

## **Copyright Warning & Restrictions**

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be “used for any purpose other than private study, scholarship, or research.” If a user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of “fair use” that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

**Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation**

Printing note: If you do not wish to print this page, then select “Pages from: first page # to: last page #” on the print dialog screen

The Van Houten library has removed some of the personal information and all signatures from the approval page and biographical sketches of theses and dissertations in order to protect the identity of NJIT graduates and faculty.

## INFORMATION TO USERS

This material was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.
2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.
3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in "sectioning" the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again — beginning below the first row and continuing on until complete.
4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from "photographs" if essential to the understanding of the dissertation. Silver prints of "photographs" may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.
5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

**University Microfilms International**

300 North Zeeb Road  
Ann Arbor, Michigan 48106 USA  
St. John's Road, Tyler's Green  
High Wycombe, Bucks, England HP10 8HR

77-22,452

MORADI, Jacob Yaghoub, 1950-  
A FAST BOUNDARY TRACKING ALGORITHM FOR  
CONSTRAINED NONLINEAR MATHEMATICAL  
PROGRAMMING PROBLEMS.

New Jersey Institute of Technology,  
D.Eng.Sc., 1977  
Engineering, mechanical

**Xerox University Microfilms**, Ann Arbor, Michigan 48106

A FAST BOUNDARY TRACKING ALGORITHM FOR CONSTRAINED  
NONLINEAR MATHEMATICAL PROGRAMMING PROBLEMS  
BY  
JACOB YAGHOUB MORADI

A DISSERTATION  
PRESENTED IN PARTIAL FULFILLMENT OF  
THE REQUIREMENTS FOR THE DEGREE  
OF  
DOCTOR OF SCIENCE IN MECHANICAL ENGINEERING  
AT  
NEW JERSEY INSTITUTE OF TECHNOLOGY

This dissertation is to be used only with due regard to the rights of the author. Bibliographical references may be noted, but passages must not be copied without permission of the College and without credit being given in subsequent written or published work.

Newark, New Jersey  
1977

# DEDICATION

*In Memory of*

*My Father,*

DAVID MORADI

## ABSTRACT

A fast search algorithm for the solution of nonlinear mathematical programming optimization problems is presented in this thesis. A gradient search procedure is combined with a "Boundary Tracking"(BT) method using the feasible direction finding method of Zoutendijk for generating a feasible starting direction along the feasible-infeasible boundary.

The algorithm is applied to the minimum weight design of submersible, circular, cylindrical shells reinforced by equally spaced "T" type frames. This problem had produced algorithm failure in two earlier studies and was only recently solved by the Direct Search-Feasible Direction Algorithm (DSFD) which was shown by recent comparison studies to be among the fastest and most reliable mathematical programming methods available. The BT procedure was found to be substantially faster than DSFD, producing a solution with about one-eighth the effort required by DSFD.

In a general comparison study a code based on the BT algorithm was compared with twenty other codes representing most of the popular numerical optimization methods on ten test problems. These problems are such that majority of the codes tested failed to solve more than half of them. The new code proved superior to all others in overall generality and efficiency. It solved all problems and was the fastest code on the constrained problems.

APPROVAL OF DISSERTATION  
A FAST BOUNDARY TRACKING ALGORITHM FOR CONSTRAINED  
NONLINEAR MATHEMATICAL PROGRAMMING PROBLEMS  
BY  
JACOB YAGHOUB MORADI  
FOR  
DEPARTMENT OF MECHANICAL ENGINEERING  
NEW JERSEY INSTITUTE OF TECHNOLOGY

BY

FACULTY COMMITTEE

APPROVED: \_\_\_\_\_ CHAIRMAN  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

NEWARK, NEW JERSEY

JUNE, 1977



## ACKNOWLEDGMENT

The author wishes to express his appreciation to his advisor, Dr. M. Pappas, for his help and guidance in the preparation of this thesis. Also, he is grateful to Dr. H. Herman for his aid and advice.

This work was supported by the Office of Naval Research, Contract No. ONR-N000-14-75-C-0987 and by the Office of the Dean of Research at New Jersey Institute of Technology.

## TABLE OF CONTENTS

	Page
ABSTRACT OF THE THESIS	iii
ACKNOWLEDGMENT	v
LIST OF FIGURES	viii
LIST OF TABLES	ix
NOMENCLATURE	x
CHAPTER 1 - INTRODUCTION	1
1.1 Optimization in Design	1
1.2 General Strategy of Mathematical Programming	4
1.3 Motivation and Structure	6
CHAPTER 2 - DESCRIPTION OF GENERAL STRATEGY OF THE BOUNDARY TRACKING ALGORITHM	10
2.1 Introduction	10
2.2 Location of the Behavior Boundary	10
2.3 Movement Along the Behavior Boundary	15
2.4 Search Termination Description	18
CHAPTER 3 MATHEMATICAL DEVELOPMENT	21
3.1 Boundary Location	21
3.1.1 Identification of Proper Constraints	21
3.1.2 The Boundary Locating Method	22
3.2 Mathematical Representation of General Strategy	26
3.2.1 Movement to the Boundary, Starting Point in the Feasible Region	28

## TABLE OF CONTENTS (Cont'd.)

	Page
3.2.2 Movement to the Boundary, Starting Point in the Infeasible Region	31
3.2.3 Initiation of Movement Along the Boundary	31
3.2.4 Movement Along the Boundary	34
CHAPTER 4 - DESIGN EXAMPLE	42
4.1 Problem Statement	42
4.2 Problem Formulation	43
4.2.1 Objective Function	43
4.2.2 Constraint Equations	44
4.3 Discussion of Results	50
4.3.1 Code Description	50
4.3.2 Code Application	51
CHAPTER 5 - COMPARISON STUDY	56
CHAPTER 6 - CONCLUSION	68
LIST OF REFERENCES	70

## LIST OF FIGURES

	Page
Figure 1: General Strategy of the Boundary Tracking Method	11
Figure 2: Block Diagram of the General Strategy	12
Figure 3: Block Diagram of the Termination Procedure	20
Figure 4: Selection of a Proper Constraint	23
Figure 5: The Boundary Locating Method	27
Figure 6: Flow Chart of the Procedure for Movement to the Boundary	29
Figure 7: Flow Chart of the Procedure for Movement Along the Boundary	35
Figure 8: Flow Chart of the Termination Procedure	39
Figure 9: Typical Shell Cross Section	45

## LIST OF TABLES

	Page
Table 1: Synthesis path for the shell design problem using DSFD method	53
Table 2: Synthesis path for the shell design problem using the Boundary Tracking method	54
Table 3: Code Descriptions	57
Table 4: Performance of Optimization codes	58
Table 5: Relative ranking of Optimization codes	60
Table 6: Ranking of Optimization codes in Problems with Behavior Constraints	65

## NOMENCLATURE

$A$	= cross sectional area of the frame
$A_j$	= number of active behavior constraints
$B_k(x)$	= behavior function
$b$	= web thickness
$b'$	= flange thickness
$c$	= distance from midplane of the shell to the surface of the frame
$C_1$	= an arbitrary large constant
$C_2$	= maximum flange to plating thickness ratio
$d$	= distance from the midplane of the hull plating to the neutral axis of the frame
$E$	= tensile modulus
$EI_f$	= bending stiffness of a frame
$e$	= eccentricity
$e_1, e_2, e_3$	= arbitrary small positive constraints
$f, f(x)$	= objective function
$f_a, \bar{f}_a$	= efficiency criteria
$G$	= shear modulus
$GJ_f$	= torsional stiffness of a frame
$g(x), g_j(x)$	= constraint function and the $j$ th constraint function
$g^*(x)$	= the smallest proper constraint
$g(t)$	= the constraint $g^*(x)$ as a function of a single variable $t$
$H_E, H_M$	= frame deflection parameters [see equations 3.11 - 3.12]
$h$	= web height
$I$	= number of variables

$J$	= number of constraint equations
$K$	= number of proper constraints
$k_j$	= arbitrary small positive number
$L'$	= distance between frames
$L_k$	= lower limit on behavior function
$L_s$	= length of the shell
$m$	= axial wave number
$N_a$	= numerical success rating
$N_1, N_2$	= arbitrary selected numbers
$n$	= circumferential wave number
$n_a$	= number of problems solved by code "a"
$p$	= hydrostatic pressure
$P_k$	= set of proper constraints
$P$	= progress towards solution (section 4.2)
$p_{cg}(n,m)$	= gross collapse pressure
$p_{cs}(n,m)$	= collapse pressure of a shell panel
$p_{cg}^*, p_{cs}^*$	= minimum collapse pressures
$Q_p$	= total radial load
$R$	= radius of the shell
$R_{min}, R_{max}$	= minimum and maximum radius of the shell
$R_i^+, R_i^-$	= active lower and upper regional constraints
$S_j$	= factor of safety
$t$	= skin thickness
$t_{ap}$	= normalized cpu time
$u$	= best movement direction vector

$V_D$	= displacement volume of the cylinder
$V_f$	= volume of the frame
$V_s$	= volume of plating
$V_w$	= volume of frame web
$W$	= weight of the hull
$W_j$	= deflection parameter
$x, x_i$	= design variable vector and its components respectively
$\alpha$	= step size
$\alpha_{\min}$	= minimum step size
$\Gamma$	= frame deflection parameter
$\gamma_s$	= specific weight of material
$\gamma_w$	= specific weight of immersion fluid
$\epsilon_j$	= constraint activity limit
$\mu$	= poisson's ratio
$\sigma_b$	= frame bending stress
$\sigma_c$	= compressive hoop stress
$\sigma_{fa}$	= allowable frame stress
$\sigma_{pa}$	= allowable plating stress
$\sigma_r$	= axial stress
$\sigma_t$	= maximum frame stress
$\sigma_\phi$	= hoop stress
$ \phi $	= magnitude of arbitrary vector function $\phi$
$\nabla\phi$	= the gradient of arbitrary vector function $\phi$
$(\phi)^T$	= transpose of arbitrary vector function $\phi$



## SUPERSCRIPTS

b	= base number
$\ell$	= comparison value
L	= Lower limit
n	= value at the $n^{\text{th}}$ direction finding problem
r	= number of redesign cycles
s	= number of iterations for locating the IF boundary
U	= upper limit
0	= initial

## CHAPTER 1

### INTRODUCTION

#### 1.1 Optimization in Design

Optimal design problems may be treated by many different methods, for example, ordinary and variational calculus, mathematical programming (MP), and some special techniques, such as the fully stressed concept used in structural design [1-10]<sup>1</sup>.

Of the above mentioned methods, MP procedures [11] seem to be the most flexible methods available for optimal design synthesis since they treat the broadest range of engineering problems and are easily adaptable.

The concept of design optimization requires that a quantity to be minimized or maximized be designated and that this quantity be expressed as a function of the design variables. This function is called the "merit" or "objective" function, and can be written in the form

$$f = f(x_i) \quad i = 1, 2, \dots, I \quad (1-1)$$

where  $x_i$  are the design variables and  $I$  is the number of design variables. Generally, the design variables are not free to take on any value, but are subject to constraints governing their range or the behavior of the design.

---

<sup>1</sup>Numbers in brackets designate references at the end of the thesis.

When limits on behavior are specified, a quantitative measure of the behavior as a function of the variables, called the "behavior function",  $B(x_i)$  is required. Thus, if  $K$  behavior functions are specified, one can write  $K$  equations of the form

$$\begin{aligned} L_k &\leq B_k \leq U_k & k = 1, 2, \dots, m \\ B_k &\leq U_k & k = m + 1, m + 2, \dots, n \\ L_k &\leq B_k & k = n + 1, n + 2, \dots, p \end{aligned} \quad (1.2)$$

where  $L_k$  is the lower bound on  $B_k$ , and  $U_k$  the upper limit.  $L_k$  and  $U_k$  may be functions of  $x_i$ . Behavior constraints representing the above behavior function limits can be written as follows

$$\begin{aligned} g_j &= U_j - B_j \geq 0 & j = 1, 2, \dots, m \\ g_j &= B_{j-m} - L_{j-m} \geq 0 & j = m + 1, m + 2, \dots, 2m \\ g_j &= U_{j-2m} - B_{j-2m} \geq 0 & j = 2m + 1, 2m + 2, \dots, 2m + n \\ g_j &= B_{j-2m+n} - L_{j-2m+n} \geq 0 & j = 2m + n + 1, \dots, 2m + n + p \end{aligned} \quad (1.3)$$

All the above equations may be written as

$$\begin{aligned} g_j &\geq 0 & j = 1, 2, \dots, J \\ J &= 2m + n + p \end{aligned} \quad (1.4)$$

Regional limits are often imposed by manufacturing or other considerations. These are of the form

$$L \leq x_i \leq U \quad (1.5)$$

$x_i^L$  and  $x_i^U$  are constants and represent the minimum and maximum values of  $x_i$  respectively. Here regional constraints are distinguished from behavior constraints since any constraint of the form

$$A \leq x_i \leq B \quad (1.6)$$

where  $A$  and  $B$  are constants, may be treated more simply than the general form. Not all the  $x_i$  need be subject to such limits. A detailed discussion of regional constraints can be found in Reference [9].

Some variables may also be restricted to certain discrete values. In structural design these values can represent material properties, geometric available shapes and sizes, or thickness gages.

The concept of a merit surface is important to the understanding of the redesign process. The function  $f(x_i)$  can be considered as a surface in  $I + 1$  dimensional space where the coordinate axes are the  $x_i$  and  $y$  where the value of  $y$  is given by  $f(x_i)$ . The constraints delineate the region of interest within which the optimum is to be found. Points which satisfy the constraint equations are called "acceptable" or "feasible" points. All other points are called "unacceptable" or "infeasible". The set of all feasible points constitute the "feasible region" and all infeasible points define the "infeasible region". The surface between these regions is called the Infeasible-Feasible (IF) boundary. Those portions of the IF boundary where at least one behavior constraint is zero are called the "behavior" boundary. Points away from the constraint surface are called "free", and those on these surfaces are called "bound" points. The merit surface is explored to find the highest point in the acceptable region. The algorithms commonly employed in such problems usually start the exploration from a free acceptable point. The variables are generally treated as

continuous quantities. If discrete variables are encountered, they may be treated initially as continuous and, upon finding an optimum, a local exploration is made to find the optimal discrete value [12].

## 1.2 General Strategy of Mathematical Programming

Like all optimization methods the MP methods try to find those values  $x$  for which the merit function  $f(x_i)$  will be minimized (or maximized). The problem may be stated as follows:

Find those  $x_i$  that produce the

$$\min. f(x_i) \tag{1.7}$$

such that all the constraints

$$g_j(x_i) \geq 0 \quad j = 1, 2, \dots, J \tag{1.8}$$

and the equality constraints

$$g_k(x_i) = 0 \quad k = J + 1, J + 2, \dots, K \tag{1.9}$$

are satisfied, where the  $x_i$  are the  $I$  real valued variables  $x_1, x_2, \dots, x_I$  in  $I$ -dimensional Euclidian space, and constraints  $g_1, g_2, \dots, g_K$  are real-valued real functions in that space.

Most MP methods do not treat the equality constraints directly. These constraints may, for example, be converted to inequality constraints of the form

$$\epsilon \leq g_k(x_i) \leq \epsilon \tag{1.10}$$

where  $\epsilon$  is an arbitrary small number. One can also treat an equality constraint by solving for one variable in terms of the others, thereby eliminating a variable and constraint.

MP methods are based on searching strategies. The search usually starts from an arbitrarily selected point  $x^0$  and, by means of some local search strategy, a set of points  $x^r$  is generated such that  $f(x^r) < f(x^{r-1})$  while also satisfying all the constraint equations. In the majority of design problems, either  $f$  and/or some or all of the constraints  $g_j$  are nonlinear in  $x$ , and therefore one has a nonlinear, constrained optimization problem.

None of the available nonlinear optimization methods guarantees an optimal solution (when it exists). There are two major difficulties. First, many of the design problems are not unimodal that is, they have more than one local optimum. The nonlinear methods are, however, designed only to locate local optima and thus the global optimum may not be attained. Secondly the search algorithm may simply fail to find even a local optimum.

The prudent designer thus usually tries several different starting points. If all the search paths terminate at the same point, then optimality is assumed and the problem can be considered to be unimodal. However, if the termination point is different for different search paths, then either the best design is accepted or additional starting points are tried in an attempt to find a still better design. Such a decision draws on the designer's experience and judgment. The failure to converge to the same point may be due either to the presence of several local optima, or to algorithm failure. Failure is said to occur when the algorithm ter-

minates at a point significantly away from the nearest local optimum. An excellent discussion of the difficulties involved in the solution of unconstrained nonlinear programming problems is provided in Reference [12]. These difficulties are greatly compounded when constraints are added. For convenience, since this and other MP methods are capable only of locating a local optimum, in the discussion below the term optimum should be taken to mean local rather than global optimum.

### 1.3 Motivation and Structure

Many efficient algorithms exist for treating linear problems, that is, problems with linear objective functions and constraint equations, and certain simple nonlinear problems [5-7, 13]. Most design problems are, however, nonlinear problems that must be treated by relatively less efficient methods.

There are a number of methods available for solution of such problems [14]. However, there are several difficulties with these methods. For example, the computations required to produce improved designs in the neighborhood of the constraint boundaries are in some cases lengthy and relatively complex. The number of function evaluations is often very large, and, most methods are not reliable, that is, the best points produced by these methods may not be a local optima.

Recently, Eason and Fenton [15] compared a group of seventeen numerical optimization methods. This group of algorithms contains

most of the currently popular procedures. The results of this study indicate that the direct search algorithms are superior with respect to generality, efficiency, running cost, speed, preparation cost and reliability [15]. Unfortunately, none of the methods presented in this study are totally reliable. None of the codes solved all of the ten problems attempted.

In a recent paper, Pappas and Moradi compared a code based on the Direct Search-Feasible Direction Algorithm (DSFD) [16] with those studied by Eason and Fenton [17]. This code solved all the test problems treated in Reference [15]. In addition it also solved a difficult six-variable problem which the relatively reliable DSDA, and popular SUMT, procedures failed to solve [17]. In this study the DSFD based code proved superior to all others in overall generality and efficiency.

Even the relatively efficient DSFD however, required several hundred to several thousand function evaluations to achieve a solution to Eason's and Fenton's test problems. Therefore substantial computational effort and cost would be required by these procedures on problems with computationally demanding functions. Since many important engineering problems are of this type, there is a clear need for a new algorithm which requires a reduced number of function evaluations for solution.

This thesis presents an apparently fast and reliable algorithm which utilizes a search strategy designed to greatly reduce the number



of function evaluations required for solution. This is accomplished by developing a scheme which keeps the search confined to the neighborhood of the behavior boundary, on the premise that this is where the optimum is usually found in constrained problems.

The search moves efficiently along this boundary by making effective use of a relatively small amount of new local function information relying primarily on information generated earlier in the search. Thus, the number of new function evaluations required to sustain movement and thereby the total number of function evaluations required for solution are kept low.

Since the advent of high speed digital computers many accurate but computationally demanding methods have been developed for engineering analysis. Due to the relatively large execution time required for each reanalysis cycle (function evaluation) of many of these techniques and the large number of reanalysis cycles required by most MP procedures, the combination of such analytic methods and MP procedures may be very costly to utilize and are therefore often impractical. This difficulty can seemingly be minimized by use of the Boundary Tracking (BT) algorithm due to its apparent efficiency in reducing the number of function evaluations and therefore total execution time required for solution.

Where the BT algorithm is not available in a compiled form, a simpler algorithm may be preferable for use on small simple problems. In such circumstances the extra compilation time required due to the

greater complexity of the BT algorithm may overshadow the time saved by a reduction in the number of function evaluations required for solution.

CHAPTER 2  
DESCRIPTION OF GENERAL STRATEGY OF  
THE BOUNDARY TRACKING ALGORITHM

2.1 Introduction

The most common strategy for constrained optimization techniques is first to move to the IF boundary from a starting point and then to move along this boundary to the optimum. A typical objective function surface is illustrated in Figure 1.

The general strategy of the Boundary Tracking (BT) algorithm is to first locate a point on the behavior boundary. Once the behavior boundary is located the best direction for movement along this boundary is determined. Movement along this boundary is then continued until the optimum point is obtained.

Figure 2 presents a general block diagram of the BT algorithm.

2.2 Location of the Behavior Boundary

The search starts from an arbitrary starting point  $x_i^0$ . If the problem has behavior constraints, these constraints are evaluated at this point. If the starting point proves to be in the feasible region, it is designated as a base point (point 1 Figure 1). A step is then taken in the direction of the gradient of the objective function. All the constraints are evaluated after the step. If the new point is in the feasible region the objective function is evaluated and compared to the last base point. If the function has improved, this new point is called a base point (points 2, 3, 4 in Figure 1) and the search

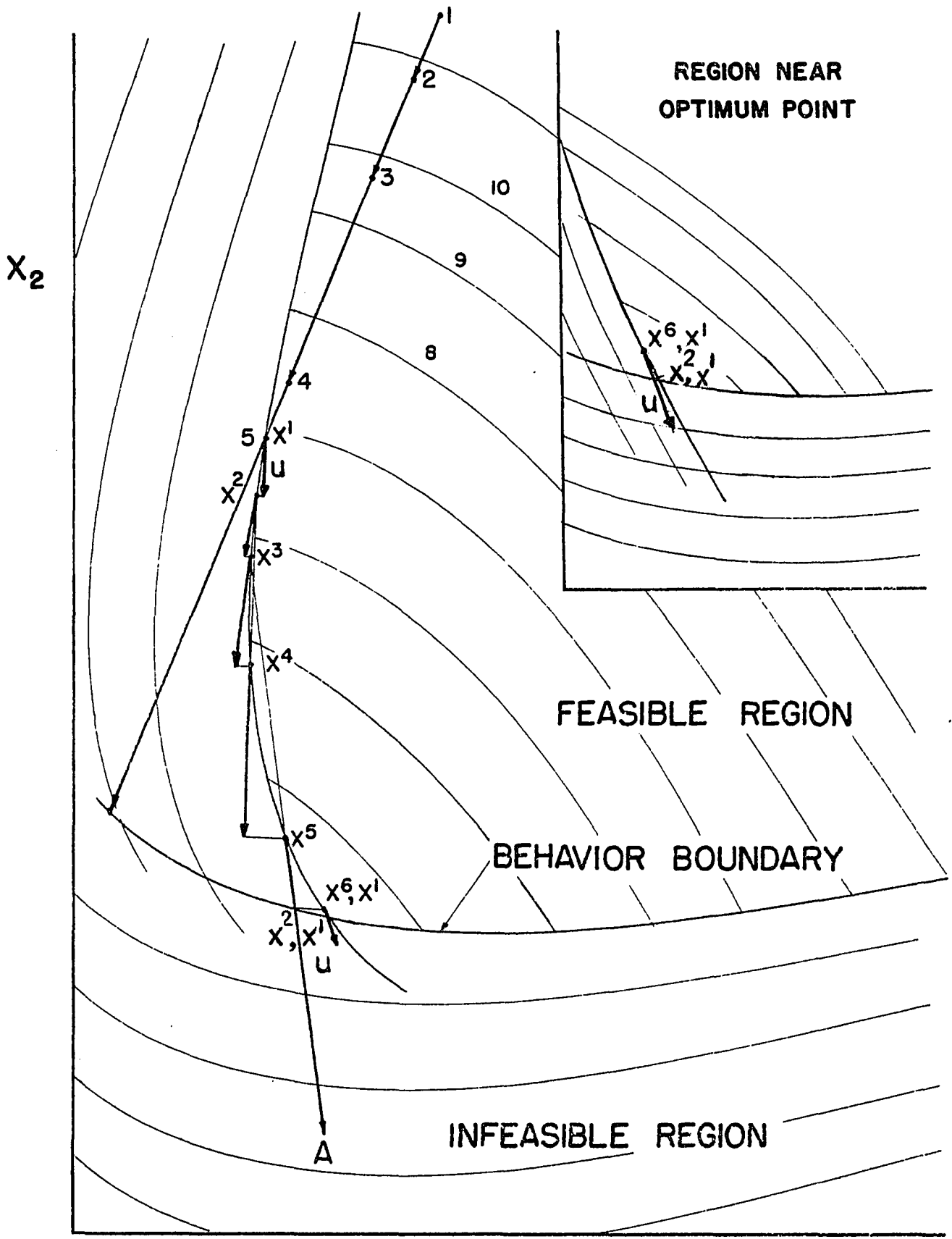


Figure 1. General Strategy of Boundary Tracking Method  $X_1$

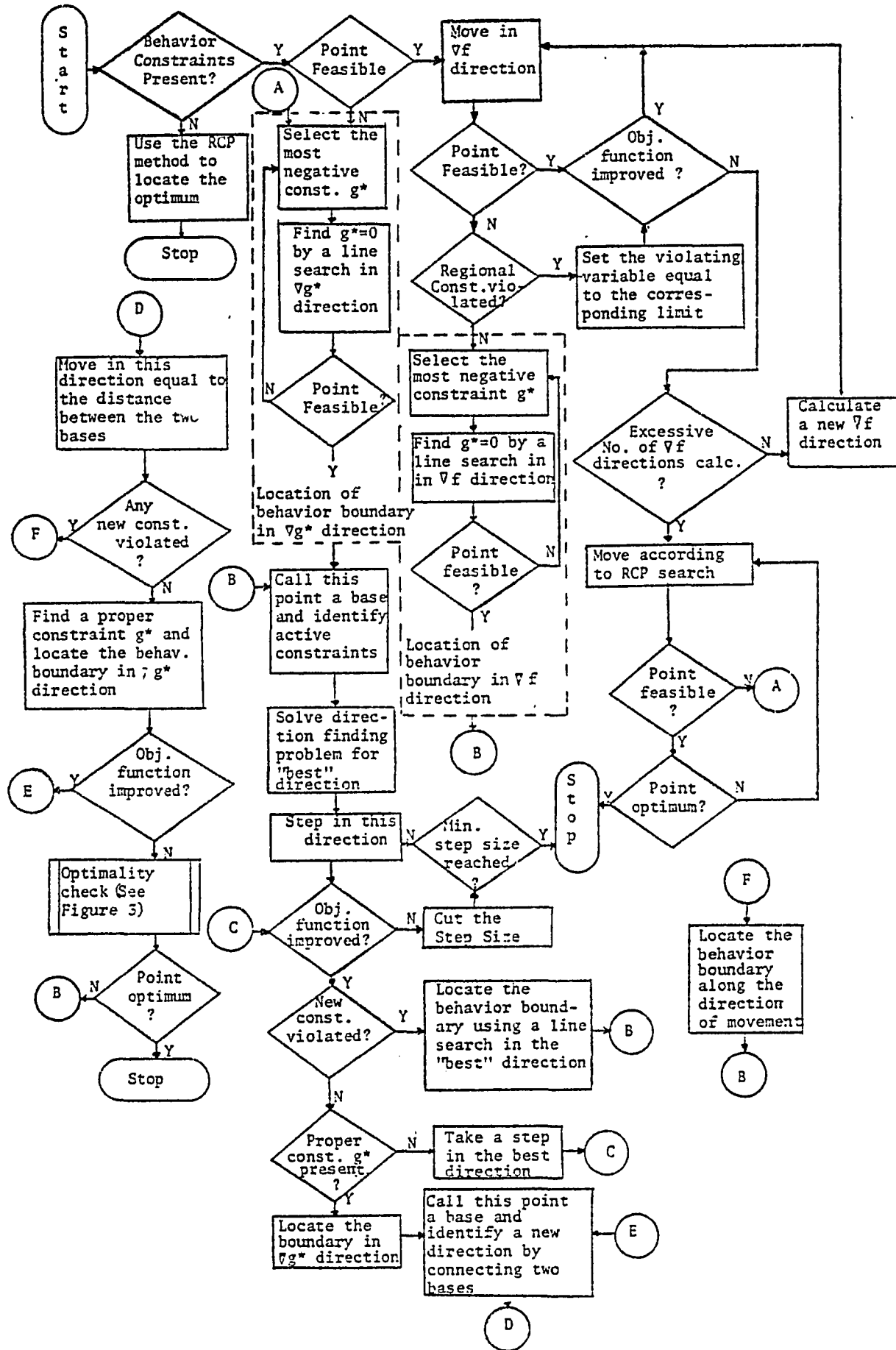


Figure 2 - Block Diagram of the General Strategy

continues, doubling the step size with each move.

Once the behavior boundary is crossed, a point on this boundary is located (point 5) using a line search in the direction of the last move.

The search is performed in the direction of movement, rather than say the direction of the gradient of the constraint function violated, since identification of the latter direction requires calculation of constraint function derivatives, while the former direction is available and thus requires no new information. The line search is used because of its simplicity.

If the search move crosses the IF boundary at a regional limit, it is possible to find the best direction to continue movement by solving Zoutendijk's direction finding problem (see section 3.2.3). Here for simplicity, however, the variables exceeding the limits are set equal to the corresponding limit and the search continued until a behavior constraint is violated. This strategy tends to deflect the direction of movement along the IF boundary.

If after a step, the point remains in the feasible region but the objective function has not improved, a new gradient direction is calculated at the last base point and the search restarted. If several direction changes fail to locate the constraint boundary, a more efficient, albeit more complex, unconstrained search method such as the modified Rotating Coordinate Pattern (RCP) search [17], is

used. This search procedure is applied here to avoid the zig-zagging problem associated with the gradient search method [12]. Thus the advantages of the gradient search method are exploited where possible, but the method is replaced by a more appropriate procedure whenever strategy dictates. If the behavior boundary is crossed in the modified RCP search, the nearby boundary may be located by any convenient method such as that already described.

The major contribution of this thesis is the procedure for movement along the behavior boundary. It is this movement which causes the difficulty associated with the solution of constrained nonlinear problems. The method used for locating the boundary, such as the gradient search or the RCP search, is of secondary importance. Since the bulk of the search is associated with movement along the boundary, the choice of the initial boundary locating procedure will not substantially effect overall performance. Thus, the procedures used for this purpose are not of great importance and need not be described in detail here. The interested reader is referred to references [12, 17] for a detailed description of these methods.

If the starting point is infeasible, the behavior boundary is located by first selecting the most negative constraint, that is, the constraint presenting the greatest violation. The point where this constraint vanishes is then located using a line search in the direction of the gradient of this constraint (see section 3.1.2). This direction is used here since it provides the shortest distance to the

surface where this constraint vanishes. The line search procedure used here is a modified Secant root finding procedure (see section 3.1.2).

All the constraints are then evaluated at the point where the selected constraint vanishes. If no violation exists the objective function is evaluated and the point is designated as a base point. However, if any other constraints are violated at this point, the smallest one (algebraically) is again selected and the point where it is zero is found. This process is continued until a point on the behavior boundary is located.

For problems without behavior constraints an unconstrained search method, such as the modified RC Pattern search [18], is used to locate the optimum point. This search method combines the well known RC Pattern search with Zoutendijk's feasible direction finding method (see section 3.2.3). The direction finding procedure is employed at the points of RC Pattern search failure.

### 2.3 Movement Along The Behavior Boundary

Once the point on the behavior boundary is located it is designated as a base point (point  $x^1$  in Figure 1). The best direction to move is then determined by solving the direction finding problem of Zoutendijk (see section 3.2.3) and a step taken in this direction. The objective function is then evaluated at this point. If the function has improved, all the constraints are evaluated.



If none of the constraints that were inactive at the base are violated, an appropriate constraint is selected from among the active constraints (see section 3.1.1). The behavior boundary is located by finding a point where this constraint is zero by means of a line search in the direction of the gradient of the selected constraint (point  $x^2$ ).

Upon locating the point on the boundary after the best direction move, the objective function is evaluated. If the function has improved the point is designated as a base point (point  $x^2$ ). A move is then made from the last base in the direction of improvement along a line connecting two bases a distance equal to the distance between these bases.

The boundary is then located as before (see point  $x^3$  of Figure 1) along the direction of the gradient of the appropriate constraint as evaluated at the last point where the best direction finding problem was formulated. It should be pointed out that this is an underlying feature of this algorithm, making multiple use of function evaluations in order to keep the number of new evaluations required as small as possible. Only if this direction fails to locate the boundary, are new gradient values computed.

When the point on the behavior boundary is found, it is compared to the last base point. If the function has improved, the point is designated as a new base point (points  $x^3$ ,  $x^4$ ,  $x^5$ ) and the search is continued. However, if the function has not improved the direction

is abandoned and the search is re-started from the last base point by calculating a new best direction.

After each move all the constraints are evaluated, if a new constraint is violated (point A) the point where this constraint is zero is located and a new best direction calculated. However, if no new constraint is violated, an appropriate constraint is selected from among the previously active constraints (see section 3.1.1) and the boundary located, the process is continued until convergence is achieved.

If no appropriate constraint is active the move along the previous direction is continued, since it is the best direction available. If after a step in the above direction, the objective function has increased rather than decreased the step size is reduced. The process is repeated until a better point is found or the convergence criterion is met (see section 3.3).

If after any move a constraint inactive at the last base point is violated, the behavior boundary is located by finding the point where this constraint is zero. This point is found by a line search in the direction of the last move. This direction is used to eliminate the need to calculate the gradient of the new constraint. The search is then re-started by calculating the best movement direction at this point on the boundary.

2.4 Search termination description. For the few cases where the optimum occurs away from the behavior boundary, a sufficient condition for optimality is that all components of the gradient of the objective function be essentially equal to zero. For most cases, however, where the optimum is on the behavior boundary no such simple condition exists.

The search termination procedure used here is as follows. The search using a specified basic step size is performed until no further improvement can be obtained. When such a point is found it is designated as an optimality comparison base. Now the basic step size is reduced in an effort to achieve improvement and the search continued until a new optimality comparison base is obtained.

In order to justify a further reduction in the step size an optimality check is performed whenever the step size is to be reduced. If a convergence limit is satisfied, the step size is again reduced for further improvement and the search is continued until another optimality comparison base is obtained. A secondary convergence check is then performed in order to determine whether the latest improvement due to the step reduction is less than the previous improvement thereby indicating convergence has occurred. If both convergence tests are simultaneously satisfied, or the minimum step size is reached, the search is terminated. Otherwise, the search is re-started by calculating a new best movement direction.

A block diagram of the search termination procedure is given in Figure 3.

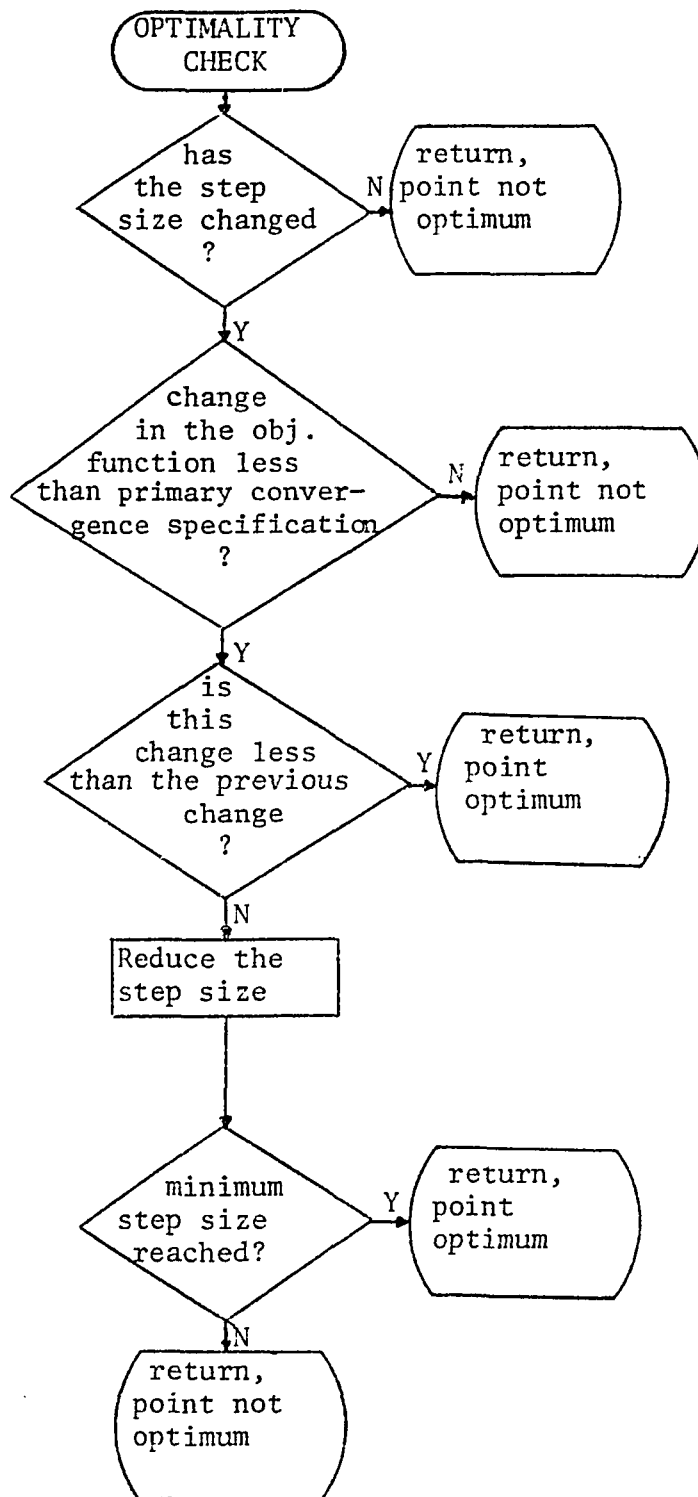


Figure 3 - Block diagram of the termination procedure

CHAPTER 3  
MATHEMATICAL DEVELOPMENT

3.1 Boundary Location

It is assumed for the purpose of devising a search strategy that the solution to the optimization problem with behavior constraints lies on the behavior boundary. The main strategy is first to locate such a boundary and then to move along this boundary. For the purpose of locating the boundary a "proper" behavior constraint is identified and method for locating the behavior boundary selected or developed.

3.1.1 Identification of "proper" constraints. All negative and small positive constraints are designated as active constraints (for definition of activity see section 3.2.3). A constraint is considered "proper" if it is necessary or desirable to locate a point where the value of this constraint is zero after making a search move in an effort to establish a new base.

Since it is necessary to eliminate any constraint violation present at a point, all the negative constraints are considered "proper".

If all active constraints are positive, a positive active constraint is considered proper if

$$-\nabla f \cdot \nabla g_j < 0 \quad (3.1)$$

This indicates that the projection of  $-\nabla f$  on  $\nabla g$  is of opposite sign to  $\nabla g$  (point 1 Figure 4) and therefore, the value of the constraint

$g_j$  will decrease as a result of a move in  $-\nabla f$  direction (minimization problem).

The best direction for improving a function is its gradient direction. However, this direction is not always feasible in constrained problems (point 1 Figure 4).

A direction is desired such that after taking a step, the objective function will improve the maximum amount possible without violating the behavior constraints. A direction along the behavior boundary often has this property. Moving along the behavior boundary, however, may not always produce the best improvement in the objective function as shown by point 2 in Figure 4. Here projection of  $-\nabla f$  on  $\nabla g_j$  has the same sign as  $\nabla g_j$ . Thus, the value of the constraint  $g_j$  will increase by moving in  $-\nabla f$  direction. A criteria is therefore needed that can indicate which positive constraint (if any) is "proper". Equation (3.1) satisfies this purpose, since if satisfied it identifies the constraints that may be violated if a move in the best direction ( $-\nabla f$ ) is made. Thus it identifies proper constraints, that is, those constraints along which it is desirable to move in order to achieve the best improvement in the objective function.

If no proper constraint can be identified, the movement along the previous direction is continued, since there is no advantage in locating and moving along the nearby boundary.

3.1.2 The boundary locating method. Call the set of proper constraints  $P_k$  where,  $k = 1, 2, \dots, K$ . If more than one "proper"

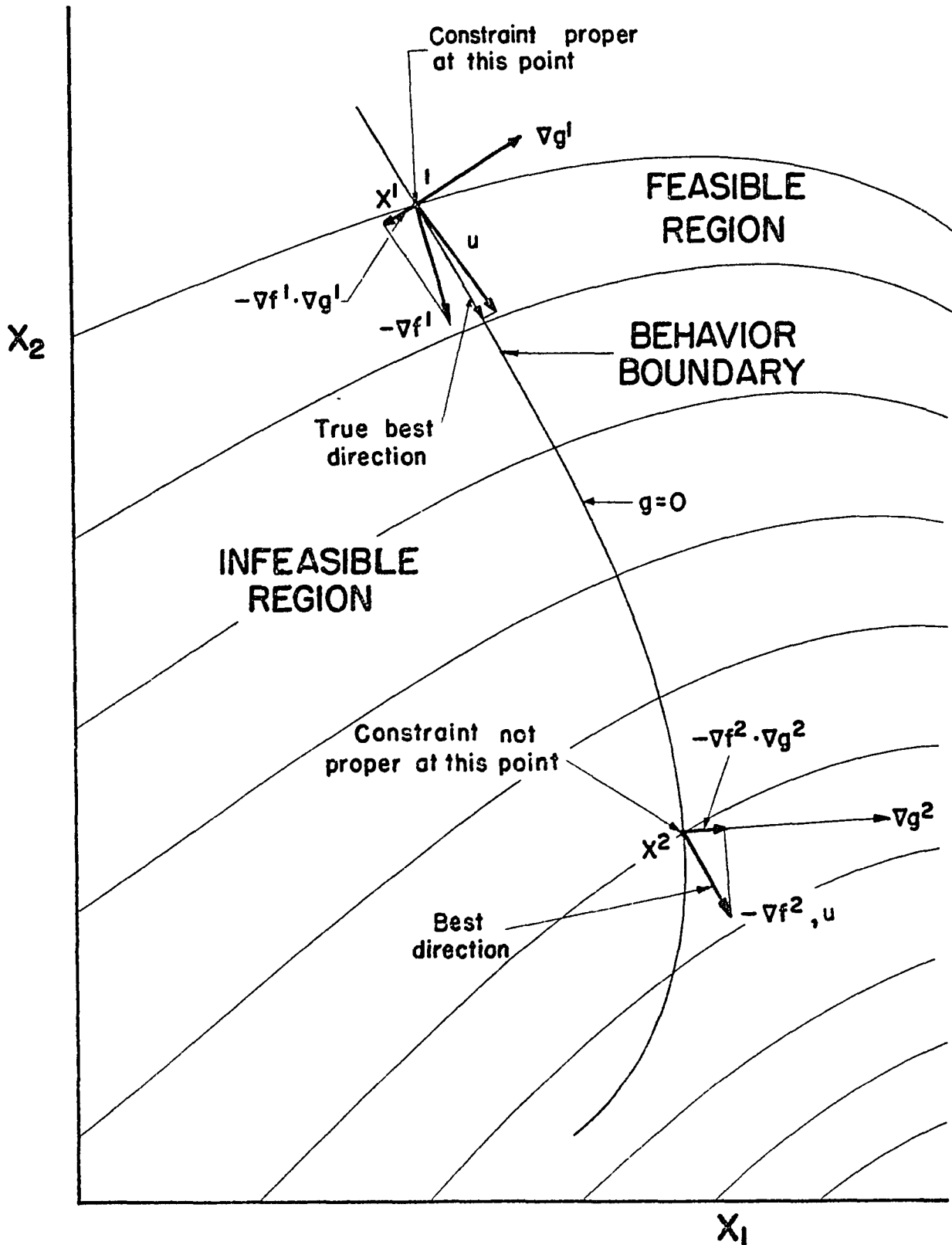


Figure 4 - Selection of a Proper Constraint



constraint is identified, that is if

$$K > 1 \quad (3.2)$$

the smallest (algebraically) is selected and called  $g^*$ . The root of  $g^*$  is then found in the direction of movement or the gradient of  $g^*$ .

The root, if it exists, of the constraint  $g^*(x)$  along a line may be found as closely as desired by means of the "secant root-finding" method [19] by iterating the equation

$$x^{s+1} = x^s - g^*(x^s)(x^s - x^{s-1}) / [g^*(x^s) - g^*(x^{s-1})] \quad s = 1, 2, \dots, N_1 \quad (3.3)$$

where  $N_1$  is the maximum number of iterations permitted, until

$$|g^*(x^{s+1})| \leq e_1 \quad (3.4)$$

is satisfied. Here  $e_1$  is an arbitrarily selected accuracy limit.

Initially  $x^s, x^{s-1}$  are two points on the line defined by

Case A: If the boundary is to be located in the direction of the last search move (see section 2.3) let the initial points on the line

( $s = 1$ ) be given by

$$x^{s-1} = x^b \quad (3.5)$$

$$x^s = x^r \quad (3.6)$$

where  $x^b$  and  $x^r$  are the end points of the last search step.

Case B: If the boundary is to be located in the gradient direction of the constraint  $g^*(x)$  (see section 2.3) let

$$x^{s-1} = x^r \quad (3.7)$$

$$x^s = x^r \pm \alpha^n \nabla g^*(x^n) / (\nabla g^*(x^n)) \quad (3.8)$$

where  $x^r$  is the point from which the boundary is to be located and  $x^n$  is the last base where a direction finding problem was formulated

(see section 3.2.3.). Here the plus and minus signs are used when  $g^*$  is negative or positive respectively. The quantity  $\alpha^n$  is the step size used at point  $x^n$ .

Case C: For the case where this procedure is used to find the behavior boundary where a starting point of the optimal search is infeasible, replace  $x^r$  by the starting point  $x^0$  and let,

$$\alpha^n = \alpha^0 \quad (3.9)$$

where  $\alpha^0$  is an arbitrarily specified initial step size, and let

$$\nabla g^*(x^n) = \nabla g^*(x^0) \quad (3.10)$$

in equations (3.7) and (3.8).

After equation (3.4) is satisfied, the remainder of constraints in the set  $P_K$  may also vanish. However, if after eliminating the the smallest violation

$$g_j(x^{s+1}) \geq -e_j \quad j = 1, 2, \dots, J \quad (3.11)$$

is not satisfied, a new  $g^*$  is selected and the procedure repeated until a point on the behavior boundary is located (equation 3.11 satisfied).

In case B, slow convergence may indicate that the gradient  $\nabla g^*(x^n)$  may no longer be applicable at point  $x^n$ . This situation is shown in Figure (5). There is no point along the gradient  $\nabla g^*(x^1)$  which falls on the boundary. A new gradient direction  $\nabla g^*(x^r)$  is therefore calculated and used in place of  $\nabla g^*(x^n)$ . If after calculating a new gradient direction the boundary still cannot be located with a reasonable number of iterations, the difficulty may be due to too large a reach step (line  $x^6 - x^r$  in Figure 5). Thus where recalculation of the

gradient fails, the step size is reduced and another effort is made to locate the boundary from a point somewhat closer to the last base.

Therefore, unless

$$s < N_1 \quad (3.12)$$

where  $N_1$  is chosen empirically, let

$$\nabla g^*(x^n) = \nabla g^*(x^r) \quad (3.13)$$

in equation (3.8) and repeat the application of equation (3.3). If equation (3.12) is again violated let

$$\Delta x^{r+1} = \Delta x^r / 2 \quad (3.14)$$

where

$$\Delta x^r = x^r - x^b \quad (3.15)$$

and  $x^b$  is the last base point. Then select the smallest proper constraint at point  $x^{r+1}$  where

$$x^{r+1} = x^b + \Delta x^{r+1} \quad (3.16)$$

(point  $x^b + \Delta x^{r+1}$  in Figure 5), replace  $x^r$  by  $x^{r+1}$  in equations (3.7) and (3.8) and repeat the iteration of equation (3.3). The process is continued until the boundary is located without violating equation (3.12) or until

$$|\Delta x^{r+1}| < \alpha^n. \quad (3.17)$$

If equation (3.17) is satisfied the attempt to locate the boundary is abandoned. The optimal search is then restarted by calculating a new starting direction (see section 3.2.3) at the last feasible point encountered.

### 3.2 Mathematical Representation of General Strategy

The general strategy of the algorithm can be represented mathematically as follows.

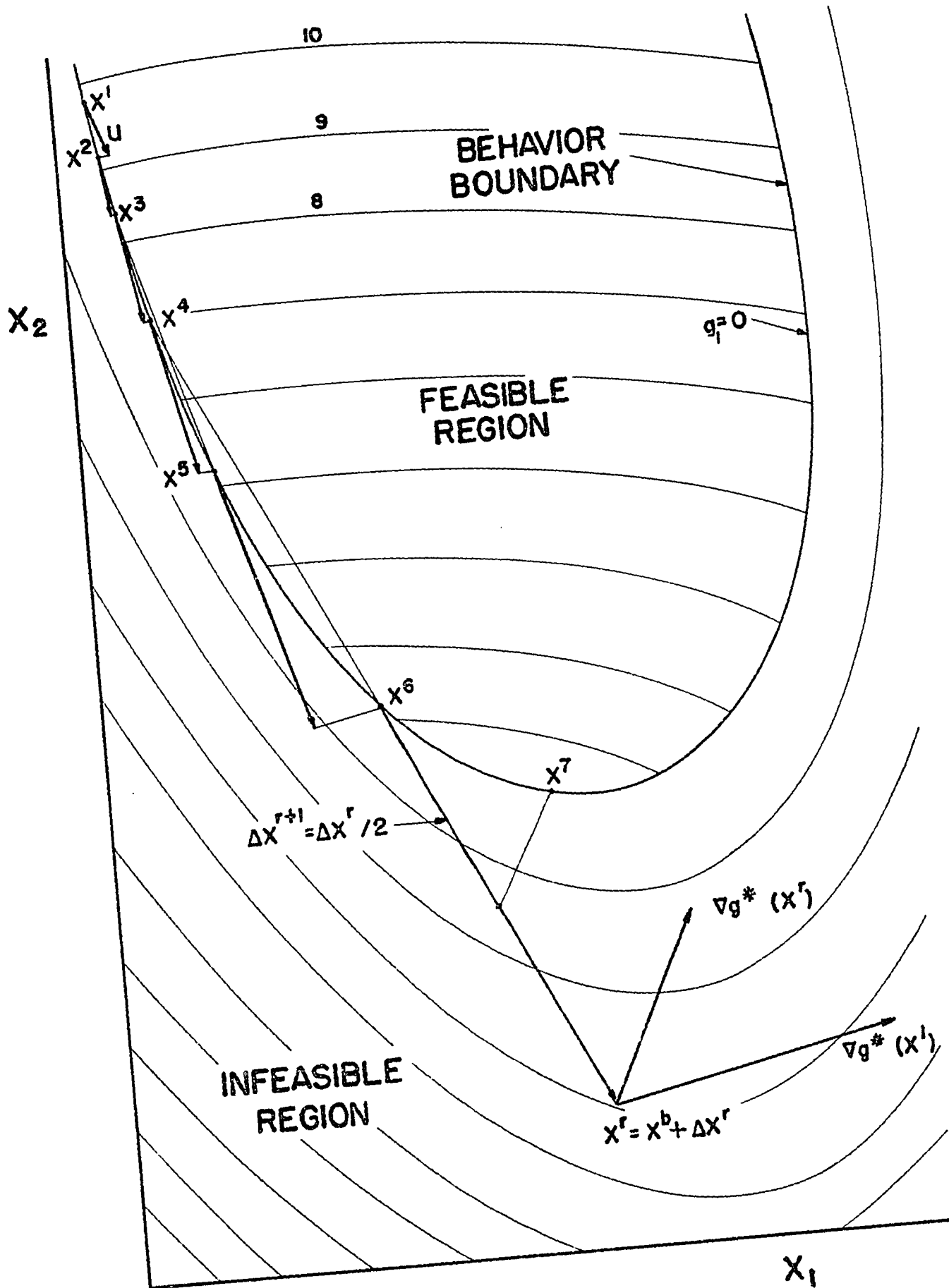


Figure 5 - The Boundary Locating Method

3.2.1 Movement to the boundary-starting point in the feasible region. A block diagram for the movement to the boundary procedure is given in Figure 6.

Given the starting point  $x_i^r$ , where initially  $r = 0$ , which satisfies regional constraints

$$x_i^L \leq x_i^r \leq x_i^U \quad i = 1, 2, \dots, I \quad (3.18)$$

and all

$$g_j(x_i^r) \geq 0 \quad j = 1, 2, \dots, J \quad (3.19)$$

the objective function  $f(x^r)$  is evaluated. This point is called a base point  $x^b$ , where initially  $b = n = 1$ , and the first comparison base  $x^n$  is defined as,

$$x^n = x^b \quad (3.20)$$

Then with

$$\alpha^r = \alpha^n \quad (3.21)$$

let

$$x^{r+1} = x^r - \alpha^r [\nabla f(x^n)] / |\nabla f(x^n)| \quad (3.22)$$

where  $\nabla$  is the gradient operator,  $|\nabla f|$  the magnitude of  $\nabla f$  and,  $\alpha^r$  is the step size at iteration  $r$ .

If any

$$\left. \begin{array}{l} x_i^{r+1} > x_i^U \\ x_i^{r+1} = x_i^U \end{array} \right\} \quad (3.23)$$

or if any

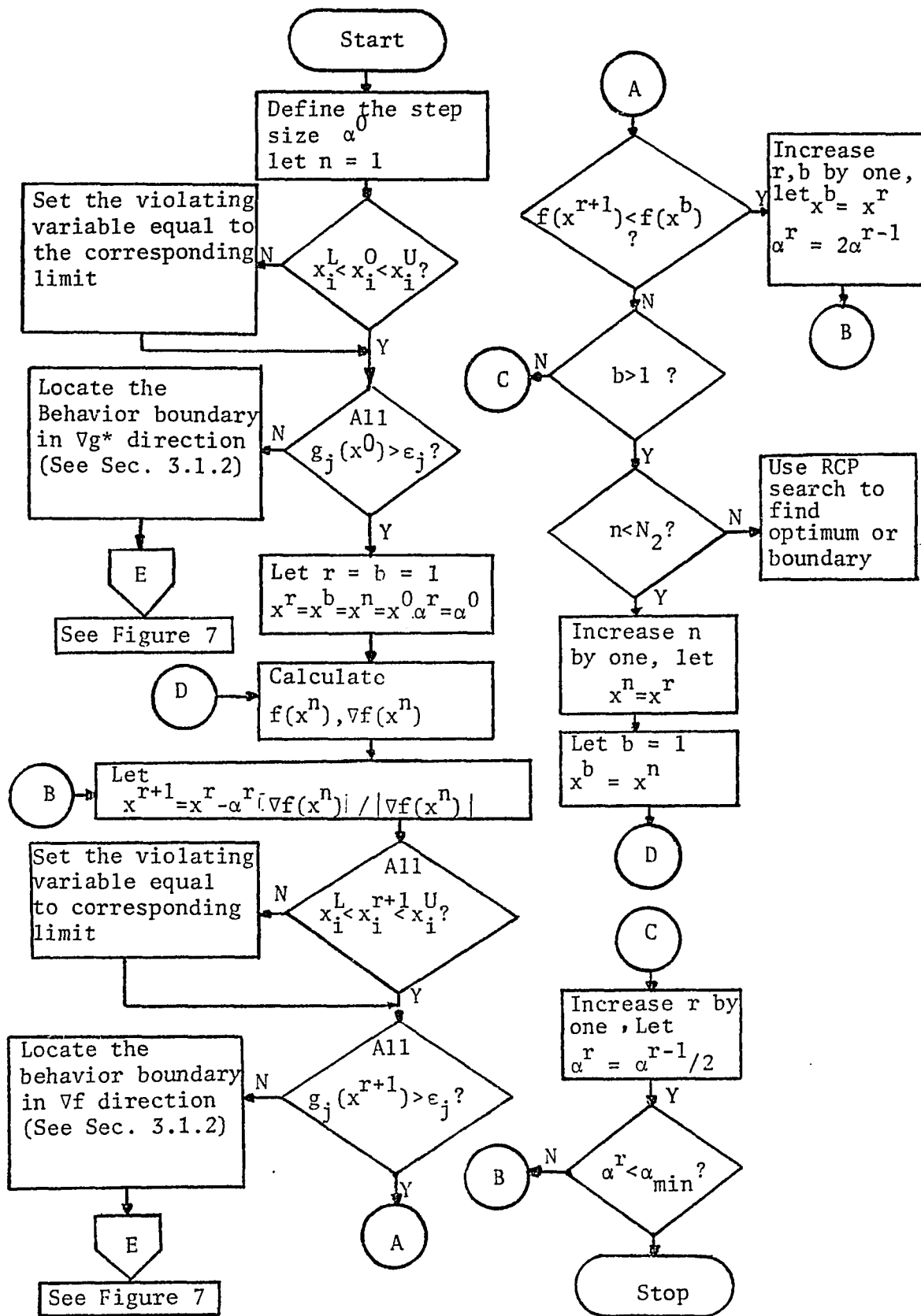


Figure 6 - Flow Chart of the Procedure for Movement to the Boundary

let

$$\left. \begin{aligned} x_i^{r+1} &< x_i^L \\ x_i^{r+1} &= x_i^L \end{aligned} \right\} \quad (3.24)$$

Now compute the  $g_j(x^{r+1})$ , if equations (3.19) are satisfied, evaluate  $f(x^{r+1})$ . If

$$f(x^{r+1}) < f(x^b) \quad (3.25)$$

call the point a new base point. Thus let

$$x^{b+1} = x^{r+1} \quad (3.26)$$

A new move is then made according to equation (3.22) with the step size redefined as

$$\alpha^{r+1} = 2\alpha^r \quad (3.27)$$

and application of equations (3.23-3.26) repeated. If  $b > 1$  and equation (3.25) is not satisfied increase  $n$  by one, let the next comparison base

$$x^n = x^b \quad (3.28)$$

and let

$$x^r = x^b \quad (3.29)$$

Now if

$$n < N_2 \quad (3.30)$$

where  $N_2$  is an arbitrary constant signifying the number of direction changes, a new gradient direction  $\nabla f$  is calculated. The application of equations (3.20-3.27) are then repeated until either a point in the infeasible region is found or equation (3.30) is not satisfied. If point  $x^{r+1}$  is in the infeasible region the behavior boundary is located in the direction of the last move (see section 3.1.2).

If  $n$  exceeds  $N_2$  the gradient search is abandoned and the RCP search invoked. This search method is continued in the unconstrained

region until a point in the infeasible region is found or the optimum is achieved (see section 3.3).

If equation (3.25) is not satisfied when  $b=1$ , the step size  $\alpha$  is redefined as

$$\alpha^{r+1} = \alpha^r/2 \quad (3.31)$$

and a new move is made according to equation (3.22). The application of equations (3.31) and (3.22) is repeated until equation (3.25) is satisfied or

$$\alpha^{r+1} \leq \alpha_{\min} \quad (3.32)$$

where  $\alpha_{\min}$  is a minimum step size. In the latter case the point is assumed to be optimum and the search is terminated.

3.2.2 Movement to the boundary - starting point in the infeasible region. If the starting point  $x_i^0$  is in the infeasible region or the RCP search has located a point in this region the behavior boundary is located by a search in a constraint gradient direction (see section 3.1.2).

3.2.3 Initiation of movement along the boundary. After locating a point on the behavior boundary a method is needed which can initiate the move along this boundary toward the optimum if optimality has not yet been achieved. The Feasible direction finding algorithm of Zoutendijk [20] is suitable for this purpose since, it either provides a direction along which an improved point can be found, or indicates the presence of a local optimum.



Definition: a) A direction vector  $u$  is called "feasible" if after taking a sufficiently small step along this direction no constraint is violated [20]. This will be true if

$$(u)^T \nabla g_j(x) \geq 0 \quad (3.33)$$

where  $(u)^T$  is the transpose of  $u$ , since a small step in this direction will produce no change or an increase in  $g_j(x)$ .

b) A feasible vector  $u$  is called "usable" if a move in the direction  $u$  can also provide an improvement in  $f(x)$ . This will be the case if

$$(u)^T \nabla f(x) < 0 \quad (3.34)$$

since a sufficiently small step in the  $u$  direction will produce a decrease in the value of  $f(x)$ .

c) A behavior constraint  $g_j(x)$  is considered "active" if

$$g_j(x_i) \leq \epsilon_j \quad (3.35)$$

where

$$\epsilon_j = K_j (\alpha^n / \alpha^0) \quad (3.36)$$

is an array of positive, arbitrary small numbers. These numbers may approach zero during the search, but can not be identically zero [20]. The quantity  $\alpha^0$  is the step size at the beginning of the search and  $K_j$  is a small positive constant.

d) A lower regional constraint is considered "active" if

$$x_i - x_i^L \leq \alpha^n \quad (3.37)$$

and the upper constraints active if

$$x_i^U - x_i \leq \alpha^n \quad (3.38)$$

Denote the set of all active behavior constraints by  $A_j$ . Call  $R_i^-$  and  $R_i^+$  the active lower and upper regional constraint set respectively.

The direction finding problem can be formulated in the following manner:

Given  $x$ , find  $u$  that results in the

$$\max \sigma \quad (3.39)$$

and for which

$$\sigma > 0 \quad (3.40)$$

$$(u)^T \nabla f(x) + \sigma \leq 0 \quad (3.41)$$

$$-(u)^T \nabla g_j(x) + W_j \sigma \leq 0 \quad j \in A_j \quad (3.42)$$

$$u_i \leq 0 \quad i \in R_i^- \quad (3.43)$$

$$u_i \geq 0 \quad i \in R_i^+ \quad (3.44)$$

$$|u_i| \leq 1 \quad i = 1, 2, \dots, I \quad (3.45)$$

Equations (3.45) bound the length of  $u$  to prevent the direction finding problem from producing an unbounded solution vector.

When suitable deflection parameters  $W_j$  are used, the solution of the problem provides a usable and feasible direction  $u$  since, if  $\sigma > 0$  from equations (3.41) and (3.42) equations (3.33) and (3.34) will be satisfied. It also provides the best usable direction since, if  $\sigma$  is maximized, the left hand side of equation (3.34) will be a maximum and therefore the direction found will be the best direction for decreasing  $f(x)$  [20].

It may be seen that the direction finding problem of equations (3.39 - 3.45) is a linear programming problem. Thus, any linear programming method such as the simplex procedure [21] can be used to provide the solution.

Since the objective here is to move along the IF boundary, the direction  $u$  should be as close to this boundary as possible. The work associated with locating the boundary will therefore be minimized after the move. The deflection parameters  $W_j$  here are thus set to be equal to zero. This produces a direction  $u$  that tends to be tangent to the boundary.

3.2.4 Movement along the boundary. A flow chart for the movement along the boundary procedure is given in Figure 7.

Call the point on the behavior boundary  $x^b$  and define

$$\alpha^\ell = \alpha^n \quad (3.46)$$

where initially  $b = \ell = 1$ ,  $x^b$  is a base point, and  $\alpha^\ell$  is a comparison step size. A direction  $u$  is then calculated (see section 3.2.3) and the point  $x^r$  defined as

$$x_i^r = x_i^b + \Delta x_i^r \quad (3.47)$$

where

$$\Delta x_i^r = \alpha^\ell u_i / |u| \quad (3.48)$$

if

$$f(x^r) < f(x^b) \quad (3.49)$$

and

$$g_j(x^r) > \varepsilon_j \quad \begin{matrix} j = 1, 2, \dots, J \\ j \notin A_j \end{matrix} \quad (3.50)$$

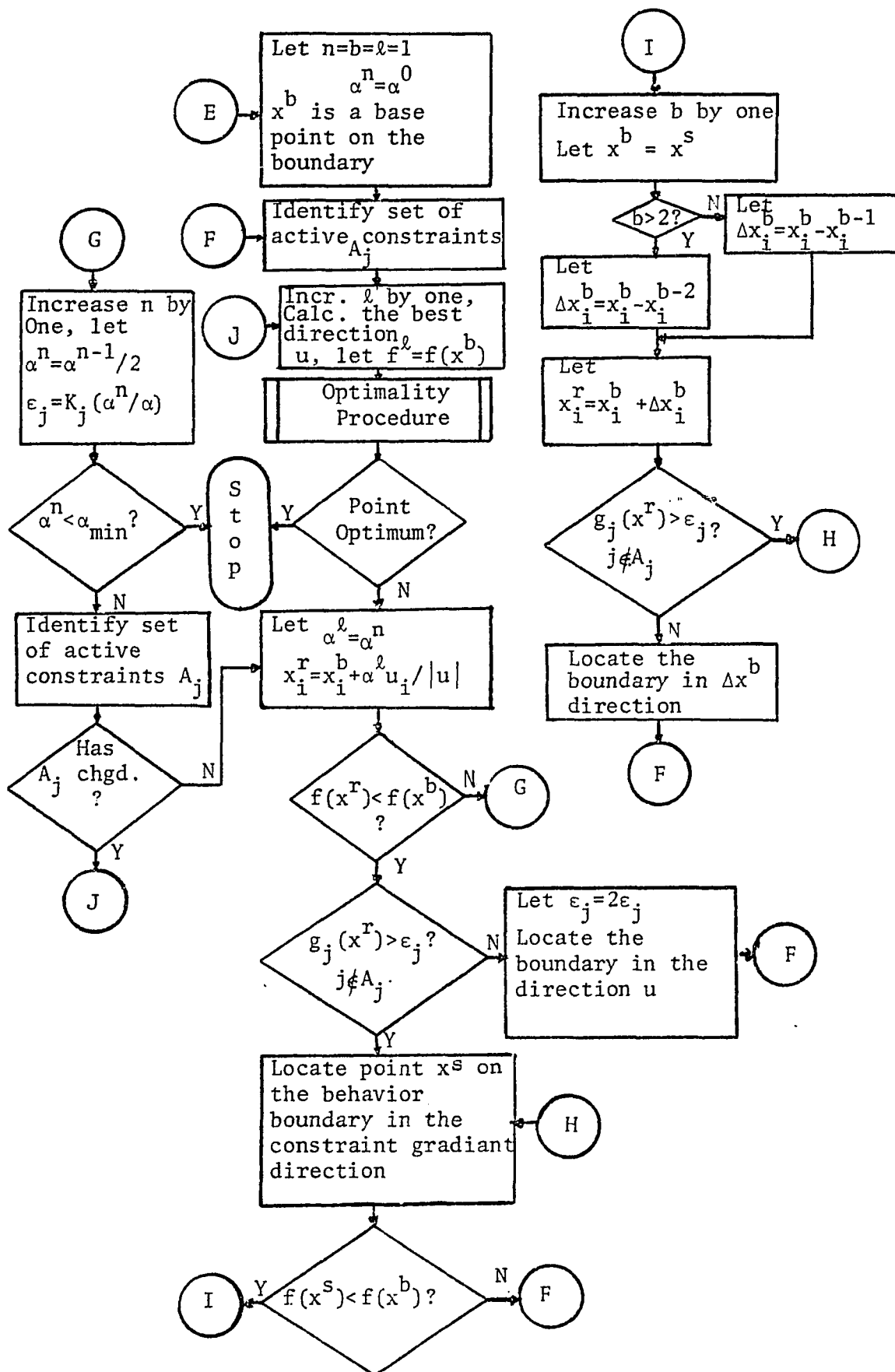


Figure 7 - Flow Chart of the Procedure for Movement along the Boundary

are satisfied, point  $x^S$  on the behavior boundary is located by a search in the constraint gradient direction and the objective function evaluated. Now, if equation (3.49) is satisfied at  $x^S$ , let

$$x^{b+1} = x^S \quad (3.51)$$

now index  $b$  and define the next step as

$$x_i^r = x_i^b + \Delta x_i^r \quad (3.52)$$

where

$$\Delta x_i^r = x_i^b - x_i^{b-2} \quad b > 2 \quad (3.53)$$

or

$$\Delta x_i^r = x_i^b - x_i^{b-1} \quad b = 2 \quad (3.54)$$

The constraints are now evaluated. If equation (3.50) is satisfied, the behavior boundary is located in the constraint gradient direction and the application of equations (3.51 - 3.54) repeated.

If when  $b=1$ , equation (3.50) is not satisfied, that is, if new constraints have become active, the value activity limit  $\epsilon_j$  is doubled for all constraints in set  $A_j$  not satisfying equation (3.50), since its previously assigned value was not sufficient to properly define activity. The boundary is then located in the direction  $u$  and the application of equations (3.47 - 3.50) repeated.

If equation (3.50) is not satisfied when  $b > 2$  the boundary is located in the direction of the last move  $\Delta x^r$ . Then a new set of active constraints is defined and the search re-started from the last base point (see section 3.2.3).

In application of equations (3.49 - 3.54) if equation (3.50) is satisfied, but equation (3.49) is not satisfied, this indicates that the last search move direction was unproductive, therefore the last direction of movement is abandoned and the search is re-started from the last base point  $x^b$  by calculating a new direction  $u$ , and setting  $b=1$ .

After employing equations (3.47) and (3.48), if equation (3.49) is not satisfied, this indicates that the step size is too large, therefore let

$$\alpha^{n+1} = \alpha^n/2 \quad (3.55)$$

$$\alpha^{l+1} = \alpha^{n+1} \quad (3.56)$$

Now, since new activity limits are defined [see equation (3.36)] the active constraints are again determined, if the set  $A_j$  has changed a new direction  $u$  is calculated, otherwise the previous direction is used and application of equations (3.47 - 3.49) repeated using the new step size. This procedure is continued until equation (3.49) is satisfied or convergence is achieved.

Application of equations (3.47 - 3.56) is invoked when applicable, as explained above, until the convergence or optimality criterion are met (see section 3.3).

### 3.3 Search Termination

A flow chart for the search termination procedure is given in Figure 8.

The question of optimality is considered at all points where the direction finding problem is formulated. For the few cases where a local optimum occurs away from the behavior boundary that is where all

$$g_j(x) > \epsilon_j \quad (3.57)$$

a sufficient condition for optimality is

$$|\nabla f| \leq e_2 \quad (3.58)$$

where  $e_2$  is an arbitrary small number.

For most cases, however, where optimum is constrained, the solution to Zoutendijk's direction finding problem [20] provides a test for optimality. A null solution vector  $u$  indicates a local optimum where all  $\epsilon_j = 0$ . However, where some  $\epsilon_j \neq 0$  and the active constraints are also not zero, the point may be merely near, rather than at the optimum, since in this type of problems the optimum point is usually on the behavior boundary. Therefore, when  $u = 0$  attempts are made to re-start the search and the optimality procedure is invoked only when a non-zero direction vector  $u$  is obtained. In order to re-start the search the step size is redefined as

$$\alpha^{n+1} = \alpha^n/2 \quad (3.59)$$

and therefore the activity limits  $\epsilon_j$  are redefined according to equation (3.36). A new direction  $u$  is then calculated if  $A_j$  has changed. This procedure is repeated until a non-zero direction is found or the minimum step size is encountered, that is

$$\alpha^n \leq \alpha_{\min} \quad (3.60)$$

in which case the search is terminated and the point is assumed to be

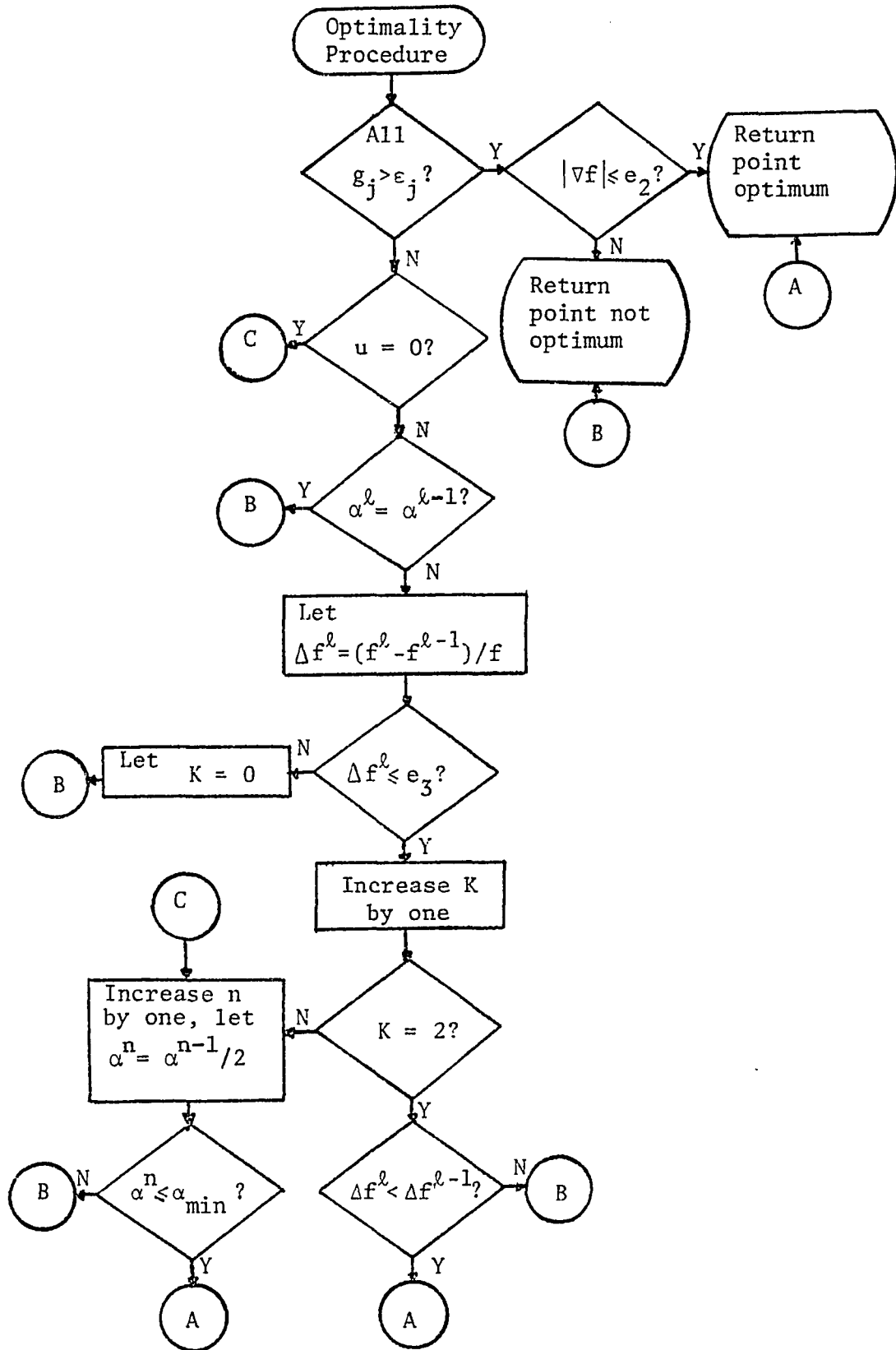


Figure 8 - Flow Chart of the Termination Procedure



an optimum.

For the cases where  $u \neq 0$ , increase  $\ell$  by one and let

$$\alpha^\ell = \alpha^n \quad (3.61)$$

$$f^\ell = f(x^b) \quad (3.62)$$

where  $f^\ell$  is a comparison value with initially  $\ell = 0$ , and  $f^0 = C_1$ , where  $C_1$  is an arbitrary large number. Now if

$$\alpha^\ell = \alpha^{\ell-1} \quad (3.63)$$

no optimality check is made since  $\ell$  has increased because the search was re-started due to the fact that the previous movement direction was unproductive. However, if the step size has changed, this indicates that the maximum improvement in the function has been achieved using the previous step size and the step size must be reduced for additional improvement. A convergence check is now made in order to determine if a search using the new step size is justified. Thus, where equation (3.63) is not satisfied, define

$$\Delta f^\ell = |(f^\ell - f^{\ell-1}) / f^\ell|. \quad (3.64)$$

Unless

$$\Delta f^\ell < e_3 \quad (3.65)$$

where  $e_3$  is the primary convergence criteria, the search is continued.

A secondary convergence check is initiated whenever the primary convergence criteria is met, [equation (3.65) satisfied] in order to confirm optimality. For this purpose the step size  $\alpha$  is reduced according to equation (3.59) and the search is re-started. When further movement with the new step size terminates the primary convergence check is invoked. Therefore, whenever equation (3.65) is not

satisfied at point  $x^b$ , let

$$K = 0 \quad (3.66)$$

and where equation (3.65) is satisfied index  $K$  and let

$$\alpha^{n+1} = \alpha^n/2 \quad (3.67)$$

$$\alpha^{\ell+1} = \alpha^{n+1} \quad (3.68)$$

If equation (3.60) is satisfied, the point is assumed to be a local optimum and the search is terminated. Otherwise a new direction  $u$  is computed and the search continued. When the primary convergence check is satisfied in two consecutive tries, that is if

$$K = 2 \quad (3.69)$$

the secondary convergence check is invoked. Thus if

$$\Delta f^{\ell} < \Delta f^{\ell-1} \quad (3.70)$$

the search is terminated. Otherwise a new direction  $u$  is computed,  $K$  is set equal to 1 and the search continued. When equation (3.70) is satisfied, the change in the value of the objective function ( $\Delta f$ ) has decreased from the previous convergence check, even though a reduced step size was used, the point is thus assumed to be optimum and the search is terminated.

## CHAPTER 4

### DESIGN EXAMPLE

#### 4.1 Problem Statement

Consider the minimum weight design of submersible, circular, cylindrical shells reinforced by equally spaced "T" type frames. The design variables with respect to which the optimization is to be carried out are: Plate thickness, frame web and frame flange thickness, frame flange width, web height, and frame spacing. The objective function to be minimized is the ratio of shell weight to the weight of fluid displaced. All the standard design equations used in submersible shell design practice are to be satisfied. All variables will be treated as continuous. The fixed design parameters are: The operating depth, shell diameter, shell segment length, shell eccentricity, construction material properties, factors of safety to be used in design, maximum and minimum values permitted for the design variables, and, when required, a maximum (when external frames are used) or minimum (when internal frames are used) frame diameter.

This problem is selected here since it is a difficult and computationally demanding engineering problem which the relatively reliable optimization code, DSDA and the popular SUMT procedure, could not solve [17].

In a previous study [22] it has been shown that the merit surface in this problem is fairly flat, therefore a wide range of variables

with similar objective function values may be generated during the search. This study also demonstrates that movement along the behavior boundary is very difficult in this problem.

A modified version of this problem with only four variables was treated in Reference [22], using the DSDA procedure. The formulation of the more difficult six variable problem presented here is similar to that used in Reference [22].

Only CADOP3 [23], a modified version of CADOP2 code, [16] reached an optimal solution to this problem. This code is three times faster than CADOP2 code in problems with behavior constraints. However, due to the difficulties involved in moving along the behavior boundary, the execution time required by CADOP3 to achieve the optimum is quite long (see section 4.3). The problem, therefore, was treated using the BT algorithm in order to demonstrate its superior ability in moving along a difficult behavior boundary.

## 4.2 Problem Formulation

4.2.1 Objective function. For cylindrical vessels with periodic "T" type reinforcing frames the objective function may be written as

$$f(x) = \begin{cases} W/\gamma_w V_D & \text{internal frames} \\ W/[\gamma_w (V_D + V_w + V_f)] & \text{external frames} \end{cases} \quad (4.1)$$

where  $W$  is the weight of the hull segment, given by

$$W = \gamma_s (V_s + V_w + V_f) \quad (4.2)$$

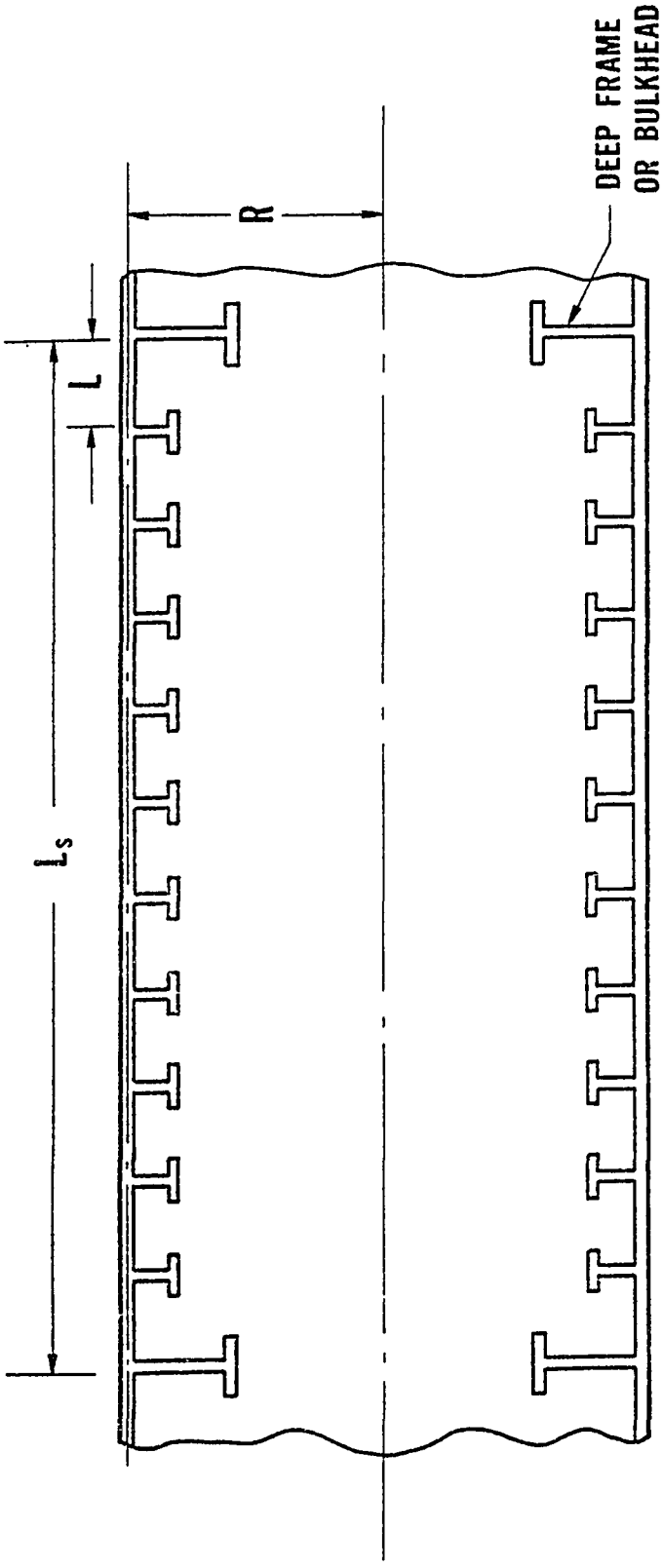
The  $\gamma_S$  and  $\gamma_W$  are the specific weight of the material and immersion fluid, respectively.  $V_S$ ,  $V_W$ ,  $V_f$  are the volume of the plating, frame webs, and frame flanges, respectively, and  $V_D$  is the displacement volume of the cylinder enclosed by the plating envelope. The variables  $x_1, x_2, \dots, x_6$ , which in turn represent the quantities  $t, b, b', w, L',$  and  $h$ , respectively, are defined in Figure 9. The objective function  $f(x)$ , therefore represents the ratio of the shell weight to the weight of the fluid displaced for the shell segment of length  $L_S$ . The weight of the bulkheads is omitted.

4.2.2 Constraint equations. The constraint equations  $g_j(x)$  are divided into two groups: a) Behavior constraints which control the failure modes or impose limitations on the space relationships among the variables, and b) regional constraints which specify ranges of the variables. The basic behavior constraint equations are formulated using a modified version of the design equations given in References [24] and [25].

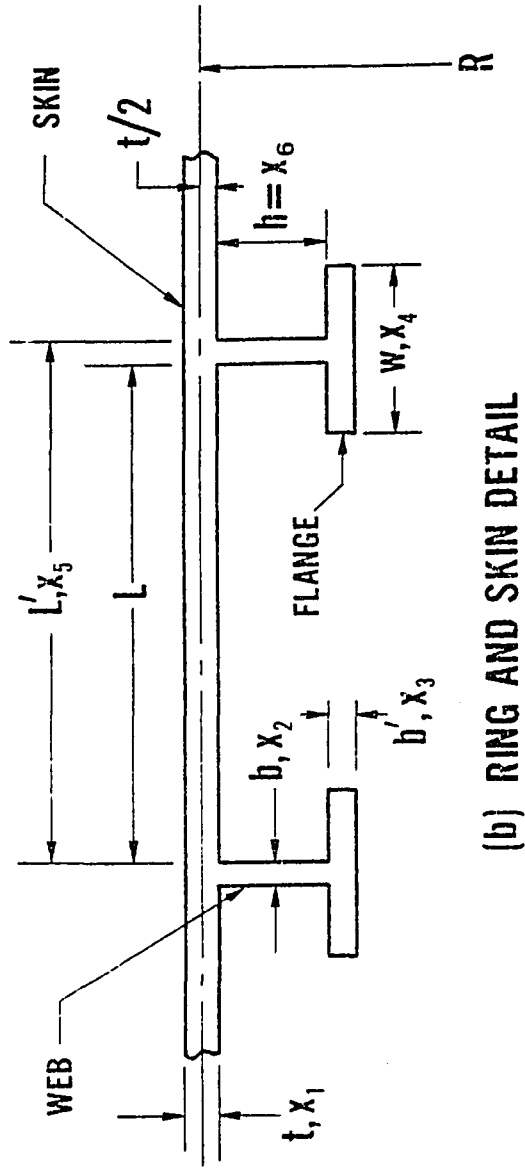
The general instability constraint is given by

$$g_1 = (p_{cg}^* - S_1 p) / p_{cg}^* \geq 0 \quad (4.3)$$

where  $p$  is the applied hydrostatic pressure;  $p_{cg}^*$  is the minimum of the collapse pressure  $p_{cg}(n,m)$  due to general instability [26], and  $S_1$  is the factor of safety for this failure mode. The collapse pressure is given by



(a) SHELL SEGMENT CROSS-SECTION



(b) RING AND SKIN DETAIL

Figure 9 - Typical Shell Cross Section

$$p_{CG}(n,m) = [Dm^2(1 + \beta^2)^2 + \frac{\beta^2 m^2}{L'} (GJ_f + EI_f \beta^2) + \frac{12DZ^2}{\pi^4 m^2} \frac{(1 + F\Lambda_f)}{\Lambda}] \frac{\pi^2}{L_s^2 R(1/2 + \beta^2)} \quad (4.4)$$

where  $p_{CG}(n,m)$  is the buckling pressure for a specified  $n$  and  $m$  and

$$\Lambda_f = 1 + 2n^2 \bar{d} (1 - \beta^2 \mu) + n^4 \bar{d}^2 (1 + \beta^2)^2$$

$$\Lambda = (1 + \beta^2)^2 + 2\beta^2 F (1 + \mu) + \beta^4 F (1 - \mu^2) \quad (4.5)$$

$$Z^2 = L_s (1 - \mu^2) / R^2 t^2$$

$$\beta = nL_s / m\pi R$$

$$F = (bw + bh) / tL'$$

$$D = Et^3 / 12 (1 - \mu^2) \quad (4.6)$$

$$\bar{d} = d / R$$

where  $d$  is the (algebraic) distance to the neutral axis of the frame from the midplane of the hull plating, taken as positive when it is outward from the central axis of the hull [25]. The quantities  $GJ_f$  and  $EI_f$  are the torsional and bending stiffness of the frame, respectively. Symbols representing hull and frame dimensions are defined in Figure 9. The quantities  $G$ ,  $E$ , and  $\mu$  are the shear modulus  $G = E / 2(1 + \mu)$ , tensile modulus, and poisson's ratio, respectively.

Buckling of the shell between frames is controlled by the constraint

$$g_2 = (p_{cs}^* - S_2 p) / p_{cs}^* \geq 0 \quad (4.7)$$

where

$$p_{CS}(n,m) = (\pi^2/L^2R) [Dm^2(1 + \beta^2)^2 + 1/\Lambda(12DZ^2/\pi^2m^2)]1/(1/2 + \beta^2) \quad (4.8)$$

For a given set of design parameters and pressures the buckling pressure functions  $p_{CG}(n,m)$  and  $p_{CS}(n,m)$  depend on the two discrete independent variables, the axial and circumferential wave number  $m$  and  $n$  respectively. To obtain the critical buckling pressure one must find the minimum of  $p_{CG}(n,m)$  and  $p_{CS}(n,m)$  with respect to  $n = 0, 1, 2, \dots$  and  $m = 1, 2, \dots$ . For this purpose the procedure of Reference [26] is used here.

The necessity to avoid plating yield produces the constraint

$$g_3 = (\sigma_{pa} - \sigma_p) / \sigma_{pa} \geq 0 \quad (4.9)$$

where  $\sigma_{pa}$  is the allowable plating stress and

$$\sigma_p = (\sigma_r^2 + \sigma_\phi^2 - \sigma_r\sigma_\phi)^{1/2} \quad (4.10)$$

The quantities  $\sigma_r$  and  $\sigma_\phi$  are found from

$$\sigma_\phi = -pR/t[1 + \Gamma(H_M + \mu H_E)] \quad (4.11)$$

$$\sigma_r = -pR/2t[1 + 2\Gamma H_E] \quad (4.12)$$

The terms  $-pR/t$  and  $-pR/2t$  are the hoop and axial stresses, respectively;  $\Gamma$  is a frame deflection parameter, and  $H_M$  and  $H_E$  are the functions that define the bending effect on the shell due to local frame reinforcing [24]. Equations for  $H_M$ , and  $H_E$  are given in Reference [24].

The constraint against the frame yielding failure mode is given by

$$g_4 = (\sigma_{fa} - \sigma_T) / \sigma_{fa} \geq 0 \quad (4.13)$$



where  $\sigma_{fa}$  is the allowable frame stress and  $\sigma_T$  the maximum frame stress. The quantity  $\sigma_T$  is given by

$$\sigma_T = \sigma_b + \sigma_c \quad (4.14)$$

with the compressive hoop stress  $\sigma_c$  given by

$$\sigma_c = Q_p pR / (A + bt) \quad (4.15)$$

Here  $A$  is the cross-sectional area of the frame and  $Q_p$  is the total radial load per inch of circumference.  $Q_p$  is given by equation (23) of Reference [24]. The frame bending stress  $\sigma_b$  results from slight out-of-roundness of the hull. It is calculated using

$$\sigma_b = [Ece (n^2 - 1) / R^2] [p / (p_{cs} - p)] \quad (4.16)$$

where  $c$  is the distance from the midplane of the shell to the surface of the frame,  $e$  is the eccentricity from the true circle radius, and  $n$  the wave number that minimizes  $p_{cg}$ .

It should be noted that  $\sigma_b$  is discontinuous at points near  $p = p_{cg}$  and is discontinuous with respect to  $n$ , since  $n$  is an integer.

Flange buckling is controlled by letting

$$g_5 = \sigma_b - [.5\pi^2 E / 12 (1 - \nu^2)] [x_3 / (x_4 - x_2)]^2 \geq 0 \quad (4.17)$$

Two geometric constraint equations can be used. A maximum flange thickness is specified to prevent the flange from becoming excessively thick. It is desirable to specify a maximum flange thickness as a fraction of the plating thickness. Thus, the constraint

$$g_6 = (C_2 t - b') / C_2 t \geq 0 \quad (4.18)$$

is used, where  $C_2$  is the maximum flange to plating thickness ratio.

The constant  $C_2$  is made arbitrarily large if the designer does not wish to apply this constraint. Minimum flange width is controlled by the inequality

$$g_7 = (w - C_3 h) / w \geq 0 \quad (4.19)$$

to insure that the flange width is sufficient so that the design rule for the control of the web cribbing is not invalidated,  $C_3$  is an arbitrary constant controlling the minimum ratio of flange width to web height.

It is desirable to limit the minimum or maximum (depending on whether internal or external frames are used) radius of the frame because of space consideration. One can then use the constraint

$$g_8 = (R + h + b' + t/2 - R_{\min}) / R_{\min} \geq 0 \quad (4.20)$$

where  $R_{\min}$  is the minimum specified radius for internal frames or

$$g_8 = (R_{\max} - R - t/2 - h - b') / R_{\max} \geq 0 \quad (4.21)$$

for external frames where  $R_{\max}$  is the maximum specified radius.

To insure against web buckling the following design constraint is invoked.

$$g_9 = \sigma_b - [4\pi^2 E / 12 (1 - \nu^2)] (x_2 / x_6)^2 \geq 0 \quad (4.22)$$

Side constraints of the form

$$\left. \begin{aligned} (x_i^U - x_i) / x_i^U &\geq 0 \\ (x_i - x_i^L) / x_i &\geq 0 \end{aligned} \right\} \quad (4.23)$$

are used to limit the range of the variables  $x_i$  for manufacturing, space, or other practical reasons. The flange width clearly cannot

exceed the frame spacing, imposing an upper limit on  $x_4$ . Sometimes  $x_4$  is limited by design consideration, thus if  $x_4^{U'}$  is the designer specified maximum

$$x_4^U = \begin{cases} x_4^{U'} & , & x_4^{U'} \leq x_5 = L' \\ x_5 & , & x_4^{U'} > x_5 \end{cases} \quad (4.24)$$

The formulation of the example treated here is now complete, but other forms of constraints and objective functions, which do not arise in this example are clearly possible.

### 4.3 Discussion of Results

4.3.1 Code description. A FORTRAN IV code called CADOP4 based on this algorithm was developed and used in this study. The code is operational with IBM FORTRAN levels G and H and the UNIVAC TDOS systems. The user is required to supply the objective and constraint functions, the initial values of the variables, and side constraint values.

The initial step size is internally generated so that at the starting point a step of size  $\alpha^0$  in the  $\nabla f$  direction produces a one percent change in the value of the objective function. This step is constrained so that  $\alpha^0 > .01$ . The minimum step size is defined as  $\alpha_{\min} = \alpha^0 / 1000$ . The program uses  $e_1 = 10^{-5}$ ,  $e_2 = e_3 = 10^{-6}$ ,  $N_1 = 6$ ,  $N_2 = 10$ ,  $C_1 = 10^5$  and the activity limit constant is initially specified as  $K_j = 0.01$  for all constraints.

Nondimensional constraint equations of the form

$$g_j(x_i) = (U_k - B_k) / U_k \geq 0 \quad \text{when } U_k \neq 0 \text{ and } B_k \neq 0$$

or

$$g_j(x_i) = (B_k - L_k) / L_k \geq 0 \quad \text{when } L_k \neq 0 \text{ and } B_k \neq 0$$

(4.25)

are used. Otherwise, a dimensional form of equations is used.

A constraint given as  $0 \leq b(x_i) \leq A$  would be written as two constraint equations. For example, one could be of the form of the first of equations (4.25) where  $B_1 = b(x_i)$  and  $U_1 = A$ . The second would be of the form  $g_2 = -b(x_i)$  since  $L_k = 0$ .

4.3.2 Code application. The above problem was treated using CADOP3 and CADOP4 codes. Except that the initial step size is specified for this problem as  $\alpha = 4/R$ , where  $R$  is the radius of the shell. The minimum step size is defined as  $\alpha_{\min} = \alpha^0/200$ . Runs were made on an IBM 370 model 168 using FORTRAN level G. The same starting points were used in these runs. The execution time in seconds along with other necessary information were printed at each base point in order to closely track the synthesis path of each procedure.

Tables 1 and 2 present the results obtained from CADOP3 and CADOP4 codes, respectively. A total of 72810 function evaluations were performed using CADOP3 code prior to termination. Due to the great execution time required by CADOP3 for this problem, the run was terminated prior to reaching the optimum. The execution time for this run was 56.2 seconds. CADOP4 code required only about 4000 function evaluations with 4.8 seconds execution time to reach a similar

level of convergence.

The constraint preventing the web buckling ( $g_9$ ) was active from near the beginning of the search, while the plate yielding ( $g_3$ ) became active after a relatively short time and remained active through the rest of the search. The general buckling constraint was also active at the latter part of the search. Thus, the optimal design is constrained by general instability, local web buckling and plating yield.

The CADOP3 code came within 1% of the optimum point after 23 seconds and approximately 43000 function evaluations, while CADOP4 required only 1.4 seconds and about 1000 function evaluations to reach this level of convergence. It should be pointed out that for most practical engineering purposes, a 1% level of accuracy is quite acceptable. It is the refining process that is responsible for the greatest portion of the total execution time.

It must be pointed out that due to the presence of intermediate print statements, the execution time required for solution without these statements is appreciably less than the above mentioned time. A run made without any intermediate print statements using CADOP4 code terminated after 2.1 seconds at the optimum point while the original run required 8.7 seconds.

Comparison of the above results demonstrate the apparent superiority of the Boundary Tracking algorithm to the DSFD procedure with respect to speed. This is the result of great reduction in the number

TABLE 1  
SYNTHESIS PATH FOR THE SHELL DESIGN PROBLEM USING DSFD METHOD

No. of Bases	Obj. Function Value	x <sub>1</sub>	x <sub>2</sub>	x <sub>3</sub>	x <sub>4</sub>	x <sub>5</sub>	x <sub>6</sub>	No. of Obj Function Evaluations	No. of Constraint Function Evaluations	Constraints Active	CPU Time in Sec.
0	.191149	1.	.5	.5	10	10	20	3	22		.02
50	.131655	.91745	.39056	.21840	6.5253	12.9584	19.83179	1510	3209	3, 9	2.13
100	.1128944	1.19171	.35006	.13070	4.20835	27.60905	16.13659	2657	6332	3	4.21
150	.10910318	1.19397	.29135	.11307	3.2443	24.03268	13.11856	4044	9974	3	6.52
200	.108462	1.19684	.28109	.10376	2.92241	23.7068	12.60682	6782	14494	3	9.96
250	.1081338	1.1984	.27466	.10326	2.6263	23.3650	12.27031	8519	19026	3	13.37
300	.107774	1.19696	.2675	.10326	2.26339	22.437	11.887	10529	23145	3	16.67
350	.107417	1.20908	.25968	.10326	1.88243	23.6244	11.48603	12621	27486	3	20.29
400	.107051*	1.20184	.25190	.10326	1.44585	21.4827	11.0595	14621	32447	3	26.99
450	.106670	1.16294	.22833	.20389	1.02888	16.8998	10.1515	15925	35804	3, 9	31.79
500	.106545	1.1822	.22409	.25339	1.03081	16.62096	9.9441	17511	39112	3	37.51
550	.106497	1.18498	.21779	.29590	1.00365	16.785	9.7805	19730	43882	1, 3, 9	46.46
600	.106483	1.18528	.21826	.34346	.93264	16.7932	9.70967	21578	47987	1, 3, 9	52.5
623	.106477**	1.18425	.21752	.35553	.89695	16.57193	9.67398	22580	50230	1, 3, 9	56.21

\*Converged within 1% of the optimum point after approximately 375 bases.

\*\*Within .2% of the optimum point.

TABLE 2

## SYNTHESIS PATH FOR THE SHELL DESIGN PROBLEM USING THE BOUNDARY TRACKING METHOD

No. of Bases	Obj. Function Value	x <sub>1</sub>	x <sub>2</sub>	x <sub>3</sub>	x <sub>4</sub>	x <sub>5</sub>	x <sub>6</sub>	No. of Obj Function Evaluations	No. of Constraint Function Evaluations	Constraints Active	CPU Time in Sec.
Starting Point	.191149	1.0	.5	.5	10.0	10.0	20.0	1	10		
1	.146414	.95870	.37494	.37567	9.93493	11.918	19.93510	19	95	9	.11
10	.118556	1.3393	.37116	.27278	8.52995	56.78	18.52975	54	401	2, 9	.52
20	.11011	1.26089	.26746	.12878	2.20746	30.388	12.20792	81	664	9	.98
30	.10726*	1.20859	.25749	.10027	1.45954	23.173	11.3766	134	1085	3	1.63
40	.106886	1.17902	.24437	.10019	.74477	17.08	10.6618	192	1475	3	2.29
50	.106721	1.18417	.23362	.38917	.67781	17.49	10.0889	242	1822	3	2.98
60	.106652	1.18309	.22614	.471329	.75997	17.08	9.7476	316	2288	3	3.89
70	.106596	1.1837	.21881	.55404	.8405	16.99	9.4114	390	2766	3	4.81
80	.106535	1.1777	.21182	.57461	.85095	15.7	9.0787	456	3204	3	5.61
90	.106400	1.18178	.20257	.94152	.62714	15.868	8.6507	514	3603	1, 3	6.37
100	.106286	1.18286	.19107	1.0497	.60277	15.572	.503	614	4318	1, 3, 9	7.72
110	.106283	1.18246	.190911	1.05799	.59457	15.485	8.4948	632	4477	1, 3, 9	8.01
120	.1062828	1.18199	.19072	1.0673	.58527	15.384	8.4855	650	4636	1, 3, 9	8.27
130	.106275**	1.18188	.19068	1.0693	.58319	15.361	8.4834	684	4869	1, 3, 9	8.72

\*Converged within 1% of the optimum point after approximately 25 bases.

\*\*Optimum point.

of function evaluations required for convergence. It demonstrates the ability of the BT procedure to move efficiently along a difficult behavior boundary.



## CHAPTER 5

### COMPARISON STUDY

The CADOP4 code as described in section 4.3.1 was used in this study along with all the constants and parameters presented earlier.

This comparison study is based on the works of Eason and Fenton given in References [14] and [15]. All ten problems treated in these references were run, using the new code with the starting points given in Reference [14]. All the control constants and the step sizes are internally generated so that there was no special tuning for individual problems. Computations were performed in double precision on an IBM 370 model 168 system, using a level G compiler, thus closely simulating the Eason and Fenton study.

A brief description of the codes studied are given in Table 3, while Table 4 contains the data for rating the success of the various codes in solving the ten problems to which they were applied. A detailed description of the problems is given in Reference [14]. A normalized time required for the solution of each problem successfully solved is given in Table 4. Normalized time is defined here as the execution CPU time divided by the CPU time required to execute a timing standardization program [14]. The symbol "P" denotes progress toward a solution, and a blank denotes failure. The criteria used in References [14] and [15] for defining successful solution or progress were also applied here.

TABLE 3  
CODE DESCRIPTIONS

Name	Description	Algor Class*
ADRANS	Random search followed by pattern moves	DS
CLIMB	Rosenbrock search	DS
DAVID	Davidon-Fletcher-Powell with numerical derivatives	GF
DFMCG	Fletcher-Reeves conjugate gradient method with secant approximation derivatives	G
DFMFP	Davidon-Fletcher-Powell with secant approximation derivatives	G
FMIND	Hook & Jeeves pattern search	DS
GRAD4	Steepest descent method	G
GRID1	Grid and star network search, with shrinkage	AR
MEMGRD	Davidon-Fletcher-Powell with retained step length information	GF
NMSERS	Simplex search	DS
PATSH	Modified pattern search, dome strategy	DS
PATRNI	Modified pattern search, ridge strategy	DS
RANDOM	Random search with shrinkage	AR
SEEK1	Pattern search followed by random search	DS
SEEK3	Modified pattern search	DSF
SIMPLX	Modified simplex search	DSF
DSDA	Modified pattern search followed by Mugele's search	DS
CADOP2	Modified pattern search followed by Zoutendijk feasible direction method	DS
CADOP3	Modified CADOP2	DS
CADOP4	Boundary Tracking method	BT

\*DS = direct search, DSF = direct search employing SUMT strategy and penalty function, G = gradient procedure, GF = gradient procedure using SUMT strategy and penalty function, AR = area reduction method, BT = Boundary Tracking.

TABLE 4  
PERFORMANCE OF OPTIMIZATION CODES\*

Problem No.	1	2	3	4	5	6	7	8	9	10
Variables	5	3	5	4	2	2	3	2	2	4
Constraints**	10	2	6	0	0	1	2	0	6	0
Code Names										
ADRANS	2.64	0.458	1.65	P	0.100	0.654	0.159	0.069	P	P
CLIMB				0.015	0.005			0.007		
DAVID		0.188	0.84	1.132	0.075	0.046				
DFMCG			P	0.015	P			0.004		
DFMFP			P	0.038	0.250		P	0.387		
FMIND		0.004	P		0.140		0.003	0.003	P	3.74
GRAD4		P	P	0.283	0.393		P	0.004		P
GRID1	P	P		0.033	P			0.037	P	P
MEMGRD				0.059	0.066		P			
NMSERS		0.143	0.91	0.060	0.002	0.067		0.007	0.39	P
PATSH	1.78	0.019	0.045	0.013	0.018	0.012	P	0.009	P	3.54
PATRNO	P	0.220	1.00	0.021	P	0.020	0.010	0.002	P	1.95
PATRN1	P	P	P	0.008	0.002			0.001	1.02	1.20
RANDOM		P		P		P		0.013		P
SEEK1	P	P	P		0.010	0.024	1.20	0.007	P	1.53
SEEK3		0.102	0.14	0.212	0.035	0.191	0.010	0.007		
SIMPLX	2.97	0.297	1.31	0.196	0.035	0.191	0.141	0.013	4.20	P
DSDA	P	0.021	0.047	0.104	0.003	0.008	0.262	0.050	P	
CADOP2	0.307	0.016	0.090	0.075	0.008	0.011	0.145	0.004	1.94	1.35
CADOP3	0.119	0.006	0.030	0.080	0.004	0.011	0.069	0.003	1.55	1.73
CADOP4	0.044	0.003	0.0125	0.080	0.004	0.005	0.025	0.005	1.00	1.60
							0.003	0.005	0.39	1.60

\*Numerical entry indicates normalized time required for solution, and P indicates progress toward a solution [14, 15].

\*\*Excluding regional constraints.

The important variables effecting execution time are; the problem, the algorithm, the code, the input and output requirements, the architecture of the machine and the compiler system. In order to minimize the effects of those variables not involved in the comparison, similar computer machines and compiler systems along with the same test problems and output requirements are used here. Unfortunately the effects of the code structure can not accurately be determined. Thus, data of Table 4 and ratings shown in Table 5 can not be considered to accurately reflect the effectiveness of the basic optimization algorithms.

The data and rating procedures used here however, can be directly applied, with reasonable accuracy, for relative comparison of the codes tested. The results obtained are thus useful to the user of such codes. The relative effectiveness of basic optimization algorithms can only be inferred from this data if one assumes that the codes for each procedure have been prepared with essentially comparable effectiveness.

This study uses Eason and Fenton's comparison procedures since they are the best available and the most current. Furthermore, use of another investigator's comparison methods rather than one proposed by the developer of a new method reduces the tendency toward developer bias in reporting on the effectiveness of his method.

The data from Table 4 may be applied to a number of rating methods for comparing the codes tested. Table 5 presents relative ranking of

TABLE 5

RELATIVE RANKING OF OPTIMIZATION CODES

	Generality		Efficiency		Generality and Efficiency	
	$n_a$	$N_a$	$f_a$	$F_a$		$T_a$
10	CADOP4 CADOP3 CADOP2	CADOP4 CADOP3 CADOP2	10	1.3 PATRNI	0.18 NMSERS 0.19 CADOP4	2.2 CADOP4 2.9 CADOP3 4.1 CADOP2
9	DSDA PATSH	DSDA PATSH	9.5	2.5 CADOP4	0.26 CADOP3 0.28 PATRNI	9 DSDA
8	SEEK3 SIMPLX	SEEK3 SIMPLX ADRANS	8.5	3.8 CADOP3	0.40 CADOP2	15 PATSH
7	ADRANS NMSERS	ADRANS NMSERS	8.0	4.1 NMSERS	0.54 DSDA	16 NMSERS PATRNI
5	PATRNI FMIND MEMGRD DAVID	PATRNI FMIND SEEK1	7.0 6.0	6.6 CADOP2 9.5 DSDA 15.7 FMIND	0.82 MEMGRD 0.85 PATSH 0.97 FMIND	18 SEEK3 21 SIMPLX 24 ADRANS
4	SEEK1	MEMGRD PATRNO	5.5	23.8 PATSH	1.2 SEEK3	25 FMIND
3	CLIMB DFMFP GRAD4 PATRNO RANDOM	DAVID GRAD4 GRID1 RANDOM	5.0	24.7 SEEK3 34.8 MEMGRD 61.1 SIMPLX	1.9 SIMPLX 2.4 DAVID 2.9 ADRANS	26 MEMGRD 27 DAVID
2	DFMCG GRID1	DFMFP CLIMB DFMCG	4.0 3.0	63.6 DAVID 92.5 ADRANS		

the codes using the criteria of Reference [15]. The rating equations are as follows:

Let the number of problems solved by code "a" be  $n_a$ , and let  $n'_a$  be the number of problems with a "P" rating. Then the numerical success rating  $N_a$  is given by

$$N_a = n_a + n'_a/2 \quad (5.1)$$

One of two computational efficiency ratings is given as

$$f_a = [\sum_{p=1}^{10} b_{ap} t_{ap} / \min(t_{ap})] / n_a \quad (5.2)$$

where  $b_{ap} = 1$  if code "a" solved problem "p" and zero otherwise,  $t_{ap}$  is the normalized time required for solution and  $\min(t_{ap})$  is the shortest time required by any of the codes studied to solve problem "p". The other efficiency criterion is

$$\bar{f}_a = [\sum_{p=1}^{10} b_{ap} t_{ap} / \text{mean}(t_{ap})] / n_a \quad (5.3)$$

where  $\text{mean}(t_{ap})$  is the average time required by the codes studied to solve problem "p". An overall rating number which can be considered a composite measure of generality (reliability) and efficiency (speed) is given by

$$T_a = \sum_{p=1}^{10} t'_{ap} \quad (5.4)$$

where  $t'_{ap}$  is set equal to  $t_{ap}$  if algorithm "a" solves a problem "p", and to twice the time used by the slowest code solving a problem "p" if code "a" could not solve it. This penalty time is used to penalize code unreliability. Only codes that solved half or more of the problems are rated for efficiency.

The tables presented here are similar to those given in References [15] and [17], except that the CADOP3 and CADOP4 codes are included.

The CADOP3 code is a modified version of CADOP2 code presented in Reference [17]. Thus, the rating methods and presentation of results used are essentially those of Reference [15] and [17].

From Tables 4 and 5 it can be concluded that CADOP4 is the fastest code in overall generality and efficiency rating. This Boundary Tracking method presented here appears comparable in speed to the fastest methods presented in Reference [17], (NMSERS, PATRNI, DSDA, and CADOP2). CADOP4 is faster than average in all problems. It is the fastest code solving problems 1, 2, 3, 6, 7, and 9, which are problems with behavior constraints. On the other hand, problems 4, 5, 8, and 10, where the CADOP3 and CADOP4 codes performed identically, but less efficiently than some other procedures, are problems without such constraints. This is expected since the two codes are identical in treating the unconstrained problems.

Of the problems tested, problem 1 which has the largest number of behavior constraints (10) was apparently also the most difficult, since it was solved by only six of the twenty-one codes presented in Table 4. The second most difficult problem was problem 9, for which only 7 of the 21 codes led to the optimum point. This problem has only 6 behavior constraints and two variables. However, there is a rapid change in the slope of the behavior boundary in the region near the optimum point. Because of this rapid variation in slope, a major part of the total execution time was spent on refining the location of the point so as to meet the convergence criterion. Problem 10, which required the longest execution time, has no behavior constraints. The long execution time

required by this problem is due to the complexity of the objective function and the number of the function evaluations needed to reach the optimum.

It can be seen from Table 4 that the CPU time required for treating problems 9 and 10 in most cases is much greater than the sum of the CPU times required for the solution of the rest of the problems tested. Thus, the speed of a code in solving problems 9 and 10 plays an overwhelming role in determining the code's overall generality and efficiency rating,  $T_a$ . For example, a code that is very efficient on the majority of problems but requires long solution times for problems 9 and 10 would have a relatively low overall generality and efficiency rating using equation (5.4). Therefore, this rating method is not very useful for comparing codes solving all the test problems. For such codes the efficiency ratings  $f_a$  and  $\bar{f}_a$  represent the ability of the method more clearly. Furthermore, the overall generality and efficiency rating  $T_a$  does not sufficiently penalize those codes which failed to solve any of the test problems 1 through 8, but performed well on problems 9 and 10. The penalty time used here fails to accurately take into account the weaknesses of such codes in solving problems 1 through 8.

The superior performance of CADOP4 code in problems with behavior constraints compared to CADOP3 code is due to the fact that the number of function evaluations required in CADOP3 is proportional to the number of bases needed for solution multiplied by the number of variables. On the other hand, the number of function evaluations in CADOP4 is



proportional only to the number of base points, and does not have to be multiplied by the number of variables. Therefore, in CADOP3 the number of function evaluations increases directly with the number of variables, while in the CADOP4 code it does not.

Compared to the relatively reliable codes ( $n_a \geq 8$ ), CADOP4 is faster than CADOP3 on six of the ten problems solved by CADOP3, faster than DSDA on six of nine, and faster than PATSH on eight of nine, and faster than SEEK3 in all problems solved by these schemes. As may be seen, the CADOP2 and CADOP3 codes are the only reliable codes comparable to CADOP4. CADOP4 is significantly faster than CADOP3, CADOP2, PATSH, SEEK3, and SIMPLX. The straight pattern (PATRN1) or simple (NMSERS) codes are ranked relatively high in efficiency primarily due to their superior performance in unconstrained problems (Table 5). Of this group, only NMSERS appears to be sufficiently reliable to merit consideration for use. The minor difference in efficiency between NMSERS and CADOP4 is, however, overshadowed by the superior reliability of the latter. Thus, in the overall speed, generality, and efficiency rating, CADOP4 stands out. Viewed on the basis of these comparisons, CADOP4 appears to be a superior nonlinear mathematical programming code.

The new algorithm is intended to be a constrained optimization algorithm. Therefore, the performance of CADOP4 is compared to the codes of References [15] and [17] on problems having behavior constraints. This comparison is shown in Table 6.

Table 6 is similar to Table 5 except that the problems without

TABLE 6

RANKING OF OPTIMIZATION CODES IN PROBLEMS WITH BEHAVIOR CONSTRAINTS

	Generality		Efficiency		Generality and Efficiency	
	$n_a$	$N_a$	$f_a$	$\bar{E}_a$	$T_a$	$\bar{E}_a$
6	CADOP4 CADOP3 CADOP2	CADOP4 CADOP3 CADOP2	1.0 CADOP4	0.07 CADOP4	0.46 CADOP4 1.2 CADOP3	
5	DSDA PATSH SEEK3 SIMPLX ADRANS	DSDA PATSH SIMPLX ADRANS	3.2 CADOP3 3.3 NMSERS 8 CADOP2	0.15 NMSERS 0.19 CADOP3 0.43 CADOP2 0.83 DSDA	2.0 CADOP2 8.1 DSDA 10.7 SEEK3	
	4	NMSERS	13 DSDA	1.0 PATSH	11.4 PATSH	
	3	MEMGRD DAVID	4.5 NMSERS 3.5 MEMGRD	28 SEEK3 39 PATSH 43 MEMGRD 63 DAVID	1.1 MEMGRD 1.2 DAVID 1.3 SEEK3	13.4 SIMPLX 14 ADRANS 17 NMSERS
2	FMIND RANDOM	PATRN1 FMIND SEEK1	78 SIMPLX 10.4 ADRANS	2.1 SIMPLX 3.2 ADRANS	18 MEMGRD 18 DAVID	
1	PATRN1 SEEK1	DAVID RANDOM				
0	CLIMB DFMFP GRAD4 PATRNO DFMCG GRID1	PATRNO GRID1 GRAD4 PATRNO DFMCG GRID1				
	1	DFMFP				
	0.5	DFMCG				
	0	CLIMB				

behavior constraints are excluded. The rating equations used for these values are the same as those used for Table 5. The results of Table 6 indicate the superiority of CADOP4 with respect to speed, generality, and efficiency. Only codes that solved half or more ( $n_a \geq 3$ ) of the problems are rated for efficiency.

The superior performance of the CADOP4 code in all the above ratings strongly suggests the superiority of the BT method for treating problems with behavior constraints. CADOP4 proved to be substantially more efficient than the CADOP3 and NMSERS codes on such problems. In the overall generality and efficiency rating CADOP4 code again stands alone with CADOP3 code in the second place being approximately three times slower. However, as explained earlier, due to the relatively long execution time required for problem 9 this rating method does not reflect accurately the relative effectiveness of the codes which solved all the test problems. The efficiency ratings  $f_a$  and  $\bar{f}_a$  may again be more useful in comparing such codes. In these ratings CADOP4 code is two to three times faster than other relatively fast codes (CADOP3, NMSERS).

Thus, the Boundary Tracking algorithm presented here appears to be superior to DSFD with regard to efficiency and to all other optimization procedures tested with respect to generality and efficiency.

Based on its performance on the ten problems of Reference [15] and on the other comparison studies, the Boundary Tracking method appears to be a fast and reliable nonlinear mathematical programming optimization procedure.

In conclusion, it must be pointed out that the test problems used in the above comparison study are relatively small and simple, while many practical engineering problems are large and contain complex and computationally demanding functions. Therefore, although it is reasonable to assume that the above comparison study is representative of the performance of the new code on most practical problems of size and complexity similar to that of the study, this performance may not be representative for large complex problems. Unfortunately, no comparison study utilizing large complex problems is available, and is not likely to be available soon due to the difficulty and cost associated with such a project. Furthermore, one cannot guarantee that the performances noted in the present study are typical for all relatively small simple problems. Nevertheless, this is the most complete of the available general comparison studies and should be useful as a guide in selecting a suitable algorithm for a particular problem. The designer must, however, proceed with care in comparing his problem to the above test problems, taking into account such things as the number of variables, constraints, nature of the function to be evaluated, etc. in selecting a desirable algorithm.

## CHAPTER 6

### CONCLUSION

The successful application of the CADOP4 code to the relatively complex problem of shell synthesis and its performance in the general comparison study presented in section 4.1, imply that the Boundary Tracking method developed here is a superior nonlinear mathematical programming procedure. Although the code proved to be the best in the general comparison study, the real potentials of the algorithm are demonstrated in the shell design problem. Since many engineering problems are of such a class, the new algorithm shows the promise of adding a major contribution to the field of automated design.

In the above studies the BT method proved to have great potential for use in computationally demanding problems. However, as in the case of most new methods, additional studies may lead to improvement, particularly in the speed of the algorithm.

The only apparent disadvantage of CADOP4 is its relative complexity compared to some of the other reasonably reliable methods. It contains 1320 FORTRAN statements, while, for example, the CADOP3 contains 790, DSDA contains 372, and PATSH only 75 FORTRAN statements. Thus, if CADOP4 code is not available in a compiled form, for simple problems of relatively low complexity and dimensionality, one of the simpler codes may be preferable since the time required for compilation of the program may exceed the time saved by using CADOP4.

In addition, its performance on problems without behavior constraints, although quite good, is not as outstanding as on problems with such constraints.

In conclusion, the user of this or any other algorithm for automated design, should be aware that the general nonlinear constrained optimization problem is quite difficult to handle. Also, none of the available techniques will guarantee an optimum solution. One should, therefore, be careful in the application of such an algorithm and analyze the results thoroughly. One also, should make use of several synthesis runs, using different starting points where possible, before assuming the value is an optimum solution.

LIST OF REFERENCES

1. Shen, C. Y., and Prager, W., "Recent Developments in Optimal Structural Design," Appl. Mech. Rev., Oct. 1968, pp. 985-992.
2. Wasintynski, Z., and Brandt, A., "The Present State of Knowledge in the Field of Optimum Design of Structures," AMR, 16, 1963, pp. 341-350.
3. Parewonesky, P., "Optimal Control; A Review of Theory and Practice," AIAA J., Nov. 1965, pp. 1985.
4. Gerard, G., "Optimum Structural Design Concepts for Aerospace Vehicles," J. Spacecraft and Rockets, Jan. 1966, pp. 5-18.
5. Hadley, G., Linear Programming, Addison-Wesley, 1964.
6. Hadley, G., Nonlinear and Dynamic Programming, Addison-Wesley, 1964.
7. Graves, R. L., and Wolf, P., Recent Advances in Mathematical Programming, McGraw-Hill, 1963.
8. Schmit, L. A., and Mallet, R. H., "Structural Synthesis and Design Parameter Hierarchy," J. Struct. Div. Am. Soc. Civil Engrs., Aug. 1963, pp. 269-299.
9. Schmit, L. A., Kicher, T. P., and Morrow, W. M., "Structural Synthesis Capability for Integrally Stiffened Waffle Plates," AIAA J., Vol. 1, No. 12, Dec. 1963, pp. 2820-2836.
10. Schmit, L. A., "Automated Design," International Science and Technology, No. 54, June 1966, pp. 63-78 and 115-117.
11. Mangasarian, O. L., "Techniques of Optimization," J. of Engineering for Industry, Trans, ASME, Series B, Vol. 94, No. 2, May, 1972, pp. 365-372.
12. Wilde, D. J., Optimum Seeking Methods, Prentice Hall, 1964.
13. Abaide, J. Nonlinear Programming, North Holland Pub. Co. 1967.
14. Eason, E. D., and Fenton, R. G., "Testing and Evaluation of Numerical Methods for Design Optimization," UTIME-TP 7204, University of Toronto, Canada, 1972, 1972.
15. Eason, E. D., and Fenton, R. G., "A Comparison of Numerical Optimization Methods for Engineering," J. of Engineering for Industry, TRANS. ASME, Series B, Vol. 96, No. 1, Feb. 1974, pp. 196-201.

16. Moradi, J. Y., "A Direct Search Mathematical Programming Algorithm," Master's Thesis, Newark College of Engineering, 1974.
17. Pappas, M., and Moradi, J. Y., "An Improved Direct Search Mathematical Programming Algorithm," J. of Engineering for Industry, Vol. 97, No. 4, Nov. 1975, pp. 1305-1310.
18. Pappas, M., "Performance of the 'Direct Search Design Algorithm' as a Numerical Optimization Method," AIAA Journal, Vol. 13, No. 6, June 1975, pp. 827-829.
19. Shampine, L. F., and Allen, R. C., Numerical Computing; an Introduction, Saunders Co. 1973.
20. Zoutendijk, G., Methods of Feasible Directions, Elsevier, Amsterdam, 1960.
21. Wilde, O. L., and Beightler, C. S., Foundation of Optimization, Prentice-Hall, Englewood Cliffs, N. J. 1967.
22. Pappas, M. and Allentuch, A., "Automated Optimal Design of Frame Reinforced, Submersible, Circular, Cylindrical Shells," J. of Ship Research, Vol. 17, No. 4, Dec. 1973, pp. 206-216.
23. Pappas, M., "An Improved Direct Search-Feasible Direction Optimization Algorithm," N.J.I.T. Report No. NV 10, 1977.
24. MacNaught, D. F., "Strength of Ships," Principles of Naval Architecture, SNAME, 1967, Chap. IV, Section 8.
25. Pappas, M. and Allentuch, A., "Pressure Hull Optimization Using General Instability Equation Admitting More than One Longitudinal Buckling Half-Wave," J. of Ship Research, Vol. 19, March 1975, pp. 18-22.
26. Pappas, M. and Amba-Rao, C., "A Discrete Search Procedure for the Minimization of Stiffened Cylindrical Shell Stability Equations," AIAA Journal, Vol. 8, No. 11, Nov. 1970, pp. 2093-2094.



## VITA

Jacob Moradi

- 1968      Graduated Adib High School, Tehran, Iran.
- 1970      Attended Newark College of Engineering, Newark, New Jersey.
- 1973      B.S.M.E., Newark College of Engineering.
- 1973-74   Graduate work in Mechanical Engineering, Newark College of Engineering.
- 1974      M.S.M.E., Newark College of Engineering.
- 1974-77   Graduate work in Mechanical Engineering, New Jersey Institute of Technology, Newark, New Jersey.
- 1977      Doctor of Engineering Science in Mechanical Engineering.