1974

An Introduction to Optimization Methods for Engineering Design

K. M. Ragsdell
Purdue University

AN INTRODUCTION TO OPTIMIZATION METHODS FOR ENGINEERING DESIGN

K. M. Ragsdell
Assistant Professor
School of Mechanical Engineering
Purdue University
West Lafayette, Indiana 47907

ABSTRACT

The engineering design process is a multi-faceted endeavor. Ideation, modelling, analysis, decision making and optimization certainly play important roles. Optimization is a natural design activity, and is enjoying ever increasing usage due in large part to the increasing availability of high speed digital computers. This paper discusses several optimization methods which seem to be useful in a design environment. References are given to stimulate and guide additional study.

THE ENGINEERING DESIGN PROCESS

"The design process is a trial and error sequence of choices among a number of alternatives, in which each decision is affected by compromise between a number of conditions and constraints. It demands meticulous attention to detail, coordination of a wealth of information, the search for ideas at each stage, and an over-all necessity to achieve the best performance at the lowest cost in the shortest time" [1].

Thoughts of this type lead to a morphology or structure of design as shown in Figure 1. Very often a given designer will, through experience or prejudice, elevate the importance of certain elements to the discredit of the others. A fair examination of the engineering design process will show these elements to be, at the very least, of importance. There is, in fact, a rather strong relationship and interaction between the various elements. For instance, analysis may motivate refinements in the modelling or vice-versa; computation may suggest a problem reformulation or suggest additional needs and/or resources. Design is truely an iterative process.

It is quite common for several feasible designs to exist for a given mechanical system. We choose between these candidate designs according to some criteria or index of merit. That is, we choose an objective for the design, such as minimum cost, maximum reliability, maximum strength ,etc. We seek a combination of

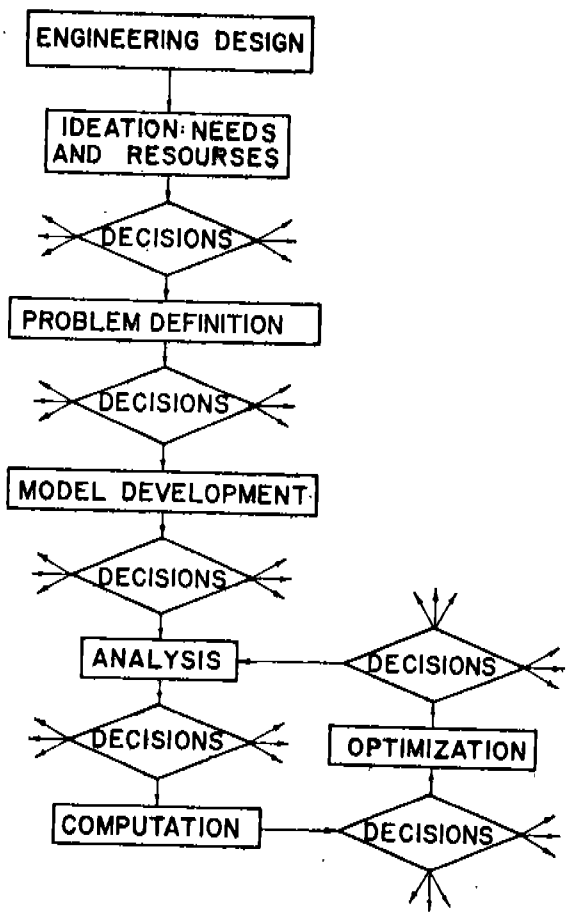

Figure 1: A Morphology of Design

the design variables which gives a design meeting our objective, often in the presence of constraints. These constraints define feasible combinations of the design variables. For instance, in the design of a helical coil spring, we can not allow negative values for the wire diameter. Experience indicates that certain combinations of material, wire diameter, and coil diameter will cause premature failure, and these regions should obviously be avoided. For years machine design has been an art where the number of constraint equations plus the objective was exactly equal to the number of design variables. Very often "rules-of-thumb" were substituted for con-

straints or objectives. Optimization allows us to treat this process more formally, and with increased opportunity for success. Optimization is truely a natural design activity [2].

NONLINEAR PROGRAMMING

Nonlinear programming is a misnomer. A nonlinear program is not a computer program, although digital computers play an important role in the area. A nonlinear program is a formal design situation, much like those discussed earlier, where the objective and constraint functions are nonlinear in the design variables. Formally then, we define a nonlinear program with the following notation:

MINIMIZE:

$$F(\bar{x});\ \bar{x} = [x_1, x_2, x_3, \ldots, x_N]^T,\ \bar{x} \epsilon R^N \qquad (1)$$

SUBJECT TO,

$$\phi_k(\bar{x}) \geq 0,\ k = 1, 2, 3, \ldots, K \qquad (2)$$

AND,

$$\psi_\ell(\bar{x}) \equiv 0,\ \ell = 1, 2, 3, \ldots, L \qquad (3)$$

where

$\bar{x}$ = a column vector of design variables, $x_1, x_2, x_3, \ldots, x_N$; where N is the number of design variables.

$F(\bar{x})$ = a nonlinear scalar function of the design variables called the objective function.

$\phi_k(\bar{x})$ = K nonlinear constraint functions. These functions delimit regions in the design space, because of the inequalities.

$\psi_\ell(\bar{x})$ = L nonlinear constraint functions. These functions vastly reduce the number of candidate designs, because they require specific combinations of the design variables.

We shall call the solution to this nonlinear program, $\bar{x}_M$ where M signifies the number of steps or iterations required to achieve the solution from a prescribed starting point, $\bar{x}_o$. This paper very briefly examines several useful methods or algorithms for finding $\bar{x}_M$. But before the methods let us solidify the formulation by example.

A Simple Example

A Rectangular Storage Bin. Consider the storage bin shown in Figure 2. We wish to design the bin, that is, choose the design variables a, b, and c such that a prescribed volume can be stored with minimum cost. Let us assume that cost is surface area. The nonlinear program is:

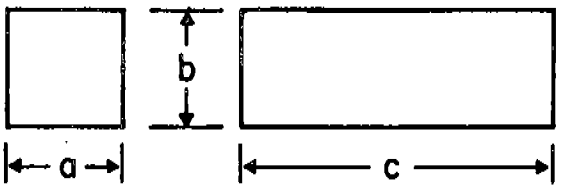

Figure 2: Open-Top Rectangular Storage Bin

MINIMIZE:

$$F(\bar{x}) = 2x_2(x_1 + x_3) + x_1x_3;$$

$$\bar{x} = [a,b,c]^T,\ \bar{x} \epsilon R^3 \qquad (4)$$

SUBJECT TO,

$$\phi_k(\bar{x}) = x_k \geq 0,\ k = 1, 2, 3 \qquad (5)$$

$$\psi_1(\bar{x}) = V^* - x_1x_2x_3 \equiv 0 \qquad (6)$$

Where in this case the ϕ_k's are called nonnegativity restraints. We see that

$$N = 3,\ K = 3,\ L = 1$$

V^* is the prescribed volume which the solution design must store. An interesting variation on this problem occurs in the design of grain storage bins. The stored material has a prescribed angle of repose, θ and the bin sides are allowed to slant to introduce an additional degree of design freedom.

METHODS FOR THE SINGLE VARIABLES CASE

Problems without constraints have been given considerable attention in the past. This is due in part to the fact that unconstrained problems are significantly easier to handle than the constrained variety. Later we shall see that the constrained problem can be reformulated as a sequence of unconstrained problems, hence, the contemporary need for unconstrained methods. Similarly many of these unconstrained methods call for a one dimensional search in N-space. This coupled with the fact that some very interesting problems can be formulated with a single degree of design freedom, motivates the need for dependable one dimensional methods.

Bounding the Minimum

Let us seek the minimum, x* of a function, F(x) of a single variable, x, as shown in

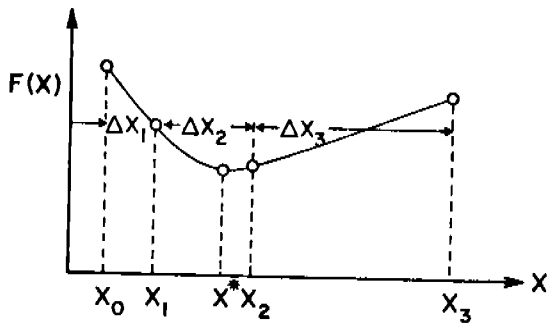

Figure 3: Searching Along a Line

Figure 3. In most useful methods there are two distinct phases to the search; a bounding phase and a refinement phase. In the bounding phase, which we discuss here, two values of x, a and b are sought which bracket the minimum point x*.

This bounding of the minimum can be achieved quite simply assuming an initial estimate of x*, x_o. Let us insist that

$$x_o < x^*, \tag{8}$$

and

$$F(x_o) > F(x^*). \tag{9}$$

Increment x,

$$x_1 = x_o + \Delta x_1 \tag{10}$$

and check if the minimum is being approached;

$$F(x_1) < F(x_o) \tag{11}$$

If (11) is not satisfied either Δx_1 is too large or (8) is violated and we are looking in the wrong direction. Assume then that Δx_1 is sufficiently small and (11) is satisfied. Now let $\Delta x_2 = 2\Delta x_1$ to get

$$x_2 = x_1 + \Delta x_2 \tag{12}$$

Now test to see if the minimum has been bounded, when:

$$F(x_2) > F(x_1). \tag{13}$$

If this is true, we may stop with the knowledge that the minimum is bounded between x_o and x_2. If (13) is violated, generate an additional point x_3,

$$x_3 = x_2 + \Delta x_3 \tag{14}$$

with $\Delta x_3 = 2\Delta x_2$,

and test this point;

$$F(x_3) > F(x_2). \tag{15}$$

In Figure 3 we see tha x_3 would be the last point generated, and x* is bounded between x_1 and x_3. Recognize that only three points must be handled at a time, and recall the procedure in the following algorithmic form.

Bounding a Local Minimum

1. given x_o.
2. calculate $F(x_o)$ = F0.
3. let Δx_1 = a small value.
4. let $x_1 = x_o + \Delta x_1$.
5. calculate $F(x_1)$ = F1
6. is F1 < F0?
   yes: go to 7.
   no: let $\Delta x_1 = \Delta x_1/2$ and go to 4.
7. set $\Delta x_2 = 2\Delta x_1$.
8. let $x_2 = x_1 + \Delta x_2$.
9. calculate $F(x_2)$ = F2.
10. is F2 < F1?
    yes: let F0 = F1, F1 = F2, $x_o = x_1$, $x_1 = x_2$ and go to 7.
    no: stop, x* has been bounded between x_o and x_2.

Remember that this algorithm assumes that Δx_1 is taken in the correct direction from x_o, and a local, not global minimum is sought.

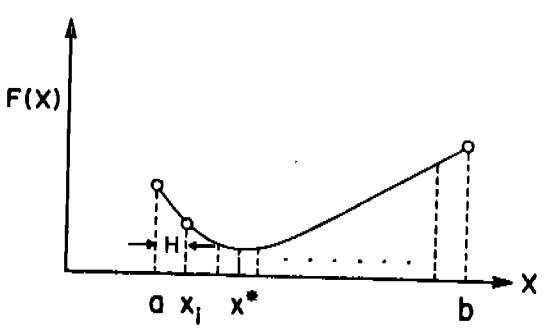

Figure 4: A Bounded Minimum Point

Exhaustive Search [3,4]

Let us assume that x* is bounded from above and below as shown in Figure 4. That is, we are certain that the minimum of F(x), x* is in the interval a to b;

$$a \leq x^* \leq b \tag{16}$$

Therefore, we have an original interval of uncertainty of b - a;

$$IU(0) = b - a \tag{17}$$

Now we wish to reduce this interval of uncertainty to a tolerable level. To do so let us sample F(x) at N equally spaced interior points; $a < x_i < b$, $i = 1,2,3,\ldots.N$, and examine the value of $F(x_i)$ at each point. At the completion of this exhaustive search the interval of uncertainty will be:

$$IU(N) = 2H = 2\left(\frac{b-a}{N+1}\right) \tag{18}$$

where

$$H = x_{i+1} - x_i \tag{19}$$

The fractional reduction of the original interval is;

$$FR(N) = \frac{IU(N)}{IU(O)} \tag{20}$$

therefore

$$FR(N) = 2\left\{\frac{b-a}{N+1}\right\}\left\{\frac{1}{b-a}\right\} = \left\{\frac{2}{N+1}\right\} \tag{21}$$

and for a given FR(N), N is:

$$N = \frac{2-FR(N)}{FR(N)} \tag{22}$$

Let us say that we wish to find $x^* \pm .01$. Then IU(N) = .02 and

$$N = 100\ IU(0) - 1 \tag{23}$$

Therefore, if IU(0) = 2.0 we see that

$$N = 199. \tag{24}$$

to locate x* with a tolerance of .01 using the exhaustive search scheme. We can do better than this, as we see in the next section.

Interval Halfing: A Sequential Search

We see that the exhaustive search scheme causes us to waste much time in unpromising regions of the original interval. A better strategy is to execute a sequence of increasingly refined searches in the most promising interval at each stage. This is the basis of the family of sequential search methods, of which interval halfing is a member.

Let us adopt as a new strategy a sequence of M stages containing N sample points. At the first stage the function will be sampled at N equally spaced points and a new interval of uncertainty will be equal to 2H, where H is the original point spacing. At the second stage only this 2H interval will be considered further, with N additional sample points, and so on. The fractional reduction in the interval of uncertainty after M such stages is:

$$FR(N)_M = \prod_{m=1}^{M} \frac{IU(N)_m}{IU(N)_{m-1}}, \tag{25}$$

where $IU(N)_o = IU(0)$ as before. Therefore we see,

$$FR(N)_M = \frac{IU(N)_M}{IU(0)} = \left\{\frac{2}{N+1}\right\}^M \tag{26}$$

Let η equal the total number of function evaluations;

$$\eta = MN \tag{27}$$

and $$FR(N)_M = \left\{\frac{2}{N+1}\right\}^{\eta/N} \tag{28}$$

so $$\eta = \frac{N \ln(1/FR(N)_M)}{\ln[(N+1)/2]} \tag{29}$$

Now what is the value of N which will make η a minimum for a given value of $FR(N)_M$? Since N must be an integer, N = 3 is the best policy at each stage. So

$$FR(3) = \left\{\frac{2}{3+1}\right\} = \frac{1}{2} \tag{30}$$

Hence, the name interval halfing. Notice that something very interesting happens when interval halfing is employed; the center function evaluation is saved at all stages except the first. Treating the first stage differently;

$$\eta' = \eta + 1 \tag{31}$$

wher η' = total number of function evaluations.

$\eta = 2M.$

M = number of stages.

so; $$\eta' = 2M + 1 \tag{32}$$

and $$FR(3)_{\eta'} = \left\{\frac{1}{2}\right\}^{\left(\frac{\eta'-1}{2}\right)} \tag{33}$$

which gives

$$\eta' = 1 + \left\{\frac{2\ell n(1/FR(3)_{\eta'})}{\ell n 2}\right\} \tag{34}$$

Therefore, to achieve a fractional reduction of .01 as before $\eta' = 15$ function evaluations will be required using the interval halfing scheme, which compares quite favorably with the 199 required before. Additionally, the logic of interval halfing is easily automated, because we deal with only three points at a time.

METHODS FOR THE UNCONSTRAINED CASE

Now let us consider the situation where there are N design variables and no constraints, so

$$K = L \equiv 0 \text{ in (2) and (3).}$$

The situation becomes more complex and much of the thinking of the previous section does not directly extrapolate. Almost all methods have two phases; choice of a direction in N-space and minimization in this direction, choice of a new direction, etc. We shall discuss here a few methods which have proven to be dependable.

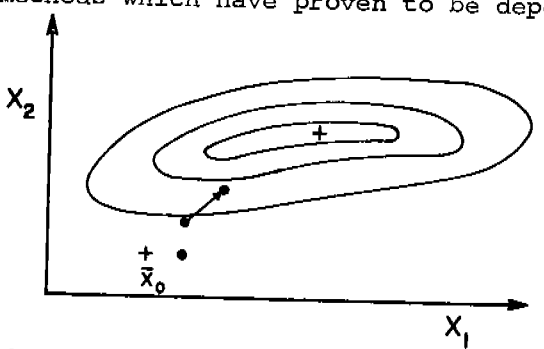

Figure 5: Hooke and Jeeves Method

Hooke and Jeeves [5]

Given a starting point the Hooke and Jeeves method moves toward the minimum point using a sequence of one dimensional trials followed by a pattern move. The general scheme is shown in Figure 5 for a function of two variables. Variable x_1 is incremented a small amount in the positive direction and the function is evaluated, and tested:

$$F(\tilde{x}) < F(\bar{x}) \tag{35}$$

where $$\tilde{x} = [x_1 + \Delta x_1, x_2]^T \tag{36}$$

If (35) is satisfied the trial was a success in the positive direction. If this trial was a failure, i.e., (35) violated; a point in the opposite direction is generated;

$$\tilde{x} = [x_1 - \Delta x_1, x_2]^T, \tag{37}$$

and tested as before. Note that both trials may fail. In any case, the search proceeds to the next variable, x_2, in similar fashion,

$$\tilde{x} = [x_1, x_2 \pm \Delta x_2]^T, \tag{38}$$

and (35) is applied. This process continues until all variables have been incremented, after which a pattern direction is established. One trial is performed in the pattern direction, as shown in the figure, before returning to the univariate searching. The process terminates when no progress can be made for even small Δx's. The simplicity of the method speaks for itself.

Cauchy [6]

Cauchy's method is often called the method of steepest descent, since it employs the local gradient in the move logic. Strickly speaking an entire class of methods exists employing the local gradient. Consider a family of trial points, $\tilde{x}$:

$$\tilde{x} = \bar{x} - \alpha\, g(\bar{x}) \tag{39}$$

where

$\tilde{x}$ = trial point in N-space.

$\bar{x}$ = current location in N-space.

α = step length parameter.

$g(\bar{x}) = [\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \ldots, \frac{\partial F}{\partial x_N}]^T$ = gradient vector.

The simple gradient method results when α is chosen and held as a constant throughout the search. This method is easy to execute, but will surely be slow in approaching the minimum point, since the gradient approaches zero as $\tilde{x}$ approaches the solution, $\bar{x}_M$.

The situation can be improved by choosing α at each step. It would seem reasonable to select α to give the maximum decrease in the function in the direction dictated by $g(\bar{x})$. That is choose α^* such that

$$g(\tilde{x}) = 0. \tag{40}$$

Obviously, this will require a one dimensional search method similar to the previously discussed methods. The exact mechanics of executing a line search in N-space are discussed in the next section. When α^* is so determined we have the method of steepest descent. This method will require fewer steps than the simple gradient method, but is still too slow to be practical for functions usually encountered in design.

Searching a Line in N-Space

The previous and following methods use a common move prescription in N-space.

$$\tilde{\bar{x}} = \bar{x} - \alpha P(\bar{x}) \tag{41}$$

Where $P(\bar{x})$ is the direction of search, and is different for each method. Each method calls for a single variable search along a line in the N-dimensional design space defined by $P(\bar{x})$. Using the previous discussion the following algorithm is useful:

Line Search in N-Space

1. set $\alpha = 0$.
2. calculate $F(x(\alpha)) = F0$.
3. set α = a small value.
4. let $\tilde{\bar{x}} = \bar{x} - \alpha P(\bar{x})$
5. calculate $F(\tilde{\bar{x}}(\alpha)) = F1$
6. is $F1 < F0$?
   yes: go to 7.
   no: let $\alpha = \alpha/2$ and go to 4.
7. set $\alpha = 2\alpha$
8. let $\tilde{\bar{x}} = \bar{x} - \alpha P(\bar{x})$.
9. calculate $F(\tilde{\bar{x}}(\alpha)) = F2$.
10. is $F2 < F1$?
    yes: let $F0 = F1$, $F1 = F2$, and go to 7.
    no: α^* has been bounded, begin interval halfing.

Fletcher-Reeves [7] and Davidon-Fletcher-Powell [8]

These methods use the move prescription given in (41), with the search direction $P(\bar{x})$ given below:

$$P(\bar{x}) = g(\bar{x}), \text{ simple gradient (Cauchy).} \tag{42}$$

$$P(\bar{x}) = g(\bar{x}) + q(\bar{x}), \text{ conjugate gradient (Fletcher-Reeves).} \tag{43}$$

$$P(\bar{x}) = Ag(\bar{x}), \text{ variable metric (Davidon-Fletcher-Powell).} \tag{44}$$

With each of these methods α^* is found directly at each step. The Fletcher-Reeves search direction is chosen to maintain conjugacy of the sequence of directions:

$$q(\bar{x}) = \frac{g(\bar{x})^T g(\bar{x})}{g(\hat{x})^T g(\hat{x})} P(\hat{x}) \tag{45}$$

where we consider three contiguous points in the search, $\hat{x}$, $\bar{x}$, $\tilde{x}$. Therefore, $P(\hat{x})$ = previous search direction. This method possesses the property called quadratic convergence. Given a generalized quadratic function:

$$F(\bar{x}) = a + b^{-T} \bar{x} + \frac{1}{2} x^{-T} H\bar{x} \tag{46}$$

where a = a scalar.

$\bar{b}$ = a column vector of constants.

H = an N x N matrix of constants.

the Fletcher-Reeves method will find the solution, $\bar{x}_M$ in N steps or less from an arbitrary starting point, $\bar{x}_o$.

The Davidon-Fletcher-Powell method changes the search direction by updating the matrix A:

$$A = \hat{A} + \frac{\delta\hat{x}\,\delta\hat{x}^T}{\delta\hat{x}^T \Delta\hat{g}} - \frac{\hat{A}\,\Delta\hat{g}\,\Delta\hat{g}^T\,\hat{A}}{\Delta\hat{g}^T\,\hat{A}\,\Delta\hat{g}} \tag{47}$$

where

$\hat{A}$ = metric at last step.

$\delta\hat{x}$ = previous step = $\bar{x} - \hat{x}$.

$\Delta\hat{g}$ = gradient change = $g(\bar{x}) - g(\hat{x})$.

The method begins with A as any positive definite matrix, such as the identity matrix, I. It is interesting to note that for a quadratic function with A = I at the start, the variable metric method performs exactly as the conjugate gradient method. Also if $A = H^{-1}$ at the start, one step convergence is achieved for a quadratic function, because the Newton method is generated. For a nonquadratic function A approaches H^{-1} in a finite number of steps. Restarting is necessary when the positive definiteness of the A matrix is violated. The method is quadratically convergent. A disadvantage is the need to carry a full N x N matrix throughout the search.

Miele's Memory Gradient Method [8]

Miele's memory gradient method is a generalization of the Fletcher-Reeves conjugate gradient method. The move prescription is:

$$\tilde{\bar{x}} = \bar{x} - \alpha g(\bar{x}) + \beta\,\delta\hat{x} \tag{48}$$

where $\delta\hat{x} = \bar{x} - \hat{x}$ = previous step, and α and β are free parameters to be found at each step. A closer look, than space will allow, at conjugate gradient algorithms is necessary to see the connection with Miele's method. Briefly, there are really two free parameters at each point in the conjugate gradient method. One of these

parameters is selected assuming a quadratic objective function. The other parameter, α, is found directly to help accommodate the nonquadratic effects. In the memory gradient method, no such concession is made to quadratic objective functions. Published numerical results indicate an increase in the rate of convergence over the Fletcher-Reeves method, especially in the initial stages of the search. The move prescription is very simple, but requires a two dimensional search at each generic point, $\bar{x}$. This two dimensional search is a major practical disadvantage of the method.

METHODS FOR THE CONSTRAINED CASE

In the previous discussion we have seen that in most practical engineering design problems the variables are constrained in some way. These constraints normally appear in one of two forms; inequalities or equalities, as given in (2) and (3). Many very practical design problems contain variables which must take on discrete values, and other problems possess variables which exhibit randomness. These two latter complications are of tremendous practical importance, but will be ignored here. We consider here three fundamentally different procedures for handling the constrained problem.

The Penalty Function [9]

Consider the following function

$$P(\bar{x}) = F(\bar{x}) + \varpi(\bar{x}, R, \bar{\phi}, \bar{\psi}) \qquad (49)$$

which we call a generalized weighted penalty function. ϖ is the penalty term, and is constructed so that feasible points (points which satisfy the constraints) are given favored treatment. Also we wish to choose ϖ such that the unconstrained minimum of $P(\bar{x})$ can be easily found, and has a solution which is related to $\bar{x}_M$, the solution to the constrained problem. Choosing ϖ is very much an art, and much work remains to be done in this area. A penalty term which has proven to be useful is:

$$\varpi(\bar{x}, R, \bar{\phi}, \bar{\psi}) = R^2 \sum_{k=1}^{K} \{\frac{1}{\phi_k(\bar{x})}\} + (\frac{1}{R}) \sum_{\ell=1}^{L} \{\psi_\ell(\bar{x})\}^2 \qquad (50)$$

where R, the penalty parameter is chosen as a small positive value, say 1.0, to start and decreased toward zero after each unconstrained minimization. We assume that the starting point, $\bar{x}_o$, is feasible with respect to the inequality constraints, and notice that the weight of the inequality violations are decreased, and the weight of the equality violations are increased after each minimization phase. The major advantage of the penalty approach is the ease of usage. We have taken a relatively complex problem and broken it down into a sequence of more simple unconstrained problems, which can be solved with the previously discussed methods.

Currently a great deal of effort is being expended in the development of penalty functions [10,11], and this author expects methods to appear soon which will be vastly superior to existing methods, especially for large problems.

The Griffith and Stewart Method [12]

This method is formulated to take advantage of the very powerful linear programming methods currently available, such as the Revised Simplex Method [13]. Basically the method works as follows. Given a feasible starting point, $\bar{x}_o$, expand each of the functions, $F(\bar{x})$, $\phi_k(\bar{x})$, $k = 1, 2, 3, \ldots, K$, $\psi_\ell(\bar{x})$, $\ell = 1, 2, 3, \ldots, L$ about the point x_o, retaining only linear terms. Solve the linear program which results, and expand and linearize about the new point, etc. In order to obtain any success with the method, the changes in the design variables must be limited at each phase. The Revised Simplex Method will return a point which resides on at least two of the linearized constraints, and which may not be in the neighborhood of the current point. Therefore, the method must be forced to take small steps. The method has proven to be effective for problems where the objective and constraint functions are nearly linear. The method will often be very slow or not converge at all for very nonlinear problems.

The Constrained Derivative Method [14,15]

This method was developed to handle the nonlinear problem with equality constraints, but can be used for the full problem with the introduction of slack variables in each of the inequality constraints such that they appear as transformed equality constraints. This amounts to a bookkeeping difficulty which we will not consider further here.

Divide the design variables, $\bar{x}$, into two clases called the state and decision variables;

$$\bar{x} = [\bar{y}, \bar{z}]^T \qquad (51)$$

where the decision variables are completely free, and the state variables are slaves used to satisfy the constraints. That is, whenever a decision variable is changed all state variables are adjusted to maintain feasibility. Now, the constrained derivative (also known as the reduced gradient) is the rate of change of the objective function with respect to small changes in the decision variables, with the state variables adjusted to maintain feasibility. The strategy then is to find a set of decision variables which make the constrained derivative zero. Any of the unconstrained methods can be used to achieve this goal. The constrained derivative method is probably the most powerful method currently known for the nonlinear programming problem.

CLOSURE

This survey is not intended to be complete, but does in the author's opinion offer concepts which are useful in a design environment. Many other methods have been proposed, and are worthy of careful consideration. In fact a newcomer to the area can feel like a Prince in his father's harem, with so many methods waiting to be used, which from a practical viewpoint must be applied one at a time. The fact that these methods have practical application has and is being demonstrated in the open literature. The best method for a given problem or class of problems is very much an open question. There is a real need for additional comparative data on the performance of the various methods on problems with varying degrees of difficulty [16, 17, 18]. Careful collection and analysis of this data should lead to even better methods.

There are several geniunely useful textbooks in the area. The ones by Fox [19], Siddall [20], Himmelblau [21], and Beveridge and Schechter [22] are particularly noteworthy. Codes have varying degrees of availability. The package by Siddall [23] called Opti-Sep is user oriented, definitely useful, and available.

REFERENCES

1. Harrisberger, L., Engineersmanship: A Philosophy of Design, Brooks/Cole, 1966, p. 69.

2. Seireg, A., "A Survey of Optimization of Mechanical Design," ASME Transactions: Journal of Engineering for Industry, 1971.

3. Mischke, C. R., An Introduction to Computer-Aided Design, Prentice-Hall, 1968.

4. Wilde, D. J. Optimum Seeking Methods, Prentice-Hall, 1964.

5. Hooke, R. and Jeeves, T. A., "Direct Search Solution of Numerical and Statistical Problems," JACM, 8(2), 212-229, 1961.

6. Cauchy, A., "Method generale pour la resolution des systems d'equations simultanees," Comptes Rendus de L' Academie des Sciences, 25, 536-538, 1847.

7. Fletcher, R. and Reeves, C. M., "Function Minimization by Conjugate Gradients," Computer Journal, vol. 7, no. 2, 1964.

8. Miele, A. and Cantrell, J. W., "Study on a Memory Gradient Method for the Minimization of Functions," JOTA, vol. 3, no. 6, 1969, p. 459.

9. Fiacco, A. V., and McCormick, G. P., Nonlinear Programming: Sequential Unconstrained Minimization Techniques, John Wiley, 1968.

10. Powell, M. J. D., "A Method for Nonlinear Constraints in Minimization Problems," Optimization, Academic Press, 1969, page 283.

11. Schuldt, S. B., Personal Communication, Honeywell Corp., 1974.

12. Griffith, R. E. and Stewart, R. A., "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems," Management Science, vol. 7, 1961, pp. 379-392.

13. Kuo, Numerical Methods and Computers, Addison-Wesley, 1965, Chapter 15, p. 278.

14. Wilde, D. J. and Beightler, C. S., Foundations of Optimization, Prentice-Hall, 1967, Chapter 2.

15. Abadie, J. and Carpentier, J., "Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints," Optimization, Academic Press, 1969, page 37.

16. Colville, A. R., "A Comparative Study of Nonlinear Programming Codes," Proceedings of the Princeton Symposium on Mathematical Programming, Kuhn, H., ed., 1970, pp. 487-501.

17. Eason, E. D. and Fenton, R. G., "A Comparison of Numerical Optimization

Methods for Engineering Design," ASME paper no. 73-DET-17, June, 1973.

18. Ragsdell, K. M., National Science Foundation: Research Initiation Grant, GK-42162, "Optimization Techniques for Engineering Design," April, 1974.

19. Fox, R. L., Optimization Methods for Engineering Design, Addison-Wesley, 1971.

20. Siddall, J. N., Analytical Decision-Making in Engineering Design, Prentice-Hall, 1972.

21. Himmelblau, D. M., Applied Nonlinear Programming, McGraw-Hill, 1972.

22. Beveridge, G. S. G., Schechter, R. S., Optimization: Theory and Practice, McGraw-Hill, 1970.

23. Siddall, J. N., "Opti-Sep: Designers' Optimization Subroutines," McMaster University, Canada, 1971.