although direct-search algorithms are more time consuming in their execution, the net cost, including preparation time, for the solution of a problem may be less than for gradient methods. Thus, the algorithm presented here was designed according to direct-search logic.

STATEMENT OF THE GENERAL NONLINEAR OPTIMIZATION PROBLEM

GIVEN THE continuous functions $f(\boldsymbol{x})$, $h_1(\boldsymbol{x}), \cdots, h_m(\boldsymbol{x})$, $g_{m+1}(\boldsymbol{x}), \cdots, g_p(\boldsymbol{x})$, where $\boldsymbol{x}=(x_1, \cdots, x_n)$, it is required to find $\boldsymbol{x}^*=(x_1^*, \cdots, x_n^*)$ that minimizes (or, alternately, maximizes) $f(\boldsymbol{x})$ under the conditions that $h_i(\boldsymbol{x})=0$, $i=1, \cdots, m$, and $g_i(\boldsymbol{x}) \geqq 0$, $i=m+1, \cdots, p$, where $m<n$. The compact problem statement is:

$$\text{minimize } y=f(\boldsymbol{x}), \tag{1}$$

subject to

$$h_i(\boldsymbol{x})=0, \qquad (i=1, \cdots, m<n), \tag{2}$$

$$g_i(\boldsymbol{x}) \geqq 0, \qquad (i=m+1, \cdots, p), \tag{3}$$

where $\boldsymbol{x}$ is a vector in E^n. The functions in equations (1), (2), and (3) are nonlinear in $\boldsymbol{x}$, although some of them in special cases may be linear. Both m and/or $(p-m)$ may be equal to zero. When both $m=0$ and $(p-m)=0$, optimization of $f(\boldsymbol{x})$ will be unconstrained.

By definition, the difference $r=(n-m)$ specifies the number of degrees of freedom, e.g., the number of decision (independent) variables, an important concept in minimizing $f(\boldsymbol{x})$ in the presence of equality constraints. For instance, if $m=n$, no continuous optimization problem exists in general and $\boldsymbol{x}^*$ will be determined directly from the simultaneous solution of the system of equations $h_i(\boldsymbol{x})=0$, $i=1, \cdots, n$.

A vector is called feasible if it satisfies equations (2) and (3) and nonfeasible if one or more of these constraints are not satisfied. 'Interior' methods, such as SUMT of FIACCO AND MCCORMICK,[2] find the minimum of $f(\boldsymbol{x})$ by generating a sequence of feasible points. 'Exterior' methods, such as that of ZANGWILL,[7] generate a sequence of nonfeasible points, yielding a solution when the sequence becomes feasible. These two techniques are primarily concerned with the optimization of $f(\boldsymbol{x})$ under inequality constraints, and although Fiacco and McCormick[3] have extended the SUMT procedure[2] to include equality constraints, the extension has yet to be established as an effective technique for nonlinear equality constraints.

THE UNCONSTRAINED FLEXIBLE SIMPLEX METHOD

THE UNCONSTRAINED flexible simplex method of Nelder and Mead minimizes a function of n independent (decision) variables using $n+1$ vertices (vectors) of a flexible geometric polygon in E^n. The point (vertex) that yields the highest value of the objective function is projected through the center of gravity (centroid) of the remaining vectors. As improved values of the objective function are found, the point with the highest value of $f(x)$ is successively replaced by better points until the minimum of $f(x)$ is found. The details of the algorithm are as follows:

Let x^i, $i=1, \cdots, n+1$, be the point (a vertex) for which the value of the objective function is $f(x^i)$. In addition, let

$$f(x^h)=\max[f(x^i), \cdots, f(x^n)] \quad \text{for which} \quad x^i=x^h,$$
$$f(x^l)=\min[f(x^i), \cdots, f(x^n)] \quad \text{for which} \quad x^i=x^l.$$

Let x^{n+2} be the centroid of all the points with $i\neq h$, the components of which are given by

$$x_j^{n+2}=(1/n)[(\textstyle\sum_{i=1}^{n+1} x_j^i)-x_j^h]. \qquad (j=1, \cdots, n) \quad (4)$$

Given the initial x vectors for $n+1$ points, usually, but not necessarily, a regular simplex, the procedure for finding a better point involves four operations:

(a) Reflection of x^h through the centroid

$$x^{n+3}=x^{n+2}+\alpha(x^{n+2}-x^h). \quad (5)$$

(b) Expansion of $(x^{n+3}-x^{n+2})$, if reflection has produced a new minimum

$$x^{n+4}=x^{n+2}+\gamma(x^{n+3}-x^{n+2}). \quad (6)$$

(c) Contraction of (x^h-x^{n+2}), if reflection results in $f(x^{n+3})>f(x^i)$ for all $i\neq h$,

$$x^{n+5}=x^{n+2}+\beta(x^h-x^{n+2}). \quad (7)$$

(d) Over-all contraction of x^i on x^l if contraction failed to produce a better point than x^h,

$$x^i=0.5(x^l+x^i). \qquad (i=1, \cdots, n+1) \quad (8)$$

In the above equations, the reflection coefficient α is a positive constant, the expansion coefficient γ is a constant greater than unity, and the contraction coefficient β is a constant such that $0<\beta<1$. The values of $\alpha=1$, $\beta=0.5$, and $\gamma=2$ have been recommended for the unconstrained minimization problem.[4]

THE PROPOSED ALGORITHM

IN MINIMIZING $f(x)$, any move Δx from the position x in E^n may violate one or more of the restrictions imposed by equations (2) and (3). To provide a flexible measure of the extent of the violation, all the constraints $h_i(x)\neq 0$ and $g_i(x)<0$ that exceed a tolerance criterion Φ, defined below in equation (11), are combined into one gross inequality. The tolerance criterion acts as a filter to discriminate between propinquent feasible or 'near-feasible' points in the flexible polygon and less feasible points. A near-feasible point defined in this manner may be an interior or exterior point.

After incorporating the concept of near-feasibility, the general optimization problem at any stage of the search becomes one of minimizing $f(x)$ subject to one inequality constraint as follows:

$$\min y=f(x), \; x\epsilon E^n, \quad (9)$$

subject to

$$\Phi-[\textstyle\sum_{i=1}^{i=n} h_i^2(x)+\sum_{j=1}^{j=k} g_j^2(x)]^{1/2}\geqq 0. \quad (10)$$

In equation (10), $\sum g_j^2(x)$ includes only those k constraints for which $g_i(x)<0$; those for which $g_i(x)\geqq 0$ are excluded. For convenience, the summation $\sum h_i^2(x)$ includes all equalities, since there would be no contribution from $h_i(x)=0$. The criterion Φ is a positive decreasing function of x selected so as to depend only on the behavior of x^i in E^n during the progress of the optimization. A nonunique definition of Φ that conforms to the particular space configuration of the flexible simplex is

$$\Phi^s=\min[\Phi^{s-1}, (r+1)^{-1}\textstyle\sum_{i=1}^{r+1} \|x^i-x^{r+2}\|], \quad (11)$$

where $\Phi^0=2t$ is the initial value of Φ, t is the size of the initial simplex, r is the number of degrees of freedom, x^i is the ith vertex of the simplex, x^{r+2} the centroid of the simplex, $s=0, 1, \cdots$ indicates the number of completed stages, Φ^{s-1} is the value of the tolerance criterion on the $(s-1)$th stage of the search, and $\| \; \|$ denotes the magnitude of the vector (x^i-x^{r+2}), $i=1, \cdots, r+1$.

The second term in the brackets of expression (11) represents the average distance from each x^i, $i=1, \cdots, r+1$, to the centroid x^{r+2} of the geometric polygon in E^n. If x lies within the boundaries of the feasible region defined by equation (10), Φ will not depend upon the values of $f(x)$. Because of the manner in which Φ is chosen in expression (11), it is a strictly positive decreasing function of x^i, $i=1, \cdots, r+2$; that is,

$$\Phi^0\geqq\Phi^1\geqq\cdots\geqq\Phi^s\geqq 0. \quad (12)$$

After any change in x^i, equation (10) is checked regardless of any possible improvement in the value of the objective function. If equation (10)

is not satisfied, the sum of the squared values of all the violated constraints is minimized by an unconstrained minimization procedure until the square root of this sum is less than or equal to Φ. Then the value of the objective function is computed to determine whether the new point (vertex) is a success or a failure.

An effective unconstrained minimization algorithm is required in order to satisfy equation (10) after every move throughout the search. For convenience, the flexible simplex method,[4] after some additions and modifications, was used as the subroutine to satisfy equation (10). The possibility of using other unconstrained minimization procedures, such as the methods of ROSENBROCK[6] or POWELL[5] to solve equation (10) has not been investigated. The improvement of $f(x)$ itself is made up of a succession of stages. When using the flexible simplex procedure, a stage consists of one reflection of x and either an expansion or a contraction of x depending upon the success or failure of the reflection. Φ is evaluated after every stage.

Convergence of the search is assured because Φ is a positive decreasing function and because of the repeated application of equations (7) and (8) when it is not possible to find better values of $f(x)$ by equation (5). The vectors x^i, $i=1, \cdots, r+1$, draw nearer and nearer to the vector corresponding to the best value of $f(x)$, namely x^l, with the eventual complete collapse of the polygon at the minimum of $f(x)$. Hence, the tolerance criterion possesses the property that

$$\lim_{x \to x^*} \Phi = 0. \tag{13}$$

The criterion Φ thus serves two purposes. First, it acts as a tolerance criterion for constraint violation throughout the entire search. Second, it serves as a very practical convergence criterion to decide when to terminate the search. The extent of violation of the constraints given by equations (2) and (3) is progressively decreased as the search moves toward the solution of the optimization problem. For practical purposes it is sufficient to carry on the search until Φ becomes smaller than an arbitrarily selected small number ϵ. A $\Phi \leqq \epsilon$ means that on the average a permissible change ϵ in the position of the best x^i will not improve the objective function. Also, it means that the square root of the sum of the squares of the violated constraints is less than ϵ. Therefore, upon the termination of the search the following inequalities are satisfied:

$$f(x) \leqq f(x^* \pm \epsilon), \tag{14}$$

$$\left[\sum_{i=1}^{i=m} h_i^2(x) + \sum_{j=1}^{j=k} g_j^2(x)\right]^{1/2} \leqq \epsilon. \tag{15}$$

In summary, we have used the flexible simplex method described in the previous section to carry out both the search to improve $f(x)$, and, as modified, to maintain near-feasibility as defined by equation (10).

In order to initiate the search using the proposed algorithm, $r+1$ initial points ($n+1$ when $m=0$) or x vectors that may or may not form a regular simplex in E^n are required. The $r+1$ points should be chosen so that any subset of r vectors is linearly independent. For practical purposes it is convenient to assume a starting vector x^0 and build a regular simplex using x^0 as a base point. The $r+1$ vectors in E^n are found from

$$x^i = x^0 + D^i, \qquad (i=1, \cdots, r+1) \tag{16}$$

where the elements of D^i are the elements of the ith row of a $(r+1) \times n$ matrix. Therefore, the rows of this matrix determine the n coordinates of each of the sought $r+1$ vectors:

$$D = \begin{Vmatrix} 0 & 0 & 0 & \cdots & 0 \\ u & v & v & \cdots & v \\ v & u & v & \cdots & v \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ v & v & v & \cdots & u \end{Vmatrix}, \tag{17}$$

where

$$u = (t/n\sqrt{2})(\sqrt{n+1}+n-1), \tag{18}$$

$$v = (t/n\sqrt{2})(\sqrt{n+1}-1). \tag{19}$$

In equations (18) and (19), t is a constant that determines the length of the simplex edges. For instance, $t=1$ builds a simplex with a unit edge.

In order to start the search with the appropriate simplex size, t should be selected as a function of the expected interval of variation of the independent variables. Usually upper and lower bounds on x are known. In this case, equation (20) can be used as a reasonable estimate of t:

$$t = \min[(0.2/n)\textstyle\sum_{i=1}^{i=n} L_i, L_1, L_2, \cdots, L_n], \tag{20}$$

where L_i is the difference between the upper and lower bounds of the independent variable x_i. If the upper and lower bounds of x are not known, any reasonable guess for t is acceptable. It is advantageous to build the initial simplex with an x^0 that is nearly feasible, for if the initial simplex were formed far away from the feasible region, equation (10) would have to be satisfied $r+1$ times in order to bring all the simplex points into the

neighborhood of the feasible region. By choosing the first value of Φ equal to two times the value of t, that is $\Phi^0=2t$, a preliminary tolerance criterion to bring x close to or into the feasible region is available. In addition Φ^0 establishes the uppermost bound on the sequence of values for Φ.

A step-by-step description of the algorithm is given below. The frequent expression "satisfy equation (10)" means "form the sum of the squares of the values of the violated constraints, if any; if the square root of this sum is greater than the current Φ minimize the sum by means of the unconstrained minimization algorithm until a vector x is found for which equation (10) is satisfied."

1. Assume x^0 and find t either by using equation (20) or by guess. Choose $\Phi^0=2t$. Satisfy equation (10). Build a regular simplex starting with the x^0 for which equation (10) was satisfied. Compute the corresponding values of $f(x^i)$.

2. Compute Φ by equation (11).

3. Find x^h, the vector for which $f(x^i)$ is the largest; find x^l, the vector for which $f(x^i)$ is the smallest; and compute x^{r+2}, the centroid of all x^i for $i\neq h$.

4. Reflect x^h to form $x^{r+3}=x^{r+2}+\alpha(x^{r+2}-x^h)$ and satisfy equation (10). Compute $f(x^{r+3})$.

5. If $f(x^{r+3})\leqq f(x^l)$, form $x^{r+4}=x^{r+2}+\gamma(x^{r+3}-x^{r+2})$ and satisfy equation (10). Compute $f(x^{r+4})$. If $f(x^{r+4})<f(x^l)$, replace x^h and $f(x^h)$ by x^{r+4} and $f(x^{r+4})$, respectively, and go to step 11. Otherwise, replace x^h and $f(x^h)$ by x^{r+3} and $f(x^{r+3})$, respectively, and go to step 11.

6. If $f(x^{r+3})>f(x^l)$, but for some $i\neq h$, $f(x^{r+3})<f(x^i)$, replace x^h and $f(x^h)$ by x^{r+3} and $f(x^{r+3})$, respectively, and go to step 11.

7. If $f(x^{r+3})>f(x^h)$, go to step 9.

8. If $f(x^{r+3})\leqq f(x^h)$, replace x^h and $f(x^h)$ by x^{r+3} and $f(x^{r+3})$, respectively, and go to step 9.

9. Form $x^{r+5}=x^{r+2}+\beta(x^h-x^{r+2})$ and satisfy equation (10). Compute $f(x^{r+5})$.

10. If $(x^{r+5})<f(x^h)$, replace x^h and $f(x^h)$ by x^{r+5} and $f(x^{r+5})$, respectively, and go to step 11. Otherwise, replace all x^i and $f(x^i)$ by $0.5(x^i+x^l)$ and compute $f(x^i)$ for all new x^i. Return to step 2 to start the $(s+1)$th stage.

11. Determine if $\Phi\leqq\epsilon$. If 'yes,' the search is terminated; if 'no,' return to step 1 to start the $(s+1)$th stage.

RESULTS AND DISCUSSION

The algorithm described above has been programmed for a CDC 6600 digital computer. A set of problems used to test the efficiency and behavior

of nonlinear optimization codes was solved by the algorithm, and the computation times compared favorably with those presented by Colville[1] for direct-search methods. A number of additional problems have been tested, and the computational experience with the algorithm has encompassed about twenty-five problems with the number of independent variables ranging from two to twenty-four, and inequality constraints ranging from none to thirty. Only a few problems incorporating nonlinear equalities have been reported. Of the twenty-five problems solved, only three contained nonlinear equalities.

The amount of computer time required for the solution of problems with more than fifteen independent variables and as many constraints is considerably more than the time required by algorithms employing first and second derivatives. In problems with many independent variables, a major portion of the computation time was spent repetitively satisfying equation (10). Nevertheless, the advantage of not having to form and compute derivatives more than compensates for the additional amount of computer time.

To illustrate the capabilities of the algorithm in handling nonlinear optimization problems subject to equality and inequality constraints, a problem consisting of a linear objective function containing twenty-four variables subject to fourteen nonlinear equalities and thirty inequalities (of which six are nonlinear and twenty-four are the nonnegativity conditions) is now listed.

Ten degrees of freedom exist.

Objective function: $f(x)=\sum_{j=1}^{j=24} c_j x_j$.

Equality Constraints:

$$h_1(x):\ (x_{13}/M_{13})/S_V-(V_1/P)(x_1/M_1)/S_L=0,$$
$$h_2(x):\ (x_{14}/M_{14})/S_V-(V_2/P)(x_2/M_2)/S_L=0,$$
$$h_3(x):\ (x_{15}/M_{15})/S_V-(V_3/P)(x_3/M_3)/S_L=0,$$
$$h_4(x):\ (x_{16}/M_{16})/S_V-(V_4/P)(x_4/M_4)/S_L=0,$$
$$h_5(x):\ (x_{17}/M_{17})/S_V-(V_5/P)(x_5/M_5)/S_L=0,$$
$$h_6(x):\ (x_{18}/M_{18})/S_V-(V_6/P)(x_6/M_6)/S_L=0,$$
$$h_7(x):\ (x_{19}/M_{19})/S_V-(V_7/P)(x_7/M_7)/S_L=0,$$
$$h_8(x):\ (x_{20}/M_{20})/S_V-(V_8/P)(x_8/M_8)/S_L=0,$$
$$h_9(x):\ (x_{21}/M_{21})/S_V-(V_9/P)(x_9/M_9)/S_L=0,$$
$$h_{10}(x):\ (x_{22}/M_{22})/S_V-(V_{10}/P)(x_{10}/M_{10})/S_L=0,$$
$$h_{11}(x):\ (x_{23}/M_{23})/S_V-(V_{11}/P)(x_{11}/M_{11})/S_L=0,$$
$$h_{12}(x):\ (x_{24}/M_{24})/S_V-(V_{12}/P)(x_{12}/M_{12})/S_L=0,$$

$h_{13}(x)$: $\sum_{i=1}^{i=24} x_i - 1 = 0$.

$h_{14}(x)$: $\sum_{i=1}^{i=12} (x_i/A_i) + [(0.7302)(14.7)(T)/P] \sum_{j=13}^{j=24} (x_j/M_j) - 1.671 = 0$.

$S_L = \sum_{j=1}^{j=12} x_j/M_j$, $S_V = \sum_{j=13}^{j=24} x_j/M_j$, $S_X = \sum_{i=1}^{i=24} x_i$.

TABLE I

j		C	M	A	V
1	13	0.0693	44.094	31.244	123.7
2	14	0.0577	58.12	36.12	31.7
3	15	0.05	58.12	34.784	45.7
4	16	0.2	137.4	92.70	14.7
5	17	0.26	120.9	82.70	84.7
6	18	0.55	170.9	91.60	27.7
7	19	0.06	62.501	56.708	49.7
8	20	0.10	84.94	82.70	7.1
9	21	0.12	133.425	80.80	2.1
10	22	0.18	82.507	64.517	17.7
11	23	0.10	46.07	49.4	0.85
12	24	0.09	60.097	49.10	0.64

Inequality Constraints:

$$g_{15}(x): 0.1 - (x_1 + x_{13})/S_X \geqq 0,$$
$$g_{16}(x): 0.3 - (x_2 + x_{14})/S_X \geqq 0,$$
$$g_{17}(x): 0.4 - (x_3 + x_{15})/S_X \geqq 0,$$
$$g_{18}(x): 0.3 - (x_7 + x_{19})/S_X \geqq 0,$$
$$g_{19}(x): 0.6 - (x_8 + x_{20})/S_X \geqq 0,$$
$$g_{20}(x): 0.3 - (x_9 + x_{21})/S_X \geqq 0,$$
$$g_{21}(x), \cdots, g_{44}(x): x_i \geqq 0. \qquad (i = 1, \cdots, 24)$$

(Note that $S_X = 1$ only at the solution of the problem.)

One set of known variables and coefficients was as follows: $P = 40$ (psia), $T = 530$ (°R), and the values given in Table I.

The initial parameters and the initial vector, x^0, for the algorithm were selected as follows: $\alpha = 1$, $\beta = 0.2$, $\gamma = 2$, $t = 0.1$, $\epsilon = 10^{-5}$, $x_i^0 = 0.04$, $i = 1, \cdots, 24$; $f(x^0) = 0.147$. Twenty-four values of x were sought to minimize $f(x)$.

After 590 stages of calculation the following results were obtained:

$f(x^*) = 0.058106$, $\Phi = 5.69 \cdot 10^{-6}$, and (the leading element is x_1^*, the second element in the first row is x_2^*, and so forth) the values of the x's were

$$x^* = \begin{Vmatrix} 0.012712 & 0.090973 & 0.115287 & \propto 10^{-7} & \propto 10^{-7} & \propto 10^{-7} \\ 0.069021 & 0.000330 & 0.000029 & \propto 10^{-8} & 0.016675 & 0.028777 \\ 0.079418 & 0.145656 & 0.266101 & \propto 10^{-7} & \propto 10^{-7} & \propto 10^{-8} \\ 0.173254 & 0.000111 & 0.000003 & 0.000002 & 0.00072 & 0.000925 \end{Vmatrix}.$$

The values of the constraints were: $h_i(x^*) =$

$$\begin{Vmatrix} 1.2 \cdot 10^{-7} & 3.2 \cdot 10^{-7} & 8.6 \cdot 10^{-7} & 1.7 \cdot 10^{-7} & -1.8 \cdot 10^{-6} & -5.6 \cdot 10^{-7} \\ -1.1 \cdot 10^{-6} & 6.9 \cdot 10^{-6} & 1.5 \cdot 10^{-7} & 1.9 \cdot 10^{-6} & 6.0 \cdot 10^{-7} & -6.6 \cdot 10^{-6} \\ -2.4 \cdot 10^{-6} & 2.5 \cdot 10^{-6} \end{Vmatrix},$$

$$g_i(x^*) = \begin{Vmatrix} 0.007869 & 0.0063369 & 0.018610 & 0.057724 & 0.599558 \\ 0.299967 \end{Vmatrix}.$$

A shift in the convergence criterion ϵ changed the time for execution of the algorithm as follows:

ϵ	Time (min)
10^{-3}	13.2
10^{-4}	15.0
10^{-5}	17.9

A value of $\epsilon = 10^{-5}$ means that changes are being made in the search in the fifth significant figure of each of elements of x. This problem is complex enough so that it should provide a good challenge for any nonlinear programming algorithm.

REFERENCES

1. A. R. Colville, "A Comparative Study on Nonlinear Programming Codes," IBM Technical Report No. 320-2949, June, 1968.
2. A. V. Fiacco and G. P. McCormick, "The Sequential Unconstrained Minimization Technique for Nonlinear Programming, a Primal-Dual Method," *Management Sci.* **10**, 360 (1964).
3. ——— and ———, "The Sequential Unconstrained Minimization Technique for Convex Programming with Equality Constraints," Research Analysis Corporation, RAC-TP-155, April, 1965.
4. J. A. Nelder and R. Mead, "A Simplex Method for Function Minimization," *Comp. J.* **7**, 308 (1964).
5. M. J. D. Powell, "An Efficient Method for Finding the Minimum of a Func-

tion of Several Variables without Calculating Derivatives," *Comp. J.* **7,** 155 (1964).
6. H. H. Rosenbrock, "An Automatic Code for Finding the Greatest or Least Value of a Function," *Comp. J.* **3,** 175 (1960).
7. W. I. Zangwill, "Nonlinear Programming Via Penalty Functions," *Management Sci.* **13,** 344 (1967).

CLOSURE OF THE RIGHT-HAND-SIDE SET FOR SYSTEMS OF NONLINEAR INEQUALITIES

F. J. Gould and H. Pashner
The University of North Carolina, Chapel Hill, N.C.

(Received October 1, 1968)

Consider the set of all vectors b such that the system $g(x) \leqq b$ has a solution, where $g: D \to R^m$, $D \subseteq R^n$. This paper presents necessary and sufficient conditions for this set to be closed, with special results on continuous and convex functions, and gives an application to mathematical programming.

WE CONSIDER mappings of the form $g: D \to R^m$, where D is a nonempty subset of R^n. With such a mapping we identify m functions $g_1(x), \cdots, g_m(x)$, $x \epsilon D$, and an associated (nonempty) set $B = \{b \epsilon R^m: g(x) \leqq b \text{ for some } x \epsilon D\}$, where the inequality $g(x) \leqq b$ means $g_j(x) \leqq b_j$, $j = 1, \cdots, m$. Our purpose is to present necessary and sufficient conditions for B to be a closed set. Section 1 discusses the problem without further qualification on g, and proves a necessary and sufficient condition. Section 2 gives special consideration to more restricted functions (continuous, convex). Finally, Section 3 presents two examples in mathematical programming.

One reason for interest in the set B relates to its role as the domain of the primal supremal function in mathematical programming. In particular, suppose $f: D \to R$ and let $\bar{b}$ be a specified element of R^m. Then consider the primal problem:

$$\text{supremize } f(x) \text{ subject to} \tag{1}$$

$$x \epsilon D \quad \text{and} \quad g(x) \leqq \bar{b}. \tag{2}$$

Now, given any $b \epsilon B$ we identify the feasible set

$$S_b = \{x \epsilon D: g(x) \leqq b\}. \tag{3}$$

In different notation, the primal problem can be expressed as supremize $f(x)$ subject to $x \epsilon S_{\bar{b}}$. On the set B the primal supremal function, designated $f_{\sup}$, is now defined as

$$f_{\sup}: B \to R \cup \{+\infty\}, \quad f_{\sup}(b) = \sup_{x \epsilon S_b} f(x), \quad b \epsilon B. \tag{4}$$

The dual to (1) and (2) can now be written[2] as

$$\text{infimize } (u, \bar{b}) + u_0 \quad \text{subject to} \tag{5}$$

$$u \epsilon R^m, \quad u \geqq 0, \quad u_0 \epsilon R, \quad \text{and} \quad (u, b) + u_0 \geqq f_{\sup}(b), \quad \text{all} \quad b \epsilon B, \tag{6}$$

where the variables are u, u_0, and (u, b) is the inner product in R^m.