ALGORITHMS FOR OPTIMAL DESIGN

by

E. Polak

Memorandum No. UCB/ERL M80/18

25 May 1980

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

ALGORITHMS FOR OPTIMAL DESIGN

E. Polak

Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, California 94720

ABSTRACT. We review in this paper a class of recently developed algorithms for the solution of functional inequalities and of optimization problems with functional inequality constraints, which arise in the context of optimal design. These algorithms fall into the categories of phase I - phase II methods of feasible directions, recursive quadratic programming methods, and outer approximations methods.

1. INTRODUCTION

In this paper we shall survey algorithms for the optimal design of structures subject to dynamic loads or disturbances. In particular, we shall examine algorithms which are applicable to the design of earthquake resistant structures. In the case of such designs one has to cope with constraints of the form

$$\varphi(z,y) \leqq 0 \quad \text{for all } y \in Y \tag{1.1}$$

In (1.1) $z \in R^n$ is the design vector, to be computed, $y \in Y \subset R^p$ may be a scalar, as when it represents time or frequency, or it may be a vector, as when it represents a construction tolerance or an element of a parametrized family of disturbances. Of course, y can also be a vector combining a number of these quantities into one. When the function $\varphi(z,y)$ represents a time response or a frequency response, it is normally found to be differentiable in (z,y). However, when it represents eigenvalues of an operator, it is generally not differentiable, which leads to additional difficulties. We shall mostly concentrate on the case where $\varphi(z,y)$ is continuously differentiable in both variables.

In designing a structure, it may be more important to satisfy the

This research was supported by the National Science Foundation ECS-79-13148 and ENV76-04264.

given specifications than to optimize a particular criterion. We shall therefore consider algorithms for two types of problems. The first is that of finding a feasible vector:

$$FEAS: \quad find\ a\ z \in F \tag{1.2}$$

where

$$F = [z \mid g^j(z) \leq 0, j=1,2,\ldots,p; \\ \varphi^j(z,y_j) \leq 0\ all\ y_j \in Y^j, j=1,2,\ldots,q] \tag{1.3}$$

The g^j are differentiable functions representing simple design constraints and the φ^j model the distributed, i.e., functional, constraints. The second problem that we shall consider is of the form

$$OPT: \quad \min [f(z) \mid z \in F] \tag{1.4}$$

We shall categorize algorithms on the basis of whether the Y are intervals or multidimensional sets and of whether the φ^j are differentiable or not. Since the algorithms we shall describe have a modular structure, it is convenient to present them in the form of Master Algorithms which call any of a set of Subalgorithms. Furthermore, since the essential features of Master Algorithms are preserved when we ignore conventional constraints and assume that there is only one functional constraint and since this results in considerable notational simplification, we shall consider only the simplest form of F, viz.,

$$F = [z \mid \varphi(z,y) \leq 0 \text{ for } all\ y \in Y] \tag{1.5}$$

2. ALGORITHMS FOR COMPUTING FEASIBLE POINTS

2.1. Feasible Directions Type Algorithms.

These algorithms can be used for solving the problem $FEAS$ (1.2) when the sets Y are compact intervals. They differ from each other by the choice of the search direction subalgorithm. We shall now describe two such subalgorithms for the simplified problem (1.5). For every $z \in R^n$, let

$$m(z) = \max [0, \max\{\varphi(z,y) \mid y \in Y\}] \tag{2.1}$$

and for any $\varepsilon > 0$, let

$$Y_\varepsilon(z) = [y \in Y \mid \varphi(z,y) \geq m(z) - \varepsilon \text{ and } y\ is\ a \\ local\ maximizer\ of\ \varphi(z,.)] \tag{2.2}$$

The nature of the physical problems we usually consider justify the following assumption:

A1: For every $z \in R^n$ and for every $\varepsilon > 0$, the set $Y_\varepsilon(z)$ contains only a finite number of points.

The simplest search direction subalgorithm computes the search direction vector $h_\varepsilon(z)$ as a solution of

$$d_\varepsilon(z) = \min\,[\max\{<\nabla_z\varphi(z,y),h>\,|\,y\in Y_\varepsilon(z)\}\,|\,h\in S\,] \qquad (2.3)$$

with S the unit hypercube centered at the origin. Thus, in view of A1, (2.3) is seen to be a finite linear program. A better direction search subalgorithm computes the search direction $h_\varepsilon(z)$ as a solution of

$$d_\varepsilon(z) = \min[\,|h|\;|\;h\in co\,\{\nabla_z\varphi(z,y),y\in Y_\varepsilon(z)\}] \qquad (2.4)$$

which we recognize as a finite quadratic program. In [P7] we find a detailed discussion of such subalgorithms, which were first introduced in [P1,G1] for optimization problems with functional constraints. For the case of ordinary constraints, the reader is referred to [P7].

Master Algorithm 2.1 [P5]

Parameters: $\varepsilon_0>0$, a, b, $c\in(0,1)$

Data: $z_0\in R^n$

Step 0: Set i = 0.

Step 1: Set $\varepsilon=\varepsilon_0$.

Step 2: If $z_i\in F$ stop. Else, compute $Y_\varepsilon(z_i)$, and use a search direction finding subalgorithm to compute $d_\varepsilon(z_i)$ and $h_\varepsilon(z_i)$.

Step 3: if $d_\varepsilon(z_i) > -\varepsilon$, set $\varepsilon = a\times\varepsilon$ and go to step 2.

Step 4: Compute the largest step size $s_i\in\{1,b,b^2,\ldots\}$ such that

$$m(z_i+s_i h_\varepsilon(z_i)) < -s_i c\,\varepsilon \qquad (2.5)$$

Step 5: Set $z_{i+1} = z_i+s_i h_\varepsilon(z_i)$, set $i=i+1$ and go to step 1 (or step 2).

Master Algorithm 2.1 has the following property (see[G1]):

If the problem FEAS (1.2) is such that $d_0(z)\neq 0$ for all $z\in F$, and the sequence $\{z_i\}$ constructed by Master Algorithm 2.1 is bounded, then Master Algorithm 2.1 finds a $z\in F$ in a finite number of iterations.

In the form stated, Master Algorithm 2.1 cannot be programmed since it lacks instructions for approximating m(z). Implementable versions of this algorithm have also been presented in [P1,G1], which improve adaptively, on the basis of tests, the precision with which m(z) is evaluated.

2.2. Newton's method for Infinite Systems of Inequalities.

This method also can only be used when the sets Y_j are compact intervals and assumption A1 holds. In addition, we must require that all functions be three times continuously differentiable. In this case, the problem (1.5) can be replaced by a problem specified by the finite set of inequalities:

$$find\ a\ z\in F = [z\,|\,\varphi(z,y)\leq 0, for\ all\ y\in Y_\varepsilon(z)] \qquad (2.6)$$

where $\varepsilon > 0$ is arbitrary. It is shown in [M4] that, under some additional, rather technical, assumptions on $\varphi_y(z,y)$ and $\varphi_{yy}(z,y)$, with $y \in Y_\varepsilon(z)$, one can construct a locally, quadratically convergent version of Newton's method, for solving (2.6), as follows: Let $v(z) \in R^n$ be defined by

$$v(z) = arg \min [|v| \; | \varphi(z,y) + <\nabla_z \varphi(z,y), v> \leqq 0 \text{ for } all \; y \in Y] \quad (2.7)$$

Then, given $z \in F$, set

$$z_{i+1} = A_1(z_i) = z_i + v(z_i) \quad (2.8)$$

Note that because $Y_\varepsilon(z)$ is finite, by assumption, the quadratic program (2.7) does not present any exceptional difficulties and can be solved by standard QP codes. It is also possible to use the L_∞ norm in (2.7), which then makes this problem an LP. Since the radius of convergence of the process defined by (2.8) may be quite limited, it was proposed in [M4] to stabilize it by switching over to the Master Algorithm 2.1, when a certain test fails, as follows. First note that when the return from Step 5 to Step 1 is used, Master Algorithm 2.1 becomes one step and the ε accepted in Step 4 is a function of z_i only, so we write it as $\varepsilon(z_i)$. Hence, let $A_2(.)$ be the map from R^n into all subsets of R^n, defined by Master Algorithm 2.1, with $h_\varepsilon(z)$ computed in Step 1 and s computed in Step 3 of Master Algorithm 2.1 so that $A_2(.)$ has the form

$$A_2(z) = z + sh_\varepsilon(z) \quad (2.9)$$

Master Algorithm 2.2

Parameters: $\varepsilon > 0$, $d \in (0,1)$, $K > 1$

Data: $z \in R^n$.

Step 0: Set i = 0, j = 0.

Step 1: If $m(z_i) < 0$, stop.

Step 2: If a solution $v(z_i)$ of (2.7) exists and

$$|v(z_i)| < Kd^j \quad (2.10)$$

set

$$z_{i+1} = A_1(z_i) \quad (2.11a)$$

Else, set

$$z_{i+1} = A_2(z_i) \quad (2.11b)$$

Step 3: Set i = i + 1 and go to step 1.

We note that as long as the Newton process (2.8) is well defined and shows signs of at least linear convergence, according to (2.10), we use it to define z_{i+1}, otherwise, we switch to Master Algorithm 2.1. It is shown in [M4] that (under suitable assumptions) Master Algorithm 2.2 has the following property:

If the sequence $\{z_i\}$ constructed by Master Algorithm 2.2 is infinite and

bounded, then there exists an i_0 such that (2.11a) holds for all $i>i_0$, and $z_i \to z \in F$, quadratically (for some $z \in F$).

In the form stated, Master Algorithm 2.1 is conceptual, since it does not specify approximation rules form computing the sets $Y_\epsilon(z_i)$. The reader will find an implementable form discussed in detail in [M4].

2.3. Outer Approximations Algorithms.

We now drop the requirement that the Y^j be intervals. They may be any compact subsets of a Euclidean space. Outer approximations algorithms decompose the problem *FEAS*, which contains infinitely many inequalities, into an infinite sequence of problems of the form

$$FEAS(k): \quad find\ a\ z \in F_k$$

where

$$F_k = [z \mid g^j(z) \leq 0,\ j=1,2,\ldots,p;\ \varphi^j(z,y_j) \leq 0,\ \text{for all } y_j \in Y_k^j,\ j=1,2,\ldots,q] \tag{2.12a}$$

or, in the simplified framework of (1.5),

$$F_k = [z \mid \varphi(z,y) \leq 0 \text{ for all } y \in Y_k] \tag{2.12b}$$

We see that the sets F_k are described by finite sets of inequalities which can be solved very rapidly by algorithms such as those described in [M1,P3]. When certain simple assumptions are satisfied, these algorithms need only a finite number iterations to to solve FEAS(k) and are called as subalgorithms by outer approximations master algorithms.

The simplest outer approximations algorithms are described by Eaves and Zangwill in [E1]. In these algorithms, given a problem FEAS(k), with a finite set Y_k , first one computes a $z_k \in F_k$ and then one computes a

$$y_k \in arg\ \max [\varphi(z_k,y) \mid y \in Y_k] \tag{2.13a}$$

The next problem, FEAS(k+1) is then constructed by setting

$$Y_{k+1} = Y_k \cup \{y_k\} \tag{2.13b}$$

It is easy to show that any accumulation point z of an infinite sequence $\{z_k\}$, constructed in this manner, must be in F. However, since the cardinality of the Y_k grows relentlessly, one may not be able to compute too many elements of such a sequence. Because of this, Eaves and Zangwill have proposed a scheme for constructing the Y_k , which periodically flushes out a number of elements from Y_k . Since their scheme had a number of shortcoming, including the fact that it was nonimplementable and highly scaling dependent, Mayne, Polak and Trahan [M2] have proposed a more sophisticated method which was implementable and which included features to minimize scaling dependence. However, it still retained at least one bad feature in [E1], viz. that convergence could be claimed not for accumulation points of the whole sequence $\{z_k\}$, but only for those of the subsequence at which the Y_k were flushed out. This

difficulty, as well as a few other ones, were overcome by the scheme proposed by Gonzaga and Polak [G2], which we now state for the simplified problem (1.5).

Master Algorithm 2.3 [G2,P5].

Parameters: K, L.

Data: $Y_0 \subset Y$.

Step 0: Set k = 0.

Step 1: Compute a $z_k \in F_k$.

Step 2: Compute a

$$y \in arg \max [\varphi(z_k,y) \mid y \in Y] \tag{2.14}$$

Step 3: If $m(z_k) = 0$, stop. Else, set

$$Y_{k+1} = [y_i \mid \varphi(z_i,y_i) \geq K \{1/(1+i)^{1/L} - 1/(1+k)^{1/L}\},\ i=0,1,\ldots,k] \tag{2.15}$$

Step 4: Set k = k + 1 and go to step 1.

An examination of formula (2.15) shows that a point y_i will always be included in Y_i and, possibly, in a few subsequent Y_k . However, as k increases, the right hand side of the inequality in (2.15) also increases and, in most likelyhood, y_i will be dropped from the Y_k after a few retentions. It is shown in [G2] that, *any accumulation point z of a sequence* $\{z_i\}$, *constructed by Master Algorithm 2.3, is in F.*

Again, as stated, Master Algorithm 2.3 is only conceptual since no rule is given for approximating $m(z_k)$. An implementable version can very easily be deduced from the schemes given in [G2], where it is shown that one merely needs to evaluate $m(z_k)$ with progressively greater precision as k increases.

2.4. Nondifferentiable Constraints.

When the functions φ^j are nondifferentiable, none of the above algorithms apply. However, when they are at least locally Lipschitz continuous, it becomes possible to make use of some of the known results in nondifferentiable optimization [C1,M5,L1, P6]. In particular, when the φ^j are eigenvalues of an operator and the Y are intervals, the problem becomes reasonably tractable, by an algorithm, essentially of the form of Master Algorithm 2.1, which uses vectors belonging to an ' ε-bundle' of generalized gradients [C1], instead of gradients in (2.4). The construction of these vectors is quite complicated and we refer the interested reader to [P6], where such an algorithm is described in detail.

This concludes our discussion of algorithms for solving inequalities which arise in the context of engineering design and we now turn to optimization.

3. ALGORITHMS FOR COMPUTING OPTIMAL POINTS

3.1. Phase I - Phase II Feasible Directions Algorithms.

These algorithms can be used for solving the problem OPT (1.4) when the Y are compact intervals and assumption A1 is satisfied. Just as the algorithms described in the preceding section, they differ from one another only by the choice of the search direction subalgorithm. There is a direct relationship between the direction subalgorithms used in Sec. 2a and the ones we are are about to describe. Thus, (2.3) becomes

$$d_\varepsilon(z) = \min[max\{<\nabla f(z),h> ; <\nabla_z \varphi(z,y),h>, y \in Y_\varepsilon(z)\} | h \in S] \quad (3.1)$$

with $Y_\varepsilon(z)$ and S as in (2.3). Similarly, (2.4) becomes:

$$d_\varepsilon(z) = \min[\|h\| \mid h \in co\{\nabla f(z), \nabla_z \varphi(z,y), y \in Y_\varepsilon(z)\}] \quad (3.2)$$

It is shown in [P1,G1] that $d_0(z) = 0$ is an optimality condition which is equivalent to the F. John optimality condition. Hence, we call the functions $d_\varepsilon(z)$optimality functions. Although the zeros of the two optimality functions defined by (3.1) and (3.2) are identical, they do lead to algorithms with different computational properties. The one defined by (3.2) has certain self scaling properties which result in a superior algorithm.

Master Algorithm 3.1 [G1].

Parameters: $\varepsilon>0$, a, b, c, $\in (0,1)$.

Data: $z_0 \in R^n$.

Step 0: Set i = 0.

Step 1: Set $\varepsilon = \varepsilon_0$.

Step 2: If $d_0(z_i) = 0$, stop. Else, use a search direction finding subalgorithm to compute $Y_\varepsilon(z_i)$, $d_\varepsilon(z_i)$, and $h_\varepsilon(z_i)$.

Step 3: If $d_\varepsilon(z_i) > -\varepsilon$, set $\varepsilon = a \times \varepsilon$ and go to step 2.

Step 4: Compute the largest step size $s_i \in \{1, b, b^2, ...\}$ such that if $z_i \in F$, then

$$z_i + s_i h_\varepsilon(z_i) \in F \quad (3.3a)$$

and

$$f(z_i + s_i h_\varepsilon(z_i)) - f(z_i) < -s_i c \varepsilon \quad (3.3b)$$

if$z \notin F$, then

$$m(z_i + s_i h_\varepsilon(z_i)) - m(z_i) < -s_i c \varepsilon \quad (3.4)$$

Step 5: Set $z_{i+1} = z_i + s_i h_\varepsilon(z_i)$, set $i = i+1$ and go to step 1 (or Step 2).

We note that Master Algorithm 3.1 concentrates at first on finding a feasible point. Once such a point is found, the remainder of the sequence is feasible. Referring to [G1], we find that Master Algorithm 3.1 has the

following property:

If the problem OPT is such that $d_\varepsilon(z) < 0$ for all $z \notin F$, then any accumulation point z of a sequence $\{z_i\}$ constructed by the Master Algorithm 3.1 is feasible, i.e., $z_i \in F$, and satisfies the F. John condition of optimality, i.e., $d_0(z)=0$.

Again, since no rule for specifying approximations to $m(z)$ and $Y_\varepsilon(z)$ are given, Algorithm Model 3.1 is only conceptual. An implementable version with the same convergence properties will be found in [P5,G1].

3.2. Recursive Quadratic Programming Algorithms.

At present, there is strong sentiment in the mathematical programming community that recursive quadratic programming algorithms, such as those described in [W1,R1,H1,P8,P9], which have evolved from Wilson's method [W1], offer substantial advantages over other constrained optimization algorithms. We recall that Wilson's method is an appropriate form of Newton's method for solving the equations and inequalities in the Kuhn-Tucker conditions.At present, there is no recursive quadratic programming algorithm for solving OPT in the literature. However, based on what we have seen in Sec. 2.2, we feel that for the case where the sets Y^j are compact intervals and assumption A1 holds, one can construct an RQP type algorithm for solving OPT, as follows. First, we observe that when we compute $d_\varepsilon(z)$ by (3.2), the quadratic program returns a set of positive weights: $u(z,f), u(z,y), y \in Y_\varepsilon(z)$, summing to unity, such that

$$h_\varepsilon(z) = u(z,f)\,\nabla f(z) + \sum_{y \in Y_\varepsilon(z)} u(z,y)\nabla_z \varphi(z,y) \tag{3.5}$$

Assuming that $u(z,f) \neq 0$, we can rescale these weights, which are potential F. John multipliers, by dividing them all by $u(z,f)$ to obtain a set of potential Kuhn-Tucker multipliers which we shall denote by $u(z,y,\varepsilon)$, to make their dependence on ε more obvious. These multipliers can now be used to compute the corresponding second derivative matrix

$$H(z,\varepsilon) = \partial^2 f(z)/\partial z^2 + \sum_{y \in Y_\varepsilon(z)} u(z,y,\varepsilon)\,\partial^2 \varphi(z,y)/\partial z^2 \tag{3.6}$$

In our interpretation, given any $\varepsilon > 0$, and z_i, Wilson's method, applied to the problem

$$\min [f(z_i) \mid \varphi(z_i,y) < 0,\ y \in Y_\varepsilon(z_i)] \tag{3.7}$$

requires us to compute $v(z_i)$ as a solution of

$$\min [\langle \nabla f(z_i), v \rangle + (1/2)\langle v, H(z_i,\varepsilon)v \rangle \mid \varphi(z_i,y) + \langle \varphi(z_i,y), v \rangle \leq 0 \text{ for all } y \in Y_\varepsilon(z_i)] \tag{3.8}$$

and to set

$$z_{i+1} = A_1(z_i) = z_i + v(z_i) \tag{3.9a}$$

On the basis of the analysis referred to in Section 2.2, we can anticipate that, under suitable assumptions, the algorithm defined by (3.8) will be

locally, quadratically, convergent. Let us denote the construction defined by the steps of Master Algorithm 3.1, with a return from Step 5 to Step 1, by

$$z_{i+1} = A_2(z_i) \tag{3.9b}$$

Next, we denote by $\varepsilon(z_i)$ the value of ε which is accepted by Master Algorithm 3.1 at z_i, i.e., which permits passage from Step 3 to Step 4.

We can now follow the scheme in Master Algorithm 2.1 to obtain a globally convergent version of an RQP type method for (3.7). Note that this method requires us to solve two quadratic programs per iteration.

Master Algorithm 3.2.

Parameters: d $\in(0,1)$, K > 1.

Data: $z \in R^n$.

Step 0: Set i = 0, j = 0.

Step 1: Compute $\varepsilon(z_i)$, $Y_\varepsilon(z_i)$, and $H(z_i,\varepsilon(z_i))$ by means of (3.2).

Step 2: If a solution $v(z_i)$ of (3.8) exists for $\varepsilon=\varepsilon(z_i)$, and

$$\|v(z_i)\| \leq Kd^j \tag{3.10}$$

set

$$z_{i+1} = A_1(z_i) \tag{3.11a}$$

Else, set

$$z_{i+1} = A_2(z_i) \tag{3.11b}$$

Step 3: Set i = i + 1 and go to Step 1.

Thus, just as in the case of Master Algorithm 2.2, we accept Wilson's method if it shows signs of converging, at least linearly, according to (3.10), and we revert to Master Algorithm 3.1 otherwise. On the basis of the results in [P8] and [M4], we conjecture as follows:

If (i) all functions are three times continuously differentiable , (ii) all Kuhn-Tucker points satisfy second order sufficiency conditions, (iii) the conditions on $\varphi_y(z,y)$ required for Master Algorithm 2.2 are satisfied,

Then(i) all accumulation points of a sequence $\{z_i\}$, constructed by Master Algorithm 3.2, are Kuhn-Tucker points. (ii) if the sequence $\{z_i\}$ constructed by Master Algorithm 3.2 is bounded, then it converges quadratically to a Kuhn-Tucker point.

Master Algorithm 3.2 can be made implementable by using the same rules as those for Master Algorithms 2.2 and 3.1, and hence implementability is not a source of difficulty. What may prove to be a source of serious difficulty is the need to evaluate second derivatives. Fortunately, as can be seen from [H1,P8,P9] there are a number of first order schemes available for approximating $H(z,\varepsilon(z))$. These schemes make use of either secant or quasi-Newton formulas.

3.3. Outer Approximations Algorithms.

We now drop the requirement that the Y^j be intervals (see 2.3) and allow them to be any compact subsets of a Euclidean space. We shall describe the scheme presented in [G1]where the decomposition of the problem OPT into a sequence of problems

$$OPT(k): \quad \min [f(z) \mid z \in F_k] \tag{3.12}$$

is carried out essentially in the same manner as in solving inequalities, with one important exception. We can solve a problem FEAS(k) in a finite number of iterations, but we cannot solve OPT(k) in a finite number of iterations (or find a stationary point, for that matter). Hence, we must specify in what sense should the solutions to OPT(k) be approximated. Such a specification is incorporated in the master algorithm, below, which calls an ordinary constrained optimization subalgorithm to "solve" OPT(k). Without essential loss of generality, we again state the algorithm in terms of the simplified problem OPT, with F given by (1.5). First, given Y_k, a finite subset of Y, for any $z \in \mathbb{R}^n$, we define

$$m(z, Y_k) = \max[0, \max\{\varphi(z, y) \mid y \in Y_k\}] \tag{3.13a}$$

Next, for any $z \in \mathbb{R}^n$ and $\varepsilon > 0$, we define

$$Y_{k,\varepsilon}(z) = [y \in Y_k \mid \varphi(z, y) > m(z, Y_k - \varepsilon] \tag{3.13b}$$

Finally, we define

$$d(z, Y_k) = \min [\|h\| \mid h \in co\{\nabla f(z), \nabla_z \varphi(z, y), y \in Y_k(z)\}] \tag{3.13c}$$

We recognize that $d(z, Y_k)=0$ is a necessary condition of optimality for OPT(k), which, as we have already mentioned, is equivalent to the F. John condition.

Master Algorithm 3.3

Parameters: ε_0 ,K, L > 0.

Data: $Y_0 \subset Y$

Step 0: Set k = 0, set $\varepsilon = \varepsilon_0$.

Step 1: Solve OPT(k) to the extent of computing a z_k such that

$$d(z_k, Y_k) > -\varepsilon \tag{3.14a}$$

and

$$m(z_k, Y_k) < \varepsilon \tag{3.14b}$$

Step 2: Compute a

$$y \in arg \max [\varphi(z_k, y) \mid y \in Y] \tag{3.15}$$

Step 3: If $m(z_k, Y) = 0$, stop. Else, set

$$Y_{k+1} = [y_j \mid \varphi(z_j, y_j) > K\{1/(1+j)^{1/L} - 1/(1+k)^{1/L}\},\ j=0,1,2,\ldots,k \tag{3.16}$$

Step 4: Set $\varepsilon = \varepsilon/2$, set k = k + 1 and go to Step 1.

The only part of Master Algorithm 3.3 which is not implementable is the computation of the y_k and $m(z_k, Y)$. In [G2], we find a simple rule which states that all that is needed is that these computations be carried out with progressively greater and greater precision, as the computation progresses (in the same manner as for OPT(k)). As far as convergence is concerned, we find in [G2] the following result:

(i) If the sequence $\{z_k\}$ constructed by Master Algorithm 3.3 is finite, then its last element is in F and satisfies the F. John condition of optimality for OPT.

(ii) If the sequence $\{z_k\}$ constructed by Master Algorithm 3.3 is infinite, then all its accumulation points are in F and satisfy the F. John optimality condition for OPT.

Finally, it should be pointed out that the rule for constructing Y_{k+1}, as given in (3.16) and also in (2.15) can be substantially modified by using double subscripted sequences $\{\varepsilon_{j,k}\}$ in the right hand side of the inequalities in (2.15) and (3.16), as explained in [G1], with the one used in this paper being only one example of such a sequence.

3.4. Nondifferentiable Optimization Algorithms.

At present, it appears that the entire arsenal of nondifferentiable optimization algorithms which apply to OPT, consists of those described in [P2] and [P6]. These algorithms make use of outer approximations master algorithms and of either ordinary or nondifferentiable optimization subalgorithms. The lack of space prevents us from describing them here. The interested reader is referred to the above mentioned journal article and report.

4.CONCLUSION

We have given in this paper a brief survey of a class of algorithms which are applicable to optimal design. Some of these algorithms, such as feasible directions, have been used by us for a number of years and have been found to be quite reliable, though somewhat expensive. The newer algorithms, such as RQP are in the process of being programmed up for testing. Given their excellent behavior on ordinary mathematical programming problems, we expect that they will eventually prove to be a most valuable tool in optimal design, as well.

REFERENCES

C1. Clarke,F. M., "Generalized gradients and applications", Trans. Amer.Math. Soc., 205, pp 247-262, 1975.

E1. Eaves,B. C.and W. I. Zangwill, "Generalized cutting plane algorithms", SIAM J. Control, Vol. 9, No. 4, pp 529-542 ,1971.

G1. Gonzaga G., E. Polak and R. Trahan, "An Improved Algorithm for optimization problem with functional inequality constraints", IEEE Trans. Vol AC-25, No. 1, pp 49-54, 1980.

G2. Gonzaga, C. and E. Polak, "On constraint dropping schemes and optimality functions for a class of outer approximations algorithms", SIAM, J. Control and Optimization, Vol. 17, No.4, pp. 477-493, 1979.

H1. Han, S. P. "Dual variable metric algorithms for constrained optimization", SIAM J. Control and Optimization, Vol. 11, pp. 546-566, 1977.

L1. Lemarechal, C. "An extension of Davidon methods to nondifferentiable optimization problems", in Nondifferentiable Optimization, M. L. Balinski and P. Wolfe eds. Mathematical Programming study 3, Amsterdam, North Holland, pp. 95-109, 1965.

M1. Mayne, D. Q., E. Polak and A. J. Heunis, "Solving nonlinear inequalities in a finite number of iterations", Publication 79/3, Department of Computing and Control, Imperial College, London, 1979. To appear in JOTA.

M2. Mayne, D. Q., E. Polak and R. Trahan, "An outer approximations algorithm for computer aided design problems", JOTA, Vol. 28, No. 3, pp 331-352, 1979.

M3. Mayne,D. Q. and E. Polak, "A superlinearly converging algorithm for constrained optimization problems", Publication No. 78/52, Department of Computing and Control, Imperial College, London, 1878.

M4. Mayne,D. Q. and E. Polak, "A quadratically convergent algorithm for solving infinite dimensional systems of inequalities" University of California, Electronics Research Lab. Memo. No. UCB/ERL M80/11, 1980.

M5. Mifflin, R. "Semismooth and semiconvex functions in constrained optimization",SIAM J. Control and Optimization, Vol. 15, pp. 959-972, 1979.

P1. Polak,E. and D. Q. Mayne, "An optimization algorithm with functional inequality constraints", IEEE Trans. Vol. AC-21, No. 2 pp. 184-193, 1976.

P2. Polak, E. "An implementable algorithm for the optimal design centering, tolerancing and tuning problem", University of California, Electronics Research Lab. Report No. UCB/ERL M79/33, 1979.To appear in JOTA.

P3. Polak, E. and D. Q. Mayne, "On the finite solution of inequalities", IEEE Trans. Vol. AC-24, No. 3, pp 443-445, 1979.

P4. Polak, E. and D. Q. Mayne, "On the solution of singular value inequalities", University of California Electronics Research Lab. Memo. No. UCB/ERL M80/8, 1980.