THESIS

NUMERICAL OPTIMIZATION ALGORITHM
FOR ENGINEERING PROBLEMS
USING MICROCOMPUTER

by

Dong Soo, Kim

September 1984

Thesis Advisor: G. N. Vanderplaats

Approved for public release; distribution unlimited.
**Title:** Numerical Optimization Algorithm for Engineering Problems Using Microcomputer

**Author:** Dong Soo, Kim

**Type of Report & Period Covered:**
Master's Thesis; September 1984

**Performing Organization Name and Address:**
Naval Postgraduate School
Monterey, California 93943

**Report Date:** September 1984

**Number of Pages:** 57

**Abstract:**
A general purpose computer program is developed to perform nonlinear constrained optimization of engineering design problems. The program is developed especially for use on microcomputers and is called Microcomputer Software for Constrained Optimization Problems (MSCOP). It will accept a nonlinear objective function and up to 50 inequality constraint functions and up to 20 bounded design variables.
MSCOP employs the method of feasible directions. Although developed for microcomputers, for speed of development, the MSCOP was implemented on an IBM 3033 using standard basic language, Waterloo BASIC Version 2.0. It is directly transportable to a variety of microcomputers.

Typical applications of MSCOP program are in the design of machine components and simple beam and truss structures. Solutions to three sample problems are given.
Numerical Optimization Algorithm
for Engineering Problems
Using Micro-computer

by

Dong Soo, Kim
Major, Republic of Korea Army
B.S., Korea Military Academy, 1976
B.E., Seoul National University, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1984

Author

Approved by:

Carret N. Vanderplaats, Thesis Advisor
R. Kevin Wood, Second Reader
Alan R. Washburn, Chairman,
Department of Operations Research
Kneale T. Marshall,
Dean of Information and Policy Sciences
ABSTRACT

A general purpose computer program is developed to perform nonlinear constrained optimization of engineering design problems. The program is developed especially for use on microcomputers and is called Microcomputer Software for Constrained Optimization Problems (MSCOP). It will accept a nonlinear objective function and up to 50 inequality constraint functions and up to 20 bounded design variables.

MSCOP employs the method of feasible directions. Although developed for microcomputers, for speed of development, the MSCOP was implemented on an IBM 3933 using standard basic language, Waterloo BASIC Version 2.0. It is directly transportable to a variety of microcomputers.

Typical applications of MSCOP program are in the design of machine components and simple beam and truss structures. Solutions to three sample problems are given.
# TABLE OF CONTENTS

I. INTRODUCTION ........................................... 10
   A. PURPOSE ............................................. 10
   B. IMPLEMENTATION ..................................... 11
   C. GENERAL OPTIMIZATION MODEL ....................... 11
   D. ORGANIZATION OF THIS THESIS ...................... 12

II. OPTIMIZATION ALGORITHM ............................... 14
   A. INTRODUCTION ........................................ 14
   B. SEARCH DIRECTION ...................................... 16
      1. Usable-Feasible Direction ....................... 17
      2. Active Constraints ............................... 18
      3. Suboptimization Problem and Push-Off Factors 18
      4. Simple Simplex-like Method for Search Direction 20
      5. Initially Infeasible Designs ..................... 22
   C. CWE-DIMENSIONAL SEARCH ............................. 24
      1. No Violated Constraints ......................... 24
      2. One or More Constraints Violated ............... 25
   D. CONVERGENCE CRITERIA ............................... 26

III. MSCOP USAGE ............................................. 27
   A. INTRODUCTION ........................................ 27
   B. PROBLEM FORMULATION ............................... 27
   C. PROBLEM ENTRY ....................................... 28

IV. EXAMPLE PROBLEMS ....................................... 31
   A. DESIGN OF CANTILEVERED BEAM ....................... 31
      1. Uniform Cantilevered Beam ....................... 31
      2. Variable Cantilevered Beam ..................... 33
LIST OF TABLES

I. The Solution of a Uniform Cantilevered Beam . . . 33
II. The Solution of a Variable Cantilevered Beam . . . 37
III. The Solution of a 5-Bar Truss . . . . . . . . . . . . 41
### LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Algorithm for the Feasible Direction Method</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>Usable-Feasible Direction</td>
<td>17</td>
</tr>
<tr>
<td>2.3</td>
<td>Push-Off Factor and Bounding of the S-Vector</td>
<td>19</td>
</tr>
<tr>
<td>4.1</td>
<td>Design of a Uniform Cantilevered Beam</td>
<td>31</td>
</tr>
<tr>
<td>4.2</td>
<td>Design of a Variable Cantilevered Beam</td>
<td>34</td>
</tr>
<tr>
<td>4.3</td>
<td>Design of a 5-Bar Truss</td>
<td>38</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENT

I am very grateful to Professor Garret W. Vanderplaats whose expert advise, technical support and guidance resulted in my much understanding of engineering optimization.

I also wish to express my sincere appreciation to Professor R. Kevin Wood for his critical review and comments during the preparation of this thesis.

The author extends a special thanks to Dr. Noriyaki Yoshida for his assistance, his time and his patience during the course of work.

Finally, to my wife, Jong Soon, whose patience and support was instrumental in the completion of this work.
I. INTRODUCTION

A. PURPOSE

This thesis describes the development of a microcomputer oriented program called MSCOP (Microcomputer Software for Constrained Optimization Problems) for constrained optimization of engineering design problems. Problems which can be solved by the MSCOP are nonlinear programming problems arising in several areas of machine and structural design, such as the minimum weight design of structures subject to stress and displacement constraints [Ref. 1].

In recent years, several powerful general purpose optimization programs have become available for engineering design problems, e.g., COPES/CONMIN [Ref. 2], and ADS-1 [Ref. 3]. These programs can handle a wide range of design problems and contain a variety of solution techniques. Also, several programs are available that include optimization in an integrated analysis/design code, e.g., ACCESS, ASOP, EAL, PAPS, SAVES, SPAR, STARS and TSO [Ref. 4]. All of the above optimization programs are written in FORTRAN, and are built for use on a mainframe computer. Their use can be cumbersome, especially for the occasional user. Since many engineers are now using microcomputers, there is a need to develop an optimization program contained in a microcomputer software package for use on microcomputers. This thesis fills that need by developing a compact program written in a standard BASIC language suitable for a wide range of microcomputers.
B. IMPLEMENTATION

The nature of an optimization program depends on the computer and programming method available. The MSCOP software is designed for use on a microcomputer. However, for the speed of development and testing, MSCOP was developed on the IBM 3033 computer at the P. F. Church Computer Center in Naval Postgraduate School, and was written in WSASIC (Waterloo Basic) Version 2.0.

To make sure that the program is easily portable to a microcomputer, only standard BASIC commands and functions are used. For example, FOR I = 1 TO "DE... NEXT I, GOSUB etc., were used. The commands and functions not available in all variations of BASIC are avoided, for example, TRN(A), MAT(A), etc.

MSCOP provides design engineers with a convenient tool for optimization of engineering design problems with up to 20 bounded design variables and as many as 50 inequality constraints.

C. GENERAL OPTIMIZATION MODEL

The general optimization problem to be solved is of the form: Find the set of design variables \( \mathbf{X} \) that will

\[
\text{Minimize} \quad F(\mathbf{X}) \quad (1.1)
\]

\[
\text{Subject to} \quad G_j(\mathbf{X}) \leq 0 \quad j = 1, \ldots, m \quad (1.2)
\]

\[
^1_1 \mathbf{X} \leq \mathbf{X} \leq ^u_1 \mathbf{X} \quad i = 1, \ldots, n \quad (1.3)
\]

where \( \mathbf{X} \) is referred to as the vector of design variables. \( F(\mathbf{X}) \) is the objective function which is to be minimized. \( G(\mathbf{X}) \) are inequality constraint functions, and \( \mathbf{X}^1 \) and \( \mathbf{X}^u \) are lower and upper bounds, respectively, on the design
variables. Although these bounds or "side constraints" could be included in the inequality constraint set given by Eq(1.2), it is convenient to treat them separately because of their special structure. The objective function and constraint functions may be nonlinear, explicit or implicit in \( X \). However, they must be continuous and should have continuous first derivatives.

In general engineering optimization problems, the objective to be minimized is usually the weight or volume of a structure being designed while the constraints give limits on compressive stress, tensile stress, Euler buckling, displacement, frequencies (eigenvalues), etc. [Ref. 5: p.254]. Equality constraints are not included because their inclusion complicates the solution techniques and because in engineering situations, equality constraints are rare.

Most optimization algorithms require that an initial value of design variables \( X^0 \) be specified. Beginning from these starting values, the design is iteratively improved. The iterative procedure is given by

\[
X^{q+1} = X^q + a^* S^q
\]

(1.4)

where \( q \) is the iteration number, \( S \) is a search direction vector in the design space, and \( a^* \) is a scalar parameter which defines the amount of change in \( X \). At iteration \( q \), it is desirable to determine a direction \( S \) which will reduce the objective function (usable direction) without violating the constraints (feasible direction). After determining the search direction, the design variables, \( X \), are updated by Eq (1.4) so that the minimum objective value is found in this direction. [Ref. 6].

Thus, it is seen that nonlinear optimization algorithms for the general optimization problem based on Eq(1.4) can be separated into two parts, determination of search direction and determination of scalar parameter \( a^* \).
D. ORGANIZATION OF THIS THESIS

This chapter has stated the purpose of the thesis and has put the general concept of engineering optimization into a preliminary perspective. Chapter 2 will describe the essential aspects of the optimization algorithm used in MSCOP such as finding a search direction, the one-dimensional search and convergence criteria. Chapter 3 describes program usage. In chapter 4, there are three examples which are solved by the MSCOP. Summary and conclusions are given in chapter 5. The program is listed in the appendix.
II. OPTIMIZATION ALGORITHM

A. INTRODUCTION

There are many optimization algorithms for constrained nonlinear problems such as generalized reduced gradient method, feasible direction method, penalty function methods, Augmented Lagrangian multiplier method, and sequential linear programming. The feasible direction method is chosen for development in this thesis for three main reasons. First it progresses rapidly to a near optimum design. Second it only requires gradients of objective and constraint functions that are active at any given point in the optimization process [Ref. 7]. Third, because it maintains a feasible design, engineer cannot fail to meet safety requirements as defined by the constraints. However, the method does have several disadvantages in that it is prone to "zig-zag" between constraint boundaries and that it is usually does not achieve a precise optimum. This method solves the nonlinear programming problem by moving from a feasible point (can be initially infeasible) to another feasible point with an improved value of the objective value.

The following strategy is typical of feasible direction method: Assuming that an initial feasible point $X_0$ is known, first find a usable-feasible direction $S$. The algorithm for this is similar to linear programming and complementary pivoting algorithms. Having found the search direction, a move is made in this direction to update the $X$ vector according to Eq(1.4). The scalar $a^*$ is found by a one-dimensional search to reduce the objective function as much as possible subject to constraints. That is: $a^*$
Figure 2.1 Algorithm for the Feasible Direction Method.

\( F(\bar{x} + a \cdot \bar{s}) \) subject to \( G(\bar{x} + a \cdot \bar{s}) \leq 0 \). It is assumed that the initial design \( \bar{x}^0 \) is feasible, but if it is not, a search
direction is found which will direct the design to the feasible region. After updating the $X_0$ vector, the convergence test must be performed in the iterative algorithm. A convergence criteria used in this implementation are described in section C. The general algorithm used in M$\textit{SCOP}$ is given in Figure 2.1

B. SEARCH DIRECTION

In the feasible direction algorithm, a usable feasible search direction $S$ is found which will reduce the objective function without violating any constraints for some finite move. It is assumed that at any point in the design space (at any $X$) the value of the objective and constraint functions as well as the gradients of these functions with respect to the design variables can be calculated. Since these gradients cannot usually be calculated analytically, the finite difference method Eq(2.1) is used in M$\textit{SCOP}$.

$$\frac{\partial \Pi(X)}{\partial x_i} = \frac{F(X + \epsilon \epsilon_i) - F(X)}{\epsilon}$$  \hspace{1cm} (2.1)

where $\epsilon_i$ is the ith unit vector

$\epsilon$ is a small scalar.

In M$\textit{SCOP}$, $\epsilon$ is 0.1% of the ith design variable.

In the feasible direction algorithm, there are usually one or more "active" constraints. A constraint $G(X) \leq 0$ is "active" at $X$ if $g(X) \approx 0$. As shown in Figure 2.1, if no constraints are active the standard steepest descent direction $S = -\nabla F$ is used.
1. Usable-Feasible Direction

Assume there are $N$ active constraints at $\mathbf{x}$. The direction $\mathbf{s}$ is "usable" if it reduces the objective function, i.e.,

$$\nabla F \cdot \mathbf{s} < 0$$

(2.2)

Similarly the direction is feasible if for a small movement in this direction, no constraint will be violated, i.e.,

$$\nabla g \cdot \mathbf{s} < 0$$

(2.3)

This is shown geometrically in Figure 2.2
2. Active Constraints

It is necessary to determine if a constraint is active or violated in the feasible direction algorithm. A constraint \( G(x) \leq 0 \) is "active" at \( x^0 \) if \( G(x^0) \approx 0 \). In order to avoid the zigzagging effect between one or more constraint boundaries, a tolerance band about zero is used for determining whether or not a constraint is active. From the engineering point of view, a constraint \( G(x) \leq 0 \) is active near the boundary \( G(x) = 0 \) whenever \( ACC \leq G(x) \leq VCC \). ACC is the active constraint criterion and VCC is the violated constraint criterion in MSCOP. Assuming the feasible constraints are normalized so that \( G(x) \) ranges between -1 and 0 for reasonable values of \( x \), the constraint \( G(x) \leq 0 \) is considered active if \( G(x) > -0.1 \). The constraint is considered to be violated if \( G(x) > 0.004 \). This is an algorithmic trick which improves efficiency and reliability of the algorithm. However, since in the one-dimensional search, all interpolations for constraint \( G(x) \) are done for zeros of a linear or quadratic approximation to \( G(x) \) in order to find \( a^* \), at the optimum the value of active constraints are very near zero, but may be as large as 0.004 [Ref. 6]. From an engineering point of view, a 0.4 "constraint violation is considered to be acceptable.

3. Suboptimization Problem and Push-Off Factors

Zoutendijk [Ref. 8] has shown that a usable feasible direction \( S \) may be found as follows:

Maximize \( \beta \) \hspace{1cm} \text{(2.4)}

Subject to :

\[ \nabla F(x) \cdot S + \beta \leq 0 \] \hspace{1cm} \text{(2.5)}
\[ \forall G(x) \cdot S + \theta_j \beta \leq 0 \quad j \in J \]  
(2.6)

\[ S \text{ bounded} \]  
(2.7)

Where scalar \( \beta \) is a measure of the satisfaction of the usability and feasibility requirements. The scalar \( \theta_j \) in Eq (2.6) is referred to as the "push-off" factor which effectively pushes the search direction away from the active constraints. In Eq (2.6), if the push-off factor is zero, the search direction is tangent to the active constraints, and if it is infinite, then the search direction is tangent to the objective function. It has been found that a

**Figure 2.3** Push-Off Factor and Bounding of the S-Vector.
push-off factor is defined as follows gives good results [Ref. 5: p.167]:

\[ \theta_j = \left[ 1 - \frac{G_j(X)}{\text{ACC}} \right]^2 \theta_0 \]  \hspace{1cm} (2.8)

where \( \theta_0 = 1 \).

To avoid an unbounded solution when seeking a usable feasible direction it is necessary to impose bounds on the search direction \( S \). One method of imposing bounds on search direction is to impose bounds on the components of \( S \)-vector of form:

\[-1 < s_i < 1 \] \hspace{1cm} (2.9)

This choice of bounding the \( S \)-vector actually biases the search direction. This is undesirable since we wish to use the push-off factors as our means of controlling the search direction. A method which avoids this bias in search direction is the circle as shown Figure 2.3. The norm here is

\[ S \cdot S \leq 1 \] \hspace{1cm} (2.9.1)

4. **Simple Simplex-like Method for Search Direction**

Vanderplaats [Ref. 5: pp.168-169] provides the matrix formulation which solves the above sub-optimization problem by using the Zoutendijk method.

Maximize \( P \cdot y \) \hspace{1cm} (2.10)

Subject to:

\[ A \cdot y \leq 0 \] \hspace{1cm} (2.11)
\[ Y \cdot Y \leq 1 \quad (2.12) \]

Where

\[
Y = \begin{bmatrix}
S_1 \\
S_2 \\
\vdots \\
S_m \\
0
\end{bmatrix}
\quad \quad
P = \begin{bmatrix}
0 \\
0 \\
\vdots \\
0 \\
1
\end{bmatrix}
\quad (2.13)
\]

\[
M = \begin{bmatrix}
\nabla G_1(X), \theta_1 \\
\nabla G_2(X), \theta_2 \\
\vdots \\
\nabla G_j(X), \theta_j \\
\nabla F(X), 1
\end{bmatrix}
\quad (2.14)
\]

and where \( j \) is the number of active constraints (NAC).

When the solution to Eq(2.10) through (2.12) is found, \( S \) may be normalized to some value other than unity, but the form of the normalization is the same. A solution to the above problem may be obtained by solving the following system derived from the Kuhn-Tucker conditions for that problem:

\[
\begin{bmatrix}
B & I
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix} = C
\quad (2.15)
\]

\[ u_i > 0 \quad v_i > 0 \quad u \cdot v = 0 \quad (2.16) \]

Where
\[ \mathbf{P} = -\mathbf{A} \cdot \mathbf{A} \quad (2.17) \]
\[ \mathbf{I} = \text{Identity matrix} \quad (2.18) \]
\[ \mathbf{C} = -\mathbf{A} \cdot \mathbf{F} \quad (2.19) \]

Above system can be solved using a complimentary pivot algorithm. Choose an initial basic solution to Eq(2.15) is to be

\[ \mathbf{y} = \mathbf{C}, \quad \mathbf{u} = \mathbf{0} \quad (2.20) \]

where \( \mathbf{y} \) is the set of basic variables and \( \mathbf{u} \) is the set of nonbasic variables. If all \( v_i > 0 \), Eq(2.16) is also satisfied and problem is solved. If some \( v_i < 0 \), the solution procedure is as follows:

Let \( E_{ii} \) be the diagonal element of the \( i \)-th nonbasic variable.

1. Given the condition that some \( c \) is less than zero, we find \( \max (c_i/E_{ii}) \) which is the incoming row to the basis.
2. The incoming column is changed to a basic column, the tableau is updated by a standard simplex pivot on \( E_{ii} \).
3. Until all \( c_i > 0 \), repeat steps 1. and 2.
4. When all \( c_i > 0 \), the iteration is complete. The value of \( u \) is now the desired solution.
5. By using \( \mathbf{y} = \mathbf{p} - \mathbf{A} \cdot \mathbf{u} \), we get the usable-feasible search direction \( S \) which is first NDV components of \( \mathbf{y} \).

6. Initially Infeasible Designs

The method of feasible directions assumes that we begin with a feasible design and feasibility is maintained throughout the optimization process. If the initial design
is infeasible, then a search direction pointing toward the feasible region can be found by a simple modification to direction finding problem.

A design situation can exist in which the violated constraints are strongly dependent on part of the design variables, while the objective function is primarily dependent on the other design variables. This suggests a method for finding a search direction which will simultaneously minimize the objective while overcoming the constraint violations. These considerations lead to the following statement of the direction finding problem [Ref. 5, pp. 171-172]:

Maximize $-\nabla F(X) \cdot S + \Xi \theta$ \hspace{1cm} (2.21)

Subject to:

$\nabla G(Y) \cdot S + \theta \theta < 0 \hspace{1cm} j \in J$ \hspace{1cm} (2.22)

$S \cdot S < 1$ \hspace{1cm} (2.23)

where $J$ is the set of active and violated constraints, and where the scalar $\Xi$ in Eq(2.21) is a weighting factor determining the relative importance of the objective and the constraints. Usually a value of $\Xi > 10000$ will ensure that the resulting $S$-vector will point toward the feasible region. Incorporating Eq(2.21) and Eq(2.22) into the direction finding algorithm requires only that we modify the $p$-vector given in Eq (2.24) and the $A$-matrix of Eq (2.25).

$$p = \left[ \begin{array}{c} -\nabla F(Y) \\ \Xi \end{array} \right]$$ \hspace{1cm} (2.24)
We use the simple simplex-like method to find the search direction toward the feasible region.

C. ONE-DIMENSIONAL SEARCH

1. No Violated Constraints

If no constraints are violated, we find the largest a* in Eq. (1.4) from all possible values that will minimize the objective on S without violating any constraints, active or inactive.

The procedure in MSCOF is as follows:

1. Let a0, a1, a2, a3 be the scalar in Eq. (1.4) corresponding to points X0, X1, X2, X3, X4.
2. a0 = 0 at given point X0.
3. In order to get a1, we can calculate the a1 to reduce the objective by at most 10% or to change each of the design variable A by at most 10%.
4. Update the design variables to X1 using Eq. (1.4).
5. Evaluate the objective for X1, and check the feasibility. If one or more constraints is violated, then a1 is reduced to a1/2, and we go to step 4.
6. In order to estimate a2, we can use the quadratic approximation with 2 points X, X1 and the ∇F.
7. Update the design variables to \( X_2 \) by Eq(1.4) and check the side constraints.

8. Evaluate the objective and constraints.

9. Now having 3 \( a \)'s, and values of objectives and constraints for design variables \( Y_0, Y_1, X_2 \) are known, so by using 3-point quadratic approximation, a value of \( a_3 \) is found.

10. Update the new optimal point in search direction by Eq(1.4).

11. Evaluate the objective and constraints.

12. Now choose last 3 values, \( a_1, a_2, a_3 \) and find a new \( a_3 \) using 3-points Quadratic approximation.

13. Choose the \( a^* \) among the 5 points which corresponds to the minimum objective function value with no-violated constraints.

2. **One or More Constraints Violated**

   If one or more constraints are initially violated, a modified usable-feasible direction is found. It is then necessary to find the scalar \( a^* \) in Eq(1.4) which will minimize the maximum constraint violation, using the most violated constraint \( j \), a good initial estimate for \( a^* \) is

   \[
   a^* = \frac{-G_j(X)}{\nabla G_j(X) \cdot S} \tag{2.27}
   \]

   Since the gradients of the violated constraints are known, the scalar which is required to obtain a feasible design with respect to violated constraint in the search direction, is given to a first approximation by Eq(2.27).

   The more detail procedure in MSCP is as follow:

   1. Choose the most violated constraint \( j \).

   2. Calculate \( a^* \) for violated constraint \( j \) using Eq(2.27).
3. Update the design variables for a* and check the side constraints.

4. If one or more violated constraints still exist, then calculate the derivative of objective, violated and active constraints and find a new search direction and then go to step 1. Otherwise proceed with the optimization in the normal fashion.

D. CONVERGENCE CRITERIA

A desired property of an algorithm for solving a nonlinear problem is that it should generate a sequence of points converging to a global optimal point. In many cases, however, we may have to be satisfied with less favorable outcomes. In fact, as a result of non-convexity, problem size, and other difficulties, we may stop the iterative procedure if a point belongs to a described set, which is defined in MSCOF as follows:

1. \( Q_1 = \{ \mathbf{x} \mid |\mathbf{x}^0 - \mathbf{x}| < \varepsilon_x |\mathbf{x}^0| \} \)

2. \( Q_2 = \{ \mathbf{x} \mid |F(\mathbf{x}^0) - F(\mathbf{x})| < \varepsilon_F |F(\mathbf{x}^0)| \} \)

In MSCOF, the algorithm is terminated if a point \( \mathbf{x} \) is reached such that \( \mathbf{x} \in Q_1 \cap Q_2 \). \( \varepsilon_x = 0.001 \) and \( \varepsilon_F \) is approximately 0.001. Since in engineering design problems it is not necessary to find solutions with more than three significant digits.
III. MSCOP USAGE

A. INTRODUCTION

Since this MSCOP is written in WATERLOG BASIC Version 2.0, it is very convenient to use. The user must first formulate the design problem with the classical machine design criteria. Given the formulation of the design problem as a nonlinear program, the user then enters the problem as a part of a BASIC program. The user defines the objective function and constraint functions using BASIC statements. Other parameters are input as data: the number of design variables NDV, the number of inequality constraints NIQC, variable bounds an initial design value and a print control number.

B. PROBLEM FORMULATION

Generally, the experienced design engineer will be able to choose the appropriate objective for optimization depending on the requirements of the particular application. The physical phenomena of significance should first be summarized for the device to be designed. The appropriate objective can then be selected and constraints can be imposed on the remaining phenomena to assure an acceptable design from all standpoints. However, the initial formulation for the optimization problem should not be more complicated than necessary and this often requires the making of some simplifying assumptions. [Ref. 9].

After completing the formulation of the design problem, the design engineer should be able to answer the following questions:

1. What are the design variables?
2. What is the objective function?
3. What are the inequality constraints?
4. What are the bounds on the variables?

The engineer is then ready to input the program to the MSCOP. However, additional study and preparation of the problem may be useful. In particular, redundant constraints should be avoided if possible. MSCOP will operate with redundant constraints but it will operate faster without them. Selection of an initial design point from which to start this program is important, since it affects performance and running time. The user should use any available information which gives a good initial approximation. If side constraints exist, the user must be sure the initial values of the design variables do not violate the side constraints. This program will automatically handle an initial design point which is infeasible with respect to the \( G(X) < 0 \) constraints. However, if the initial point does not violate these constraints, convergence will likely be more rapid.

C. PROBLEM ENTRY

Problem entry is accomplished by editing the main program directly. As an example, consider the following simple NLP with two design variables, and three constraint functions.

Minimize \( F(X) = X_1^2 + 3 X_1 X_2 + 2 X_2^2 - X_1 - X_2 + 1 \)

subject to:

\[
X_1 + X_2 - 3 \leq 0
\]

\[
\frac{1}{X_1} + \frac{1}{X_2} - 2 \leq 0
\]
\[ x^2 + x - x_2 - 2 \leq 0 \]

\[ x_i > 0.1 \]

With the MSCOP loaded into memory and listed on the CRT, modifications are made on the program lines as follows to input this example:

Line 100

Just after the word "data", three integers are added, separated by a comma. The first number is NDV which is the number of design variables, the second is NIQC which is the number of inequality constraints, and the third is IPRT which is print control number ( 0 ; only final results, 1 ; given data and final results, 2 ; given data and iterative suboptimal results)

For example:

100 data 2,3,2

Lines 201-220

Each line here corresponds to a separate design variable, beginning with \( X(1) \) and continuing in order to input \( X(NDV) \). On each line, three values are separated by commas. After the word "data", these values are the initial values of the design variable, the lower bound on the variable and the upper bound on the variable. If no bound is to be specified, the entry is filled by "no".

For the sample problem, the input is:

201 data 3.,0.1,no
202 data 3.,0.1,no
Lines 400 - 450

These lines are available for defining the objective function. The objective function must be defined in terms of subscripted design variables $x(1), x(2),$ etc.

For the sample problem, the input is:

$$400 \text{ fn}_f = x(1)^2 + x(1) x(2) + 2 \cdot x(2)^2 - x(1) - x(2) + 1.$$ 

Lines 500-650

These lines are available for defining the inequality constraint functions, which must be expressed using the format:

$$601 \text{ if } i = k \text{ then } \text{fn}_g = G_i(x) - b_i$$

For the sample problem, the input is:

$$00601 \text{ if } i = 1 \text{ then } \text{fn}_g = x(1) + x(2) - 3.$$  
$$00602 \text{ if } i = 2 \text{ then } \text{fn}_g = 1/x(1) + 1/x(2) - 2.$$  
$$00603 \text{ if } i = 3 \text{ then } \text{fn}_g = x(1)^2 + x(1) - x(2) - 2.$$ 

If there are many constant values in the constraint functions, the user may input data for these functions on lines 501-600 in order to simplify their statements.
IV. EXAMPLE PROBLEMS

A. DESIGN OF CANTILEVERED BEAM

1. Uniform Cantilevered Beam

Assume a cantilevered beam as shown in Figure 4.1 must be designed. The objective is to find the minimum volume of material which will support the load P.

The design variables are the width B and height H in the beam. The design task is as follows: Find B and H to minimize volume $V = B \times H \times l$ (4.1)
we wish to design the beam subject to limit on bending stress, shear stress, deflection and geometric conditions. The bending stress in the beam must not exceed 20,000 psi.

\[
\sigma = \frac{Mc}{I} = \frac{6pl}{BH^2} < 20,000 \tag{4.2}
\]

The shear stress must not exceed 10,000 psi.

\[
\sigma_s = \frac{3p}{2A} = \frac{3p}{2BH} < 10,000 \tag{4.3}
\]

and the deflection under the load must not exceed 1 inch.

\[
\delta = \frac{pl^3}{3EI} = \frac{4fl^3}{EBH} < 1.0 \tag{4.4}
\]

Additionally, geometric limits are imposed on the beam size.

\[
0.5 < B < 5.0 \tag{4.5}
\]

\[
1.0 < H < 20.0 \tag{4.6}
\]

\[
H/b < 10. \tag{4.7}
\]

Now we can input this problem to MSCOP.

Input NDV, MIQC, IPRT

00100 data 2,4,2

Initial starting points

00210 data 3.5,0.5,5.0
00220 data 16.0,1.0,20.0

Evaluation of objective

00400 fr_f = t1*x(1)*x(2)
Evaluation of constraints

\[
\begin{align*}
0.0000 & \quad t_1 = 2.09 \\
0.0001 & \quad t_0 = 3.7 \times 10^6 \\
0.0002 & \quad \text{if } i = 1 \text{ then } f_{n-g} = 6.0 \times bp*tl/(20000.0*b*h**2) - 1. \\
0.0003 & \quad \text{if } i = 2 \text{ then } f_{n-g} = 3.0 \times bp/(10000.0*2.0*b*h) - 1. \\
0.0004 & \quad \text{if } i = 3 \text{ then } f_{n-g} = 4.0 \times bp*tl**3/(be*b*h**3) - 1. \\
0.0005 & \quad \text{if } i = 4 \text{ then } f_{n-g} = h/t - 10.
\end{align*}
\]

<table>
<thead>
<tr>
<th>TABLE I</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Solution of a Uniform Cantilevered Beam</td>
</tr>
</tbody>
</table>

objective : 6664.0

design variable :
\[
\begin{align*}
X(1) & = 1.852 \\
X(2) & = 17.99
\end{align*}
\]

constraint :
\[
\begin{align*}
g(1) & = 0.000902 \\
g(2) & = -0.9549 \\
g(3) & = -0.0109 \\
g(4) & = -0.0286
\end{align*}
\]

As a result of this problem are in Table 4.1.

2. Variable Cantilevered Beam

The cantilevered beam shown in Figure 4.2 is to be designed for minimum material volume. The design variables are the width \( b \) and height \( h \) at each of 5 segments. We wish to design the beam subject to limits on stress (calculated at left end of each segment), deflection under the load, and the geometric requirement that the height of any segment does not exceed 20 times the width.
The deflection $y$ at the right end of segment $i$ is calculated by the following recursion formulas:

$$y_o = y'_o = 0$$  \hspace{1cm} (4.8)

$$y' = \frac{P}{E I} \left[ L + \frac{1}{2} \sum_{j=1}^{i-1} \frac{1}{l_i} \right] + y'_{i-1}$$  \hspace{1cm} (4.9)

$$y = \frac{P}{2 E I} \left[ L - \sum_{j=1}^{i-1} \frac{l_i}{l_j} + \frac{2 l_i}{3} \right] + y'_{i-1} + y'_{i-1}$$  \hspace{1cm} (4.10)
where the deflection \( y \) is defined as positive downward, \( y' \) is the derivative of \( y \) with respect to the \( x \), and \( l_i \) is the length of segment \( i \). Young's modulus \( E \) is the same for all segments, and the moment of inertia for segment \( i \) is

\[
I_i = \frac{b_i h_i^3}{12}
\]

(4.11)

The bending moment at the left end of segment \( i \) is calculated as

\[
M_i = P \left[ L + 1 - \sum_{j=1}^{i} l_j \right]
\]

(4.12)

and the corresponding maximum bending stress is

\[
\sigma_i = \frac{M_i h_i}{2 I_i}
\]

(4.13)

The design task is now defined as

\[
\text{Minimize : } V = \sum_{i=1}^{N} b_i h_i l_i
\]

Subject to :

\[
\frac{\sigma_i}{\sigma} - 1 \leq 0 \quad i = 1, \ldots, N
\]

(4.16)

\[
\frac{y_N}{y} - 1 \leq 0
\]

(4.17)

\[
h_i - 20 b_i \leq 0 \quad i = 1, \ldots, N
\]

(4.18)
\( h > 1.0 \quad h > 5.0 \quad i = 1, \ldots, N \) \hspace{1cm} (4.19)

where \( \sigma \) is the allowable bending stress and \( \gamma \) is the allowable displacement. This is a design problem in 10 variables. There are 6 nonlinear constraints defined by Eq(4.16) and Eq(4.17), and 5 linear constraints defined by Eq(4.18), and 10 side constraints on the design variables defined by Eq(4.19).

Now we can input this problem to MSCOP.

Input NDV, NIQC, IPRT

09100 data 10,11,2

Initial starting points

00210 data 5.,1.,no
00220 data 5.,1.,no
00230 data 5.,1.,no
00240 data 5.,1.,no
00250 data 5.,1.,no
00260 data 49.,5.,no
00270 data 40.,4.,no
00280 data 40.,4.,no
00290 data 40.,4.,no

Evaluation of objective

00400 fn \( f = 100. * \left( x(1) * x(6) + x(2) * x(7) + x(3) * x(8) x(4) * x(9) + x(5) * x(10) \right) \)

Evaluation of constraints.

00490 def fn g(x,i)
00498 dim gm(10),bi(10),sigi(10),ypb(10),yb(10)
00500 pch = 50000.
00501 t0 = 200.e+5
00502 tl = 200.
00503 sigb = 14000.
00504 yb = 40.
00505 sl = 40.
00506 for \( m = 1 \) to 5
00507 \( b_{m}(m) = pch *(t1+sl-m*sl) \)
00508 next \( m \)
00509 for \( m = 1 \) to 5
00510 \( km = m+5 \)
00511 bi(\( m \)) = \( x(m) * x(km) ** 3/12. \)
00512 \( sigi(\( m \)) = b_{m}(m) * x(km) / (2.*bi(\( m \))) \)
00513 next \( m \)
00514 yzo = 0.
00515 fzo = 0.
00516 for \( m = 1 \) to 5
TABLE II
The Solution of a Variable Cantilevered Beam

objective ; 62133.35

design variables

<table>
<thead>
<tr>
<th>X(1)</th>
<th>2.994</th>
</tr>
</thead>
<tbody>
<tr>
<td>X(2)</td>
<td>2.782</td>
</tr>
<tr>
<td>X(3)</td>
<td>2.528</td>
</tr>
<tr>
<td>X(4)</td>
<td>2.208</td>
</tr>
<tr>
<td>X(5)</td>
<td>1.761</td>
</tr>
<tr>
<td>X(6)</td>
<td>59.88</td>
</tr>
<tr>
<td>X(7)</td>
<td>55.62</td>
</tr>
<tr>
<td>X(8)</td>
<td>50.56</td>
</tr>
<tr>
<td>X(9)</td>
<td>44.14</td>
</tr>
<tr>
<td>X(10)</td>
<td>35.19</td>
</tr>
</tbody>
</table>

constraints

<table>
<thead>
<tr>
<th>G(1)</th>
<th>-0.00219</th>
</tr>
</thead>
<tbody>
<tr>
<td>G(2)</td>
<td>-0.00415</td>
</tr>
<tr>
<td>G(3)</td>
<td>-0.00508</td>
</tr>
<tr>
<td>G(4)</td>
<td>-0.00406</td>
</tr>
<tr>
<td>G(5)</td>
<td>-0.0177</td>
</tr>
<tr>
<td>G(6)</td>
<td>-0.4401</td>
</tr>
<tr>
<td>G(7)</td>
<td>-0.0101</td>
</tr>
<tr>
<td>G(8)</td>
<td>-0.0231</td>
</tr>
<tr>
<td>G(9)</td>
<td>0.0000</td>
</tr>
<tr>
<td>G(10)</td>
<td>-0.0248</td>
</tr>
<tr>
<td>G(11)</td>
<td>-0.0278</td>
</tr>
</tbody>
</table>
A simple truss with 5 members as shown in Figure 4.3 is designed for the minimum volume. The design variables are the sectional areas of the members. The constraints are formed for the stresses of the members not to exceed the given allowable stress. The lower bound for each design variable is also considered. The stresses are obtained by the displacement method of the finite element analysis. The equation to be solved is given by

$$\mathbf{K} \cdot \mathbf{u} = \mathbf{P}$$  \hspace{1cm} (4.20)

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{u}$ is the displacement vector and $\mathbf{P}$ is the load vector as follows:
\[ Y = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ -5000 \end{bmatrix} \quad (4.21) \]

\[ \mathbf{K} = E \begin{bmatrix} \frac{A_1}{L} + \frac{A_5}{\sqrt{2}L} & \frac{A_5}{\sqrt{2}L} & 0 & 0 \\ \frac{A_5}{\sqrt{2}L} & \frac{A_5}{L} + \frac{A_5}{\sqrt{2}L} & 0 & -\frac{A_5}{L} \\ 0 & 0 & \frac{A_5}{L} + \frac{A_5}{\sqrt{2}L} & -\frac{A_5}{L} \\ 0 & -\frac{A_5}{L} & -\frac{A_5}{L} & \frac{A_5}{L} + \frac{A_5}{\sqrt{2}L} \end{bmatrix} \quad (4.22) \]

From Eq. (4.20) the displacements are solved by

\[ \mathbf{u} = \mathbf{K}^{-1} \cdot \mathbf{P} \quad (4.23) \]

Having displacements at all nodes, we can calculate the stress for each element.

\[ \sigma_i = E \cdot \varepsilon = \frac{E \cdot \Delta l_i}{l_i} \quad (4.24) \]

where

\[ \Delta l_1 = \sqrt{(1 + u_1)^2 + v_1^2} - 1 \]

\[ \Delta l_2 = \sqrt{(1 + v_2)^2 + (u_2 - u_1)^2} - 1 \]

\[ \Delta l_3 = \sqrt{(1 + u_2)^2 + v_2^2} - 1 \quad (4.25) \]
\[
\Delta l_4 = \sqrt{(1 + u_1)^2 + (1 - v_2)^2} - l_4 \\
\Delta l_5 = \sqrt{(1 + u_1)^2 + (1 + v_1)^2} - l_5
\]

The design problem is given by

\[
\text{minimize } V = \sum_{i=1}^{5} A_i l_i \tag{4.26}
\]

Subject to

\[
G_i = \frac{1}{\sigma_i} - 1.0 < 0 \quad i = 1, \ldots, 5 \tag{4.27}
\]

\[
A_i > 0.1 \quad i = 1, \ldots, 5 \tag{4.28}
\]

The MSCOF input for this problem is given as follows:

Input NDV, NIQC, IPRT

00100 data 5, 5, 2

Initial starting point

00200 data 3., 1, no
00202 data 3., 1, no
00204 data 3., 1, no
00206 data 3., 1, no
00208 data 3., 1, no

Evaluation of objective

00400 fn_f = \text{100} \times (x(1) + x(2) + x(3) + \text{sgn}(2.) \times x(4) + \text{sgn}(2.) \times x(5))

Evaluation of constraints

0500 \text{dim } x(5)
0501 \text{te } = 2.0 \times 7
0502 t_1 = 100.
0503 \text{sigb } = 14000.
\[
\begin{align*}
0504 & \quad cs = 2. * \text{syr}(2.) \\
0505 & \quad ct = \text{te} / \text{tl} \\
0506 & \quad k11 = (x(1) + x(5) / cs) * ct \\
0507 & \quad k12 = x(5) * ct / cs \\
0508 & \quad k21 = k12 \\
0509 & \quad k22 = (x(2) + x(5) / cs) * ct \\
0510 & \quad k24 = -x(3) * ct \\
0511 & \quad k32 = (x(3) + x(4) / cs) * ct \\
0512 & \quad k34 = -x(4) * ct / cs \\
0513 & \quad k42 = k24 \\
0514 & \quad k44 = (x(2) + x(4) / cs) * ct \\
0515 & \quad k12 = \frac{-k12 + k22 * dk1}{k24} \\
0516 & \quad dk3 = -x(3) * dk2 / k34 \\
0517 & \quad pp = -6.0000. \\
0518 & \quad vv(1) = \frac{pp}{(x1.2 * dk1 + k43 * dk3 + k44 * dk2)} \\
0519 & \quad vv(2) = dk1 * vv(1) \\
0520 & \quad vv(3) = dk3 * vv(1) \\
0521 & \quad vv(4) = dk2 * vv(1) \\
0522 & \quad d11 = \text{syr} \left\{ (\text{tl} + vv(1)) * 2 + vv(2) * 2 - tl \right\} \\
0523 & \quad d12 = \text{syr} \left\{ (\text{tl} + vv(2) - vv(4)) * 2 + (vv(1) - vv(3)) * 2 - tl \right\} \\
0524 & \quad d13 = \text{syr} \left\{ (\text{tl} + vv(3)) * 2 + vv(4) * 2 - tl \right\} \\
0525 & \quad d14 = \text{syr} \left\{ (\text{tl} + vv(3)) * 2 + (\text{tl} - vv(4)) * 2 - tl \right\} \\
0526 & \quad hl = \text{syr} \left\{ (\text{ht} + vv(3)) * 2 + (\text{ht} - vv(4)) * 2 - tl \right\} \\
0527 & \quad d15 = \text{syr} \left\{ (\text{ht} + vv(3)) * 2 + (\text{ht} + vv(2)) * 2 - tl \right\} \\
0528 & \quad hl = \text{syr} \left\{ (\text{ht} + vv(3)) * 2 + (\text{ht} - vv(4)) * 2 - tl \right\} \\
0529 & \quad dl1 = \text{te} * \text{dl1} / (\text{tl} * \text{sigb}) - 1. \\
0530 & \quad dl2 = \text{te} * \text{dl2} / (\text{tl} * \text{sigb}) - 1. \\
0531 & \quad dl3 = \text{te} * \text{dl3} / (\text{tl} * \text{sigb}) - 1. \\
0532 & \quad dl4 = \text{te} * \text{dl4} / (\text{hl} * \text{sigb}) - 1. \\
0533 & \quad dl5 = \text{te} * \text{dl5} / (\text{hl} * \text{sigb}) - 1. \\
0534 & \quad fnend \\
0535 & \quad \text{TABLE III} \\
0536 & \quad \text{The Solution of a 5-Bar Truss} \\
0537 & \quad \text{objective : 108.52} \\
0538 & \quad \text{design variables} \quad \text{constraint} \\
0539 & \quad x(1) = 0.1 \quad g(1) = -1.9988 \\
0540 & \quad x(2) = 0.1 \quad g(2) = -2.0030 \\
0541 & \quad x(3) = 3.514 \quad g(3) = -0.0030 \\
0542 & \quad x(4) = 4.948 \quad g(4) = -0.1203 \\
0543 & \quad x(5) = 0.1 \quad g(5) = -1.8797 \\
0544 & \quad \text{TABLE III} \\
0545 & \quad \text{The Solution of a 5-Bar Truss} \\
0546 & \quad \text{objective : 108.52} \\
0547 & \quad \text{design variables} \quad \text{constraint} \\
0548 & \quad x(1) = 0.1 \quad g(1) = -1.9988 \\
0549 & \quad x(2) = 0.1 \quad g(2) = -2.0030 \\
0550 & \quad x(3) = 3.514 \quad g(3) = -0.0030 \\
0551 & \quad x(4) = 4.948 \quad g(4) = -0.1203 \\
0552 & \quad x(5) = 0.1 \quad g(5) = -1.8797
V. SUMMARY AND CONCLUSION

Numerical optimization is a powerful technique for those confronted with practical engineering design problems. It is also a useful tool for obtaining reasonable solutions to the classical engineering design problems. Since many engineers are now using microcomputers for solving design problems, the development of microcomputer software which can be easily used is needed.

In this thesis, an algorithm for constrained optimization problems is programmed in standard BASIC language (WBASIC version 2.0) on an IBM 3033. The users can easily convert this to other microcomputers. MSCOP (Microcomputer Software for Constrained Optimization Problems) employs the method of feasible directions and specific modifications of a one-dimensional search for constrained optimization. MSCOP has been validated by tests on three constrained optimization problems. Its performance is good and could be made better through refinement of the algorithm.

Since microcomputers are available with reasonable memory size and computational speed, their capabilities will continue to improve as more engineering software becomes available. MSCOP is considered to be a first step toward more widespread use of optimization techniques on microcomputers.
APPENDIX A

MSCOP PROGRAM LISTING

0010 cption base 1
0020 dim x(21),x0(21),gcv(51),ngcv(51),dg(51,21)
0030 dim theta(21),wryk(51,51)
0040 dim a(51,21),b(51,51),p(21),v(21),s(21),u(51),c(51)
0050 dim iwrk(51),jwrk(51),wrk1(51),wrk2(51),wrk3(51)
0060 rem input data
0070 gosub '10000
0080 rem input number of design variables and constraints.
0090 read ndv,niqc,iirt
1100 data 2,4,2
1110 for i = 1 to ndv
1120 rem input initial value of design variables
1130 read x(i)
1140 if niqc = 0 then 160
1150 if lo$ = 'no' then lowb(i) = bnl0 else lowb(i) = value(lo$)
1160 if up$ = 'no' then uprb(i) = bnum else uprb(i) = value(up$)
1170 next i
1180 data 3.5,0.5,10.
1190 data 16.,1.0,20.
1200 rem evaluate the objective-function
1210 obj = fn_f(x)
1220 itri = 1
1230 def fn_f(x)
1240 fn_f = 200.*x(1)*x(2)
1250 rem evaluate the constraints
1260 for i = 1 to niqc
1270 gcv(i) = fn_g(x,i)
1280 next i
1290 rem constraint functions
1300 def fn_g(x,i)
1310 g = 50.e+6
1320 be = 50.e+6
1330 bp = 100000
1340 if i = 1 then fn_g = (6.*bp*tl)/(20000.*x(1)*x(2)**2)**1.5
1350 if i = 2 then fn_g = (3.*bp)/(20000.*x(1)**x(2))-1.
1360 if i = 3 then fn_g = (3.*bp*tl*3)/(be*x(1)**x(2)**3)-1.
1370 if i = 4 then fn_g = x(2)/(10.*x(1))-1.
1380 fnend
1390 rem initial counting number input
1400 ical = 1
1410 if ical > 3 then stop
1420 rem call the optimization code.
1430 gosub 2000
1440 rem print results.
1450 rem re-counting number input.
1460 ical = ical+1
1470 if ical = 3 then 850
1480 rem 10% reduce the design variables.
1490 for i = 1 to ndv

43
010 \( x(i) = 0.9 \times x(i) \)
020 \( x_0(i) = x(i) \)
030 \( \text{next } i \)
040 \( \text{goto } 720 \)
050 \( \text{rem increase design variables.} \)
060 \( \text{for } i = 1 \text{ to } \text{ndv} \)
070 \( x(i) = 1.1 \times x(i) \)
080 \( x_0(i) = x(i) \)
090 \( \text{next } i \)
100 \( \text{goto } 720 \)
110 \( \text{rem calculate the obj. constraint fcn.} \)
120 \( \text{obj} = \text{fn}_f(x) \)
130 \( \text{for } i = 1 \text{ to } \text{nicc} \)
140 \( gcv(i) = \text{fn}_g(x,i) \)
150 \( \text{next } i \)
160 \( \text{itrq} = 1 \)
170 \( \text{itrq} = \text{itrq} + 1 \)
180 \( \text{rem calculate the number of active and violate constraints.} \)
190 \( \text{gosub } 3500 \)
200 \( \text{rem calculate the gradient of objective and active or violated constraints.} \)
210 \( \text{gosub } 3800 \)
220 \( \text{if } \text{nvc} = 0 \text{ then } 2190 \)
230 \( \text{gosub } 3900 \)
240 \( \text{rem calculate the push-off factors} \)
250 \( \text{gosub } 4000 \)
260 \( \text{rem making the matrix C} \)
270 \( \text{rem normalized the df(i)} \)
280 \( \text{gosub } 4100 \)
290 \( \text{rem normalized the DG(i)} \)
300 \( \text{gosub } 4200 \)
310 \( \text{if } \text{nvc} > 0 \text{ then gosub } 4400 \text{ else gosub } 4600 \)
320 \( \text{rem calculate the usable-feasible direction s(i)} \)
330 \( \text{gosub } 5000 \)
340 \( \text{gosub } 2230 \)
350 \( \text{rem normalize the df(i)} \)
360 \( \text{for } i = 1 \text{ to } \text{ndv} \)
370 \( s(i) = -(df(i)) \)
380 \( \text{next } i \)
390 \( \text{rem normalize the s(i)} \)
400 \( \text{gosub } 5700 \)
410 \( \text{rem one-dimensional search} \)
420 \( \text{if } \text{nvc} = 0 \text{ then gosub } 6300 \text{ else gosub } 9000 \)
430 \( \text{rem update x for alph} \)
440 \( \text{gosub } 7000 \)
450 \( \text{gosub } 7100 \)
460 \( \text{rem calculate new point value.} \)
470 \( \text{obj} = \text{fn}_f(x) \)
480 \( \text{rem convergence test} \)
490 \( \text{gosub } 6780 \)
500 \( \text{if } \text{walp} \leq \text{accx} \text{ and } \text{delf} \leq \text{dabf} \text{ then } 2470 \)
510 \( \text{itrp} = \text{itrp} + 1 \)
520 \( \text{if } \text{itrp} > \text{mxit} \text{ then print 'check the problem'} \)
530 \( \text{obj} = \text{obj} \)
540 \( \text{for } i = 1 \text{ to } \text{ndv} \)
550 \( x_0(i) = x(i) \)
560 \( \text{next } i \)
570 \( \text{for } i = 1 \text{ to } \text{nicc} \)
580 \( gcv(i) = \text{fn}_g(x,i) \)
590 \( \text{next } i \)
600 \( \text{if } \text{iprt} = 2 \text{ then } 2460 \)
610 \( \text{gosub } 9200 \)
620 \( \text{gosub } 2010 \)
630 \( \text{rem print final results} \)
640 \( \text{print '***** final results *****'} \)
650 \( \text{gosub } 9200 \)
660 \( \text{return} \)
670 \( \text{rem initialize the integer working array} \)
rem initialize the integer working array
for i = 1 to nigm
iwrk(i) = 0
next i
return
rem initialize the one-dimension working array
for i = 1 to nigm
wrk1(i) = 0.
next i
return
rem initialize the one-dimension working array
for i = 1 to nigm
wrk2(i) = 0.
next i
return
rem initialize the one-dimension working array
for i = 1 to nigm
wrk3(i) = gcvi
next i
return
rem initialize the one-dimension working array
for i = 1 to ndvm
df(i) = 0.
next i
return
rem initialize the derivative of objective DF(i)
for i = 1 to ndvm
for j = 1 to nigm
wrky(i,j) = 0.
next j
next i
return
rem initialize the a(i,j),p(i),y(i),c(i)
for i = 1 to ndvm
for j = 1 to nigm
for k = 1 to nigm
a(i,j,k) = 0.
next k
next j
next i
rem initialize the derivative of constraints DG(i,j)
for i = 1 to nigm
for j = 1 to ndvm
dg(i,j) = 0.
next j
next i
return
rem initialize the b(i,j)
for i = 1 to nigm
for j = 1 to nigm
b(i,j) = 0.
next j
next i
return
rem calculate the number of active and violate constraints.
gosub 3000
nac = 0
nvc = 0
for i = 1 to nigm
if \( gcv(i) \geq vcc \) then

\[ nac = nac + 1 \]

goto 3590

\[ nvc = nvc + 1 \]

next \( i \)

if \( nvc = 0 \) then

\[ \text{next } i \]

\[ navc = nac + nvc \]

\[ \text{next } i \]

rem calculate the gradient of \( f(x) \)

\[ \text{gosub } 3400 \]

\[ \text{for } i = 1 \text{ to } ndv \]

\[ dx(i) = fdm * abs(x(i)) \]

\[ \text{if } dx(i) = mfs \text{ then } dx(i) = mfs \]

\[ x(i) = x(i) + df(i) \]

\[ \text{doj} = fn_f(x) \]

\[ df(i) = (\text{doj} - \text{obj}) / dx(i) \]

\[ x(i) = x(0) \]

\[ \text{next } i \]

\[ \text{return} \]

rem calculate the \( DG(i,j) \)

\[ \text{gosub } 3600 \]

\[ \text{for } i = 1 \text{ to } ndv \]

\[ dx(i) = fdm * x(i) \]

\[ \text{if } dx(i) < mfs \text{ then } dx(i) = mfs \]

\[ x(i) = x(i) + dx(i) \]

\[ \text{for } j = 1 \text{ to } navc \]

\[ k = iwrk(j) \]

\[ dcon = fn_g(x,k) \]

\[ ig(j,i) = (dcon - wrk1(j)) / dx(i) \]

\[ x(i) = x(0) \]

\[ \text{next } i \]

\[ \text{return} \]

rem calculate the push-off factor

\[ \text{gosub } 3700 \]

\[ \text{for } i = 1 \text{ to } navc \]

\[ \text{theta}(i) = \text{theta}(1) \times (1 - wrk1(i) / acc) \]

\[ \text{if } \text{theta}(i) > \text{thm} \text{ then } \text{theta}(i) = \text{thm} \]

\[ \text{next } i \]

\[ \text{return} \]

rem normalize the \( DF(i) \)

\[ \text{gosub } 3200 \]

\[ \text{fsq} = 0. \]

\[ \text{for } i = 1 \text{ to } ndv \]

\[ \text{fsq} = \text{fsq} + df(i) \]

\[ \text{next } i \]

\[ \text{fsq} = \text{sqr} \]

\[ \text{if } \text{fsq} = 0. \text{ then } \text{fsq} = \text{zro} \]

\[ \text{for } i = 1 \text{ to } ndv \]

\[ \text{wrk3}(i) = (1 / \text{fsq}) \times \text{df}(i) \]

\[ \text{next } i \]

\[ \text{return} \]

rem normalize the \( DG(i) \)

\[ \text{gosub } 3250 \]

\[ \text{for } i = 1 \text{ to } navc \]

\[ \text{gsq} = 0. \]
for j = 1 to ndv
    gsq = gsq + dg(i,j) ** 2
next j

if gsq == 0, then gsq = zero
for j = 1 to ndv
    wrky(i,j) = (1/gsq) * dg(i,j)
next j

return

rem exist the violate constraints
for i = 1 to navc
    for j = 1 to ndv
        a(i,j) = wrky(i,j)
    next j
    a(i,ndv+1) = theta(i)
next i

for i = 1 to ndv, p(i) = -wrk3(i)
next i

for i = 1 to navc
    v^2 = 0
    for j = 1 to ndv+1
        xx = a(i,j) * p(j)
        v^2 = v^2 + xx
    next j
    c(i) = (-1) * v
next i

ndt = navc
return

rem only exist active constraints
for i = 1 to navc
    for j = 1 to ndv
        a(i,j) = wrky(i,j)
    next j
    a(i,ndv+1) = theta(i)
next i

for i = 1 to ndv, p(navc+1,j) = wrk3(j)
next j

a(navc+1,ndv+1) = 1.
p(ndv+1) = 1.
for i = 1 to navc+1
    for j = 1 to ndv+1
        cc = a(i,ndv+1) * p(ndv+1)
    c(i) = (-1) * cc
next i

ndt = navc+1
return

rem calculate the usable-feasible direction
for i = 1 to ndt
    for j = 1 to ndv+1
        wrky(j,i) = a(i,j)
    next j
next i

for i = 1 to ndt
    for j = 1 to ndb
        ff = 0.
        for k = 1 to ndv+1
            tf = a(i,k) * wrky(k,j)
            ff = ff + tf
        next k
        b(i,j) = (-1) * ff
    next j

47
F
5180 next i
5190 iter = 0
5200 nmax = 5*ndb
5210 rem begin iteration
5220 iter = iter+1
5230 cbmx = 0.
5240 ichk = 0
5250 for i = 1 to ndb
5260 ci = c(i)
5270 bli = b(i,i)
5280 if bli = 0. then 5340
5290 if ci > 0. then 5340
5300 cb = ci/bli
5310 if cb <= cbmx then 5340
5320 ichk = i
5330 cbmx = cb
5340 next i
5350 if cbmx < zro or iter > nmax then 5550
5360 if ichk = 0 then 5550
5370 jj = iwrk(ichk)
5380 if jj = 0 then iwrd(ichk) = ichk else iwrd(ichk) = 0.
5390 if b(ichk,ichk) = 0. then b(ichk,ichk) = zro 
5400 bb = 1./b(ichk,ichk)
5410 for i = 1 to ndb
5420 b(i,ichk) = bb*b(i,ichk)
5430 next i
5440 c(ichk) = cbmx
5450 for i = 1 to ndb
5460 if i = ichk then 5530
5470 bbi = b(i,ichk)
5480 b(i,ichk) = 0.
5490 for j = 1 to ndb
5500 if j = ichk then 5520
5510 b(i,j) = b(i,j)-b(i,ichk)*bb*b(ichk,j)
5520 next j
5530 c(i) = c(i) - bbi*cbmx
5540 next i
5550 goto 5220
5560 for i = 1 to ndb
5570 y(i) = 0.
5580 if y(i) > 0 then y(i) = c(j)
5590 next i
5600 for i = 1 to ndb
5610 y(i) = y(i) + iwrd(i)
5620 next i
5630 for i = 1 to ndb
5640 ff = 0.
5650 for j = 1 to ndb
5660 ff = ff+y(i)*u(j)
5670 next j
5680 for i = 1 to ndv
5690 s(i) = y(i)-ff
5700 next i
5710 return
5720 rem normalized the s(i)
5730 ssq = 0.
5740 for i = 1 to ndv
5750 ssq = ssq+s(i)**2
5760 next i
5770 ssq = sqrt(ssq)
5780 if fs1p = 0. then fs1p = zro
5790 for i = 1 to ndv
5800 s(i) = (1./ssq)*s(i)
5810 next i
5820 return
6000 rem one-dimensional search for initial feasible point.
6010 rem calculate for slope of f(x)
6020 fs1p = 0.
6030 for i = 1 to ndv
6170 return
617948
s fslp = fslp+df(i)*s(i)
next i
rem identify the initial point.
alw = 0.
flow = obj
for i = 1 to nicc
   wrkl(i) = gcq(i)
next i
rem find a1st ; the 1st mid-point.
if s(i) = 0. then s(i) = zro
a1st = abo)*flow/abs(fslp)
for i = 1 to ndv
   if s(i) = 0. then s(i) = zro
   walp = alpx*x(i)/abs(s(i))
   if walp > a1st then 6095
   a1st = walp
next i
rem update x for a1st.
alph = a1st
gosub 7000
gosub 7100
rem calculate the fist and wrkl(i)
fist = fn f(x)
for i = 1 to nicc
   wrkl(i) = fn_g(x,i)
next i
rem check the feasibility.
ncv1 = 0
for i = 1 to nicc
   if wrkl(i) < vcc then 6170
   ncv1 = ncv1+1
next i
if ncv1 = 0 then 6200
a1st = 0.5*a1st
goto 6105
rem find a2nd ; the 2nd mid-point.
rem 2-points quadratic fit interpolation for minimum f(alpa).
a0 = flow
a1 = fslp
a2 = (fist-a1*a1st-a0)/(a1st**2)
if a2 <= 0. then a2 = zro
a2nd = -a1/(2.*a2)
rem 2-points linear interpolation for g(alpa)=0.
for i = 1 to nicc
   ai = wrkl(i)
   if a1st = 0. then a1st = zro
   a1 = (wrkl(i)-a0)/a1st
   if a1 <= 0. then a1 = zro
   walp = -a0/a1
   if walp <= 0. then walp = 1000.
   if walp >= a2nd then 6205
   a2nd = walp
next i
rem update x for a2nd.
alph = a2nd
gosub 7000
gosub 7100
rem calculate f2nd and wrk2(i)
f2nd = fn f(x)
for i = 1 to nicc
   wrk2(i) = fn_g(x,i)
next i
rem find final point aupr by using 3-points quadratic fit.
f1 = flow
alp1 = alow
f2 = fist
alp2 = a1st
6330 f3 = f2nd
6331 alp3 = a2nd
6335 rem 3-points quadratic fit interpolation.
6340 gosub 6600
6342 if a2 = 0. then a2 = zro
6345 if a3rd <= 0. then a3rd = 1000.
6350 for i = 1 to niqc
6355 f1 = wrkl(i)
6356 f2 = wrkl(i)
6357 f3 = wrk2(i)
6360 gosub 6600
6365 if alps > a3rd then 6380
6370 rem update x for aupr
6375 alph = a3rd
6380 gosub 7000
6385 gosub 7100
6390 rem calculate the fupr and wrku(i)
6395 fupr = fn f(x)
6400 for i = 1 to niqc
6405 wrku(i) = fn_g(x,i)
6410 next i
6415 rem find 4th new point.
6420 f1 = f1st
6425 f2 = f2nd
6430 f3 = f3rd
6435 alp1 = alst
6440 alp2 = a2nd
6445 alp3 = a3rd
6450 rem 3-points quadratic fit.
6455 gosub 6600
6460 if a2 = 0. then a2 = zro
6465 for i = 1 to niqc
6470 f1 = wrkl(i)
6475 f2 = wrk2(i)
6480 f3 = wrk3(i)
6485 alp1 = alst
6490 alp2 = a2nd
6495 alp3 = a3rd
6500 gosub 6600
6505 gosub 6630
6510 if alps > aupr then 6540
6515 aupr = alps
6520 next i
6525 rem update x for aupr
6530 alph = aupr
6535 gosub 7000
6540 gosub 7100
6545 rem evaluate fupr and wrku(i)
6550 fupr = fn f(x)
6555 for i = 1 to niqc
6560 wrku(i) = fn_g(x,i)
6565 next i
6570 rem find optimum alpa for not violating constraints.
6575 gosub 14300
6580 return
6585 rem find optimum alpa for not violating constraints.
6590 if alp1 = alp2 cr alp2 = alp3 or alp1 = alp3
6595 then return
6600 a2 = ((f3-f1)/(alp3-alp1) - (f2-f1)/(alp2-alp1))/(alp3-alp2)
6605 a1 = ((f3-f1)/(alp3-alp1) - f1)/a2*(alp1-alp2)
6610 a0 = f1-a1*alp1-a2*alp1^2
6615 return
6620 rem zero of polynomial for g(alpa)
dd = a1**2-4.*a2*a0
if dd < 0. then
a2 <= 0. then a2 = zro
alpc = {-a1+sqrt(dd)}/(2.*a2)
if alpb <= 0 and alpc <= 0. then
a2 = 0. then a2 = zro
alpb = (-al+sqrt(dd))/(2.*a2)
if alpb >= 0. and alpc >= 0. then
a2 = 0. then a2 = zro
alpc = (-al-sqrI dlfl)/(-.*a2)
if alpb <= 0 and alpc <= 0. then
a2 >= 0. and alpc >= 0. then
alpb >= 0. and alpc < 0. then
if alpb >= 0. and alpc < 0. then
alps = alpc
goto 6720
alps = alpb
goto 6720
alps = alpb
goto 6720
alps = alpc
goto 6720
alps = 1000.
return
rem update aboj and alpx
delf = abs(obj-nobj)
diff = abs{delf/obj}
abcj = (obj+diff)/2.
walp = 0.
welx = 0.
for i = 1 to ndv
delx = abs(x0(i)-x(i))
difx = abs{delx/x0(i)}
if delx >= welx then welx = delx
if difx <= walp then y880
walp = difx
next i
alpx = (alpx+walp)/2.
diff = accf*abs(obj)
return
rem update the x(i)
for i = 1 to ndv
x(i) = x0(i)+alpx*s(i)
next i
return
rem check the side-constraints.
for i = 1 to ndv
if x(i) < lcwb(i) then x(i) = lcwb(i)
if x(i) > uprb(i) then x(i) = uprb(i)
next i
return
rem estimate the alpa
fstr = flow
alpa = alow
nvc1 = 0
for i = 1 to nigc
if wrkl(i) < vcc then
nvc1 = nvc1+1
next i
if nvc1 > 0 then
alpa = a1st
fstr = f1st
nvc1 = 0
for i = 1 to nigc
if wrk2(i) < vcc then
nvc1 = nvc1+1
next i
if nvc1 > 0 then
if f2nd > fstr then
alpa = a2nd
fstr = f2nd
nvc1 = 0

51
for i = 1 to nicc
   if wrk3(i) < vcc then
      nvc1 = nvc1 + 1
   next i
if nvc1 > 0 then
   for i = 1 to nicc
      if wrk3(i) < vcc then
         nvc1 = nvc1 + 1
   next i
   if nvc1 > 0 then
      if f3rd > fstr then
         alpha = a3rd
      fstr = f3rd
      nvc1 = 0
   for i = 1 to nicc
      if wrk3(i) < vcc then
         nvc1 = nvc1 + 1
   next i
   if nvc1 > 0 then
      if f3rd > fstr then
         alpha = a3rd
      fstr = f3rd
      nvc1 = 0
   return
rem one-dimensional search for initial infeasible point.
ii = 1
gcvm = wrk1(1)
for i = 1 to nvc
   if wrk1(i) <= gcvm then
      ii = i
gcvm = wrk1(i)
next i
rem calculate the slope of badly violation.
gslp = 0.
for i = 1 to ndv
   gslp = gslp + dg(i, i) * s(i)
next i
rem calculate the alph.
if gslp = 0. then gslp = zro
alpha = -gcvm/gslp
rem update x for alph.
gosub 7100
gosub 7000
gosub 7100
gosub 7000
gosub 7100
gosub 7000
gosub 7100
gosub 7000
gosub 7100
gosub 4200
gosub 4200
rem find the search direction.
gosub 4400
rem print the results
print ** ******** data **********
print **
rem print the number of design variables = ndv
rem print the number of inequality constraints = nvc
print 'The objective value = ', obj
print '**** design variables ****
for i = 1 to ndv
    print 'x(', i, ') = ', x(i)
next i
print ''
print '***** design variables *****'
for i = 1 to ndv
    print 'x(', i, ') = ', x(i)
next i
print ''
print 'the number of active constraints = ', nac
print ''
print 'the number of violated constraints = ', nvc
print ''
print '**** constraint value ****'
for i = 1 to nvc
    print 'g(', i, ') = ', gcv(i)
next i
print ''
print 'rem default number
mxit = 50 ! maximum iteration number
fdm = .01 ! finite difference step
mdfs = .001 ! maximum absolute finite difference step
vcc = .004 ! violated constraint criteria (thickness)
acc = -.1 ! active constraints criteria (thickness)
th0 = 1. ! push-off factor multiplier (theta zero)
thm = 50. ! maximum value of push-off factor
phid = 100000. ! weighting-factor used in direction when infeasible
accf = .001 ! absolute convergence criteria
acxc = .0001 ! absolute convergence criteria.
zero = .00001! defined zero
espl = .0005 ! used to prevent division by zero
bno = 1.e+70 ! the value of low boundary
bup = 1.e+70 ! the value of upper boundary
dalp = .01 ! step size of alpha in one-dimensional search
abcj = 0.1 ! step size for reduce objective
alpx = .1 ! reduce the design variable factor
ndvm = 21 ! the number of maximum design variable
nigm = 51 ! the number of maximum inequality constraints
return
end
LIST OF REFERENCES


BIBLIOGRAPHY


<table>
<thead>
<tr>
<th>No.</th>
<th>Copies</th>
<th>Name and Address</th>
</tr>
</thead>
</table>
| 1. | 2 | Defense Technical Information Center  
Defense Technical Information Center  
Cameron Station  
Alexandria, Virginia 22314 |
| 2. | 2 | Library, Code 0142  
Naval Postgraduate School  
Monterey, California 93943 |
| 3. | 1 | Department Chairman, Code 54  
Dept. of Operations Research  
Naval Postgraduate School  
Monterey, California 93943 |
| 4. | 3 | Professor G. N. Vanderplaats,  
Dept. of Mechanical Engineering  
Univ. of California at Santa Barbara  
California 93106 |
| 5. | 2 | Professor R. K. Wood, Code 55 Wd  
Dept. of Operations Research  
Naval Postgraduate School  
Monterey, California 93943 |
| 6. | 1 | Dr. H. Miura  
M.S. 237-11  
NASA Ames Research Center  
Moffett Field, California 94035 |
| 7. | 1 | Professor Noriyaki Yoshida  
Dept. of Civil Engineering  
419 Teine Maeda, Nishi-ku  
Sapporo, Japan 061-24 |
| 8. | 1 | Professor Yong S. Shin, Code 69 Sh  
Dept. of Mechanical Engineering  
Naval Postgraduate School  
Monterey, California 93943 |
| 9. | 1 | Professor T. Sarpkaya, Code 69 Sh  
Dept. of Mechanical Engineering  
Naval Postgraduate School  
Monterey, California 93943 |
| 10. | 1 | Professor Robert R. Nunn, Code 69 Mn  
Dept. of Mechanical Engineering  
Naval Postgraduate School  
Monterey, California 93943 |
| 11. | 1 | Professor M. D. Kelleher, Code 69 Kh  
Dept. of Mechanical Engineering  
Naval Postgraduate School  
Monterey, California 93943 |
| 12. | 1 | Professor Gilles Cantine, Code 69 Ci  
Dept. of Mechanical Engineering  
Naval Postgraduate School  
Monterey, California 93943 |
13. Professor Seong Hwan, Cho  
Dept. Mechanical Engineering  
PO Box 77, Kong Neung Dong  
Pc Pong Fu, Seoul, Korea 130-09

14. Dong Soo, Kim  
86-15 Hyoja 1 Acng  
Chuncheon City Kangwoondo.  
Seoul Korea 200

15. Personnel Management Office  
Army Headquarters  
Seoul, Korea 140-01

Embassy of the Republic of Korea  
2320 Massachusetts Avenue, Northwest  
Washington, D.C. 20008

17. Library Officer  
Korea Military Academy  
Seoul, Korea 130-09
END

FILMED

5-85

DTIC