REPORT NO.
UCB/EERC-79/16
JULY 1979

EARTHQUAKE ENGINEERING RESEARCH CENTER

OPTDYN — A GENERAL PURPOSE OPTIMIZATION PROGRAM FOR PROBLEMS WITH OR WITHOUT DYNAMIC CONSTRAINTS

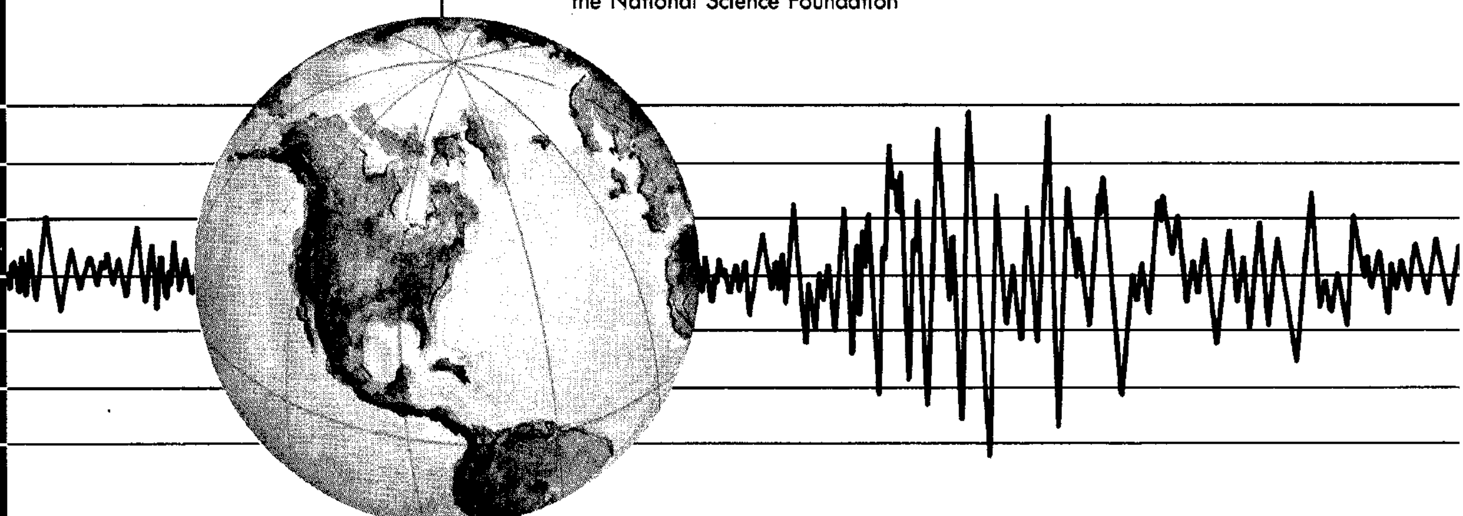
by

M. A. BHATTI

E. POLAK

K. S. PISTER

A report on research sponsored by
the National Science Foundation



COLLEGE OF ENGINEERING

UNIVERSITY OF CALIFORNIA · Berkeley, California

BIBLIOGRAPHIC DATA SHEET	1. Report No. NSF/RA-790396	2.	3. Recipient's Accession No. PB 80 167091
4. Title and Subtitle OPTDYN - A General Purpose Optimization Program for Problems With or Without Dynamic Constraints			5. Report Date July 1979
			6.
7. Author(s) M.A. Bhatti, E. Polak, K.S. Pister			8. Performing Organization Rept. No UCB/EERC-79/16
9. Performing Organization Name and Address Earthquake Engineering Research Center University of California, Richmond Field Station 47th and Hoffman Blvd. Richmond, California 94804			10. Project/Task/Work Unit No.
			11. Contract/Grant No. ENV76-04264
12. Sponsoring Organization Name and Address National Science Foundation 1800 G Street, N.W. Washington, D.C. 20550			13. Type of Report & Period Covered
			14.

15. Supplementary Notes

16. Abstracts

This report presents a general purpose optimization program for problems with or without dynamic (also called functional) constraints, such as those arising in the design of dynamically loaded structures and in designing controllers for linear multivariable systems using frequency response techniques. The program is based on an algorithm of the feasible directions type; a short description is included. It is written in FORTRAN IV language and runs on a CDC 6400 computer.

Detailed description of logic of the main program and instructions for writing the user-supplied subroutines to define a particular problem are included. Three sample problems chosen from different fields are given to clarify the use of the program. Listings of the main program and user-supplied subroutines for two of the sample problems are given in the appendices.

17b. Identifiers/Open-Ended Terms

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report) UNCLASSIFIED	21. No. of Pages 92
Release unlimited	20. Security Class (This Page UNCLASSIFIED	22. Price

OPTDYN - A GENERAL PURPOSE OPTIMIZATION PROGRAM FOR PROBLEMS WITH OR WITHOUT DYNAMIC CONSTRAINTS

by

M. A. Bhatti

E. Polak

and

K. S. Pister

Prepared under the sponsorship of the
National Science Foundation
Grant ENV76-04264

Report No. UCB/EERC-79/16
Earthquake Engineering Research Center
College of Engineering
University of California
Berkeley, California 94720

July 1979

ABSTRACT

This report presents a general purpose optimization program for problems with or without dynamic (also called functional) constraints, such as those arising in the design of dynamically loaded structures and in designing controllers for linear multivariable systems using frequency response techniques. The program is based on an algorithm of the feasible directions type; a short description is included. It is written in FORTRAN IV language and runs on a CDC 6400 computer.

Detailed description of logic of the main program and instructions for writing the user-supplied subroutines to define a particular problem are included. Three sample problems chosen from different fields are given to clarify the use of the program. Listings of the main program and user-supplied subroutines for two of the sample problems are given in the appendices.

ACKNOWLEDGMENTS

This research was conducted under National Science Foundation Grant ENV 76-04264. Computing facilities were provided by the University of California, Berkeley. The authors are grateful for this support.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Preliminary Remarks

With recent developments in computer science, mathematical programming techniques have become an indispensable tool for solution of practical problems in a wide variety of fields. A number of algorithms and computer codes exist to solve linear and nonlinear programming problems. The nonlinear programming problem treated most often is of the form:

$$\min_{\mathbf{z}}\{f^{o}(\mathbf{z}) \mid g^{j}(\mathbf{z}) \leq 0,\ j=1,\ldots,L\} \tag{1.1.1}$$

where $\mathbf{z} \in \mathbb{R}^{P}$ is the variable vector to be optimized, $f^{o} : \mathbb{R}^{P} \to \mathbb{R}$ is the objective function and $g^{j} : \mathbb{R}^{P} \to \mathbb{R}$, $j=1,\ldots,L$ are inequality constraints. Strict equality constraints may also be included.

A class of problems, such as those arising in the design of dynamically loaded structures [1,2] and in designing controllers for linear multivariable systems using frequency response techniques [3], can be expressed as:

$$\min_{\mathbf{z}}\{f^{o}(\mathbf{z}) \mid \max_{t\in T}(\varphi^{j}(\mathbf{z},t)) \leq 0,\ j\in J_m\ ;\ g^{j}(\mathbf{z}) \leq 0\ j\in J_l\} \tag{1.1.2}$$

where

$\varphi^{j} : \mathbb{R}^{P}\times\mathbb{R} \to \mathbb{R}\times\mathbb{R}$ are known as functional or dynamic constraints;

$T = [t_0\ ,\ t_f] \in \mathbb{R}$ is a compact interval;

$J_m = \{1,\ldots,M\}$;
$J_l = \{1,\ldots,L\}$.

If (1.1.2) were to be solved by using algorithms for solving (1.1.1), the functional constraints would represent infinitely many constraints. Even if it is assumed that the interval T is discretized to utilize a digital computer, the discretization would have to be small enough to insure a reasonable accuracy, which again would imply a very large number of constraints.

Recently a number of algorithms has been proposed to solve the problem

(1.1.2) directly, see references [3,4,5,6]. This report presents an implementation of the algorithm given in references [4,5].

1.2 Outline of the Report

The purpose of this report is to present a computer program implementing the algorithm presented in [4,5]. The computer program is written in FORTRAN IV language for a CDC 6400 computer. Section 2 presents the basic algorithm and necessary theoretical background . Section 3 describes the logic of the computer program, explains the function of different subroutines and gives detailed instructions for adding user's subroutines to define a particular problem. Section 4 gives some sample applications of the program. Problems from different fields are chosen to demonstrate the wide application of the program as well as to give the user a feel for the number of input parameters required by the program. Instructions on preparing input data for the program are included in Appendix A. A listing of the program is given in Appendix B. Appendices C and D give listings of the user-supplied subroutines for two of the sample problems to clarify the structure of these subroutines.

2. OPTIMIZATION ALGORITHM

This section presents an algorithm of the feasible directions type for the solution of nonlinear programming problems with functional inequality constraints (or dynamic constraints). The basic algorithm is due to Gonzaga, Polak and Trahan [5]. A short description of the algorithm is followed by detailed discussion of computational considerations. No convergence proof is given; readers interested in mathematical details and convergence proof are referred to the original paper.

2.1 Definitions and Preliminaries

The nonlinear programming problem with functional inequality constraints is defined as

$$\min_{\mathbf{z}} f^0(\mathbf{z})$$

subject to

$$\max_{t\in T} \varphi^j(\mathbf{z},t) \leq 0 \,,\ j \in J_m \qquad (2.1.1)$$

$$g^j(\mathbf{z}) \leq 0 \,, j \in J_l$$

where

T = $[t_0, t_f]$, specified time interval;

$J_l = \{1,2,\ldots,L\}$;

$J_m = \{1,2,\ldots,M\}$;

L = total number of conventional inequality constraints;

M = total number of functional inequality constraints;

$\mathbf{z} \in \mathbb{R}^P$ = the vector of optimization variables ;

P = total number of optimization variables;

$f^0:\mathbb{R}^P \to \mathbb{R}$ and $g^j:\mathbb{R}^P \to \mathbb{R}, j \in J_l$ are continuously differentiable functions in **z**. $\varphi^j:\mathbb{R}^P \times \mathbb{R} \to \mathbb{R} \times \mathbb{R}$, $j \in J_m$ are continuously differentiable functions in **z** and continuous in t.

The feasible domain, F, is defined by:

$$F = \left\{ \mathbf{z} \in \mathbb{R}^p \mid \max_{t \in T} \varphi^j(\mathbf{z},t) \leqq 0 \, , \, j \in J_m \;\; ; \;\; g^j(\mathbf{z}) \leqq 0 \, , \, j \in J_l \right\}.$$

The interval T is discretized into q+1 points and is denoted by T_q .

Define:

$$\tilde{\Psi}_q(\mathbf{z}) = \max\left\{ \varphi^j(\mathbf{z},t) \, , \, j \in J_m \, , \, t \in T_q \;\; ; \;\; g^j(\mathbf{z}) \, , \, j \in J_l \right\}$$

$$\psi_q(\mathbf{z}) = \max\{0 \, , \, \tilde{\Psi}_q(\mathbf{z})\} \tag{2.1.2}$$

Note that, if $\mathbf{z} \in F$, then $\psi_q(\mathbf{z}) = 0$.

The set of points at which a functional constraint is active is denoted by $\overline{T}^j_{q,\varepsilon}(\mathbf{z})$ and is defined as:

$$\overline{T}^j_{q,\varepsilon}(\mathbf{z}) = \left\{ t \in T_q \mid \varphi^j(\mathbf{z},t) - \psi_q(\mathbf{z}) \geqq -\varepsilon \right\} , \, j \in J_m \, .$$

Next, define the intervals $I^j_{q,\varepsilon,k}(\mathbf{z}) \subset \overline{T}^j_{q,\varepsilon}(\mathbf{z})$ $k = 1,2,\ldots,k^j_{q,\varepsilon}(\mathbf{z})$, $j \in J_m$ recursively, as follows.

To define the first interval, $I^j_{q,\varepsilon,1}(\mathbf{z})$, let t_1 be the smallest number in $\overline{T}^j_{q,\varepsilon}(\mathbf{z})$ and let n_1 be the largest integer such that $(t_1 + n_1 \Delta t) \in \overline{T}^j_{q,\varepsilon}(\mathbf{z})$, but $\left[t_1 + (n_1+1)\Delta t \right] \not\subset \overline{T}^j_{q,\varepsilon}(\mathbf{z})$, where $\Delta t = (t_f - t_0)/q$.

Then

$$I^j_{q,\varepsilon,1}(\mathbf{z}) = \left\{ t_1 , \, t_1+\Delta t , \, t_1+2\Delta t , \, \cdots \, , \, t_1+n_1\Delta t \right\}.$$

Next suppose that $I^j_{q,\varepsilon,k}(\mathbf{z})$ have been defined for k = 1,2,..., k_1 , then $I^j_{q,\varepsilon,(k_1+1)}(\mathbf{z})$ is defined as follows:

Let $t_{k_1+1} \in \overline{T}^j_{q,\varepsilon}(\mathbf{z})$ be the smallest number such that $t_{k_1+1} \not\subset \bigcup_{k=1}^{k_1} I^j_{q,\varepsilon,k}(\mathbf{z})$ and let n_{k_1+1} be the smallest integer such that

$$\left[t_{k_1+1} + n_{k_1+1}\Delta t\right] \in \overline{T}^j_{q,\varepsilon}(\mathbf{z})$$

but

$$\left[t_{k_1+1} + (n_{k_1+1}+1)\,\Delta t\right] \not\in \overline{T}^j_{q,\varepsilon}(\mathbf{z})\,.$$

Then

$$I^j_{q,\varepsilon,(k_1+1)}(\mathbf{z}) \;=\; \left\{t_{k_1+1}\;,\; t_{k_1+1}+\Delta t\;,\; t_{k_1+1}+2\Delta t\;,\;\cdots\;,t_{k_1+1}+n_{k_1+1}\Delta t\right\}.$$

For convenience, define

$$K^j_{q,\varepsilon}(\mathbf{z}) \;=\; \left\{1,2,\;.\;.\;.\;,k^j_{q,\varepsilon}(\mathbf{z})\right\}.$$

Note that

$$\overline{T}^j_{q,\varepsilon}(\mathbf{z}) \;=\; \bigcup_{k\in K^j_{q,\varepsilon}(\mathbf{z})} I^j_{q,\varepsilon,k}(\mathbf{z})\,.$$

The point at which a functional constraint is maximum in each of the above defined intervals is defined as:

$$t^j_{q,\varepsilon,k}(\mathbf{z}) \;=\; \left\{t^* \in I^j_{q,\varepsilon,k}(\mathbf{z}) \mid \varphi^j(\mathbf{z},t^*) \geq \varphi^j(\mathbf{z},t)\;,\; t\in I^j_{q,\varepsilon,k}(\mathbf{z})\right\}\;\; k\in K^j_{q,\varepsilon}(\mathbf{z}).$$

The set of points at which a functional constraint is a local maximum is defined as:

$$T^j_{q,\varepsilon}(\mathbf{z}) \;=\; \bigcup_{k\in K^j_{q,\varepsilon}(\mathbf{z})} t^j_{q,\varepsilon,k}(\mathbf{z})\,. \tag{2.1.3}$$

Figure 1 gives an illustration of these sets by taking a hypothetical example.

Now, the " ε - active constraint index " set for the functional constraints is defined as follows:

$$J^{\varphi}_{\varepsilon,q}(\mathbf{z}) \;=\; \left\{(j\;,\;t) \mid j\in J_m\;,\; t\in T^j_{q,\varepsilon}(\mathbf{z})\right\}. \tag{2.1.4}$$

The ε - active constraint index set for conventional inequality constraints is defined by:

$$J^{g}_{\varepsilon,q}(\mathbf{z}) \;=\; \left\{j \mid g^j(\mathbf{z}) - \psi_q(\mathbf{z}) \geq -\,\varepsilon\;,\; j\in J_l\right\}. \tag{2.1.2}$$

The optimality function $\vartheta_{\varepsilon,q}(\mathbf{z}) : \mathbb{R}^p \to \mathbb{R}$ for the nonlinear programming problem (2.1.1) is defined as follows:

$$\vartheta_{\varepsilon,q}(\mathbf{z}) = \min_{\mathbf{h}\in\mathbb{R}^p} \left[\frac{1}{2} ||\mathbf{h}||_2^2 + \max\left\{ <\nabla f^o(\mathbf{z}),\mathbf{h}> - \gamma\psi_q(\mathbf{z}) \; ; \right.\right.$$

$$<\nabla g^j(\mathbf{z}),\mathbf{h}> , \; j \in J^g_{\varepsilon,q}(\mathbf{z}) \; ;$$

$$\left.\left. <\nabla_z \varphi^j(\mathbf{z},t),\mathbf{h}> , \; (j,t) \in J^\varphi_{\varepsilon,q}(\mathbf{z}) \right\} \right] . \qquad (2.1.6)$$

The dual form of (2.1.6), which is actually used in the following algorithm, is as follows:

$$\vartheta_{\varepsilon,q}(\mathbf{z}) = \max_{\mu\geq 0} \left[-\frac{1}{2} || \sum_{j\in J^g_{\varepsilon,q}(\mathbf{z})} \mu_g^j \nabla g^j(\mathbf{z}) + \sum_{(j,t)\in J^\varphi_{\varepsilon,q}(\mathbf{z})} \mu_\varphi^{j,t} \nabla_z \varphi^j(\mathbf{z},t) + \right.$$

$$\left. \mu^o \nabla f^o(\mathbf{z}) ||_2^2 - \gamma\mu^o \psi_q(\mathbf{z}) \; \Big| \sum_{j\in J^g_{\varepsilon,q}(\mathbf{z})} \mu_g^j + \sum_{(j,t)\in J^\varphi_{\varepsilon,q}(\mathbf{z})} \mu_\varphi^{j,t} + \mu^o = 1 \right] (2.1.7)$$

and

$$-\mathbf{h}_{\varepsilon,q}(\mathbf{z}) = \sum_{j\in J^g_{\varepsilon,q}(\mathbf{z})} \mu_g^j \nabla g^j(\mathbf{z}) + \sum_{(j,t)\in J^\varphi_{\varepsilon,q}(\mathbf{z})} \mu_\varphi^{j,t} \nabla_z \varphi^j(\mathbf{z},t) + \mu^o \nabla f^o(\mathbf{z}) . \qquad (2.1.8)$$

where

$\nabla f(\mathbf{x})$ denotes the gradient of function $f : \mathbb{R}^p \to \mathbb{R}$ at $\mathbf{x}$. The gradient vector is treated as a column vector.

$<.,.>$ denotes the scalar product in $\mathbb{R}^p$ and is defined by $<\mathbf{x} , \mathbf{y}> = \sum_{i=1}^{p} x_i y_i$.

$||.||_2$ denotes the Euclidean norm in $\mathbb{R}^p$ and is defined by $|| \mathbf{x} ||_2 = \sqrt{<\mathbf{x} , \mathbf{x}>}$.

Theorem [5]

If $\mathbf{z}$ is optimal for nonlinear programming problem (2.1.1), then the function $\vartheta_{0,q}(\mathbf{z})$ given by Equation (2.1.7) is equal to zero.

2.2 A Feasible Directions Algorithm

A feasible directions algorithm for the solution of the nonlinear programming problem (2.1.1) can now be presented.

Algorithm

DATA: $\alpha \in (0,1)$, $\beta \in (0,1)$, $\gamma \geq 1$

$\delta \in (0,1]$, $\varepsilon_0 > 0$

$\mu_1 > 0$, $\mu_2 > 0$, $M > 0$

q_0 , $q_{\max} \geq q_0$, $\mathbf{z}_0 \in \mathbb{R}^P$.

STEP 0: Set i = 0 , q = q_0.

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

STEP 2: Compute $\left[\vartheta_{\varepsilon,q}(\mathbf{z}^i) , \mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)\right]$ by solving (2.1.7) and (2.1.8).

STEP 3: If $\vartheta_{\varepsilon,q}(\mathbf{z}^i) \leq -2\varepsilon\delta$, go to step 6; Else set $\varepsilon = \varepsilon/2$ and go to step 4.

STEP 4: If $\varepsilon < \varepsilon_0 \dfrac{\mu_1}{q}$ and $\psi_q(\mathbf{z}^i) < \dfrac{\mu_2}{q}$, set q = 2q and go to step 5; Else go to step 2.

STEP 5: If $q > q_{\max}$, STOP; Else, go to step 1.

STEP 6: Compute the largest step size $\lambda(\mathbf{z}^i) = \beta^{k(\mathbf{z}^i)} \in (0, M^*]$, where $M^* = \max\left\{1, \dfrac{M}{||\mathbf{h}_{\epsilon,q}(\mathbf{z}^i)||_\infty}\right\}$ and $k(\mathbf{z}^i)$ is an integer, such that

(i) if $\mathbf{z}^i \in F^C$ (the complement of F in $\mathbb{R}^P$)

$$\psi_q\left[\mathbf{z}^i + \lambda(\mathbf{z}^i)\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)\right] - \psi_q(\mathbf{z}^i) \leq -\alpha\lambda(\mathbf{z}^i)\delta\varepsilon,$$

(ii) if $\mathbf{z}^i \in F$

$$f^0\left[\mathbf{z}^i + \lambda(\mathbf{z}^i)\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)\right] - f^0(\mathbf{z}^i) \leq -\alpha\lambda(\mathbf{z}^i)\delta\varepsilon$$

$$g^j\left[\mathbf{z}^i + \lambda(\mathbf{z}^i)\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)\right] \leq 0 \quad j \in J_l$$

$$\varphi^j\left[\mathbf{z}^i + \lambda(\mathbf{z}^i)\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i), t\right] \leq 0 \,,\ j \in J_m \,,\ t \in T_q .$$

STEP 7: Set $\mathbf{z}^{i+1} = \mathbf{z}^i + \lambda(\mathbf{z}^i)\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)$. Set i = i+1 and go to Step 2.

Remark

The algorithm as presented above does not require an initial feasible point. If $\mathbf{z}_0 \notin F$, then $\psi_q(\mathbf{z}_0)$ is non-zero and the algorithm constructs a sequence of points which forces the point into the feasible domain. This aspect of the algorithm is very advantageous in the case of complicated problems where the choice of an initial feasible point is not obvious. For example, in earthquake-resistant design if the relative drift of a particular story in a framed structure is to be limited to a certain value, it is not easy to find an initial design that will satisfy that requirement. Of course, the algorithm is more efficient if one can start from an initial feasible point.

2.3 Explanation of the Algorithm

The algorithm has two distinct phases. First, a direction is computed by solving (2.1.7) and (2.1.8). A step is then taken in this direction in such a way that, if the current $\mathbf{z}$ is in the feasible domain, there is a maximum reduction in the objective function while still maintaining feasibility. When the current point is outside the feasible domain, the step length is chosen so as to move as close to the feasible domain as possible.

Direction Finding Subproblem

As noted, a feasible direction is found by solving the problem:

$$\vartheta_{\varepsilon,q}(\mathbf{z}) = \max_{\mu\geq 0}\left[-\frac{1}{2}\Big|\Big| \sum_{j\in J^g_{\varepsilon,q}(\mathbf{z})} \mu_g^j \nabla g^j(\mathbf{z}) + \sum_{(j,t)\in J^\varphi_{\varepsilon,q}(\mathbf{z})} \mu_\varphi^{j,t}\nabla_z \varphi^j(\mathbf{z},t) + \right.$$

$$\left. \mu^o \nabla f^o(\mathbf{z})\Big|\Big|_2^2 - \gamma\mu^o\psi_q(\mathbf{z}) \;\Big|\; \sum_{j\in J^g_{\varepsilon,q}(\mathbf{z})} \mu_g^j + \sum_{(j,t)\in J^\varphi_{\varepsilon,q}(\mathbf{z})} \mu_\varphi^{j,t} + \mu^o = 1\right] \quad (2.3.1)$$

and then computing the direction from

$$-\mathbf{h}_{\varepsilon,q}(\mathbf{z}) = \sum_{j\in J^g_{\varepsilon,q}(\mathbf{z})} \mu_g^j \nabla g^j(\mathbf{z}) + \sum_{(j,t)\in J^\varphi_{\varepsilon,q}(\mathbf{z})} \mu_\varphi^{j,t}\nabla_z \varphi^j(\mathbf{z},t) + \mu^o \nabla f^o(\mathbf{z}) . \quad (2.3.2)$$

Equation (2.3.1) can be transcribed into a standard quadratic programming problem as follows. Let k_g be the total number of points in $J^g_{\varepsilon,q}(\mathbf{z})$ and (j_φ, l_φ) be the total number of points in $J^\varphi_{\varepsilon,q}(\mathbf{z})$. Define the vector $\mu \in \mathbb{R}^{1+k_g+j_\varphi l_\varphi}$ as follows:

$$\mu^T = \left[\mu^o, \mu_g^{k_1}, \mu_g^{k_2}, \ldots, \mu_g^{k_g}, \mu_\varphi^{j_1, l_1}, \ldots, \mu_\varphi^{j_\varphi, l_\varphi}\right]. \tag{2.3.3}$$

where

$k_i \in J^g_{\varepsilon,q}(\mathbf{z})$ for $i = 1, \ldots, k_g$
$(j_i, l_j) \in J^\varphi_{\varepsilon,q}(\mathbf{z})$ for $i = 1, \ldots, j_\varphi$ $j = 1, \ldots, l_\varphi$

Define the matrix $\mathbf{A} \in \mathbb{R}^{1+k_g+j_\varphi l_\varphi} \times \mathbb{R}^P$ as:

$$\mathbf{A} = \begin{bmatrix} \left[\nabla f^o(\mathbf{z})\right]^T \\ \left[\nabla g^{k_1}(\mathbf{z})\right]^T \\ \vdots \\ \left[\nabla g^{k_g}(\mathbf{z})\right]^T \\ \left[\nabla_z \varphi^{j_1}(\mathbf{z}, t^{j_1}_{q,\varepsilon,l_1})\right]^T \\ \vdots \\ \left[\nabla_z \varphi^{j_\varphi}(\mathbf{z}, t^{j_\varphi}_{q,\varepsilon,l_\varphi})\right]^T \end{bmatrix}. \tag{2.3.4}$$

Then Equation (2.3.1) can be written as:

$$\max_{\mu \geq 0} \left[-\frac{1}{2}(\mu^T \mathbf{A})\,(\mu^T \mathbf{A})^T - \gamma \mu^o \psi_q(\mathbf{z}) \;\middle|\; \sum_{j=0}^{1+k_g+j_\varphi l_\varphi} \mu^j = 1 \right]$$

or

$$\min_{\mu \geq 0} \left[\frac{1}{2} \mu^T \mathbf{A}\mathbf{A}^T \mu + \gamma \mu^o \psi_q(\mathbf{z}) \;\middle|\; \sum_{j=0}^{1+k_g+j_\varphi l_\varphi} \mu^j = 1 \right]. \tag{2.3.5}$$

Define a vector $\mathbf{D} \in \mathbb{R}^{1+k_g+j_\varphi l_\varphi}$ such that

$$\mathbf{D}^T = \left[\gamma \psi_q(\mathbf{z}), 0, 0, \cdots\right] \tag{2.3.6}$$

and a matrix $\mathbf{Q} \in \mathbb{R}^{1+k_g+j_\varphi l_\varphi} \times \mathbb{R}^{1+k_g+j_\varphi l_\varphi}$ by

$$\mathbf{Q} = \mathbf{A}\mathbf{A}^T. \tag{2.3.7}$$

Then Equation (2.3.5) can be written as:

$$\min_{\mu \geq 0} \left[\frac{1}{2} \mu^T \mathbf{Q} \mu + \mathbf{D}^T \mu \mid \sum_{j=1}^{1+k_q+j_\varphi l_\varphi} \mu^j = 1 \right] \tag{2.3.8}$$

which is a standard quadratic programming problem. Once μ 's are obtained by solving (2.3.8), the direction is computed from

$$-\mathbf{h}_{\varepsilon,q}(\mathbf{z})^T = \mu^T \mathbf{A} . \tag{2.3.9}$$

Step Length Computation

After a feasible direction is obtained, the next step is to compute the step length in that direction. If the current design is inside the feasible domain the step length should be chosen in such a way that there is a maximum reduction in the objective function, while still maintaining feasibility. When the current design is outside the feasible domain, the objective is to take a step such that the new design is as close to the feasible domain as possible. The step size calculations begin by minimizing the objective function along the feasible direction and then checking whether any of the constraints is violated. If any of the constraints is violated, the step length is reduced and the process repeated until the new design satisfies all of the constraints. A number of methods are available for this unidirectional search, the most popular among them being Fibonacci search, Newton's method, quadratic or cubic fit, etc. [7,8]. For general non-convex problems, these methods tend to be very expensive. Since computation of the exact minimum along the feasible direction is not absolutely necessary, an approximate line search technique, known as the Armijo step size rule, is often used [7,9]. The method performs only an approximate line search and is quite efficient for general non-convex problems. The method is as follows.

Given the constants α, δ, ε, β, M, current design vector $\mathbf{z}^i$, $\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)$ and $\psi_q(\mathbf{z}^i)$, compute the largest step size $\lambda(\mathbf{z}^i) = \beta^{k(\mathbf{z}^i)} \in (0, M^*]$ where

$$M^* = \max \left\{ 1 , \frac{M}{||\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)||_\infty} \right\} ,$$

such that

(i) if $\psi_q(\mathbf{z}^i) > 0$ (i.e. $\mathbf{z}^i \notin F$), then

$$\psi_q\left[\mathbf{z}^i + \lambda(\mathbf{z}^i)\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)\right] - \psi_q(\mathbf{z}^i) \leqq -\alpha\lambda(\mathbf{z}^i)\delta\varepsilon;$$

(ii) if $\psi_q(\mathbf{z}^i) = 0$, i.e. $\mathbf{z}^i \in F$, then

$$f^o\left[\mathbf{z}^i + \lambda(\mathbf{z}^i)\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)\right] - f^o(\mathbf{z}^i) \leqq -\alpha\lambda(\mathbf{z}^i)\delta\varepsilon ,$$

$$g^j\left[\mathbf{z}^i + \lambda(\mathbf{z}^i)\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)\right] \leqq 0 \quad j \in J_l ,$$

$$\varphi^j\left[\mathbf{z}^i + \lambda(\mathbf{z}^i)\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i) \, , \, t\right] \leqq 0 \quad j \in J_m \, , \, t \in T_q \, .$$

The algorithm to implement the above process is as follows.

STEP 1: Set $\lambda = \beta$. Compute $M^* = \max\left\{1, \frac{M}{||\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)||_\infty}\right\}$. Set FLAG = 0. Set n = 0.

STEP 2: Compute $\mathbf{z}_n^{i+1} = \mathbf{z}^i + \lambda\mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)$.

STEP 3: If $\psi_q(\mathbf{z}^i) > 0$, go to step 5. Else , go to step 4.

STEP 4: Compute $f^o(\mathbf{z}_n^{i+1})$. If $f^o(\mathbf{z}_n^{i+1}) + \alpha\lambda\delta\varepsilon \leqq -f^o(\mathbf{z}^i)$, go to step 6. Otherwise, go to step 8.

STEP 5: If $\psi_q(\mathbf{z}_n^{i+1}) + \alpha\lambda\delta\varepsilon \leqq \psi_q(\mathbf{z}^i)$, go to step 7. Otherwise, go to step 8.

STEP 6: Compute $g^j(\mathbf{z}_n^{i+1}), j \in J_l$ and $\varphi^j(\mathbf{z}_n^{i+1}, t), j \in J_m \; t \in T_q$. If $g^j(\mathbf{z}_n^{i+1}) \leqq 0, j \in J_l$ and $\varphi^j(\mathbf{z}_n^{i+1}, t) \leqq 0 \; j \in J_m, t \in T_q$, go to step 7. Otherwise, go to step 8.

STEP 7: If $\lambda / \beta > M^*$ or FLAG = -1, go to step 9. Otherwise, set $\lambda = \lambda / \beta$, FLAG = 1, n = n + 1 and go to step 2.

STEP 8: Set $\lambda = \lambda \beta$. If FLAG = 1, got to step 9. Otherwise, set FLAG = -1, n = n + 1 and go to step 2.

STEP 9: Set $\lambda = \lambda^*$ and the new design vector is $\mathbf{z}^{i+1} = \mathbf{z}^i + \lambda^* \mathbf{h}_{\varepsilon,q}(\mathbf{z}^i)$.

2.4 Computational Considerations

The quadratic programming problem as formulated in Equation (2.3.8) may be computationally ill-posed because of different magnitudes of the gradients of different functions. Proper scaling is therefore essential to make the problem computationally efficient. In the present version the following scaling was used. Define

$$\begin{aligned} s_g^j &= \|\nabla g^j(\mathbf{z})\|_\infty , \; j \in J_{\varepsilon,q}^g(\mathbf{z}) ; \\ s_\varphi^{j,t} &= \|\nabla_z \varphi^j(\mathbf{z},t)\|_\infty , \; (j,t) \in J_{\varepsilon,q}^\varphi(\mathbf{z}) ; \\ s_o &= \|\nabla f^o(\mathbf{z})\|_\infty . \end{aligned} \tag{2.4.1}$$

where

$\| \, . \, \|_\infty$ is the maximum norm in $\mathbb{R}^P$ defined by

$$\| \, \mathbf{x} \, \|_\infty = \max_{i \in \mathbb{R}^P} |x_i| .$$

The matrix **A** defined in (2.3.4) is scaled as follows.

$$\mathbf{A} = \begin{bmatrix} \left[\nabla f^o(\mathbf{z})\right]^T / s_o \\ \left[\nabla g^{k_1}(\mathbf{z})\right]^T / s_g^{k_1} \\ \vdots \\ \left[\nabla_z \varphi^{j_1}(\mathbf{z}, t_{q,\varepsilon,l_1}^{j_1})\right]^T / s_\varphi^{j_1,l_1} \\ \vdots \end{bmatrix} . \tag{2.4.2}$$

Define a vector $\mathrm{R} \in \mathbb{R}^{1+k_g+j_\varphi l_\varphi}$ as

$$\mathbf{R} = \left[\rho_o, \rho_g^{k_1}, \ldots, \rho_g^{k_g}, \rho_\varphi^{j_1,l_1}, \ldots, \rho_\varphi^{j_\varphi,l_\varphi}\right]^T \tag{2.4.3}$$

where ρ_o, ρ_g^j and $\rho_\varphi^{j,l}$ are called " push-off " factors and can be adjusted to force the direction vector toward or away from a constraint. If any of these factors is large as compared to the rest, then the constraint corresponding to that factor will dominate the direction finding problem. If the constraint functions are well scaled, all the push-off factors could be set equal to one, in which case all the active constraints will get equal importance. For a general case the following scheme of choosing the push-off factors seems to work well:

$$\rho_o = \xi_o \, (1/s_o - 1) \tag{2.4.4}$$

$$\rho_g^j = \xi_g^j + \eta \left[1 + \frac{g^j(\mathbf{z}) - \psi_q(\mathbf{z})}{\varepsilon}\right]^2 \quad j \in J_l \tag{2.4.5}$$

$$\rho_\varphi^{j,t} = \xi_\varphi^j + \eta \left[1 + \frac{\varphi^j(\mathbf{z},t) - \psi_q(\mathbf{z})}{\varepsilon}\right]^2 \quad t \in T_{q,\varepsilon}^j(\mathbf{z}) \ , \ j \in J_m \tag{2.4.6}$$

where ξ_o , ξ_g^j , ξ_φ^j and η are input parameters.

An arbitrary upper limit of fifty was set for these push-off factors in the present study to prevent any instability in the direction finding process.

With these definitions, the scaled version of the quadratic programming problem (2.3.8) can be written as:

$$\min_{\mu \geq 0} \left\{ \frac{1}{2} \mu^T \mathbf{Q} \, \mu + \mathbf{D}^T \mu \mid \mathbf{R}^T \mu = 1 \right\} \tag{2.4.7}$$

where $\mathbf{Q} = \mathbf{A}\mathbf{A}^T$ with $\mathbf{A}$ defined by (2.4.2) and

$$\mathbf{D}^T = \left[\gamma \psi_q(\mathbf{z}) / s_o, 0, 0, \cdots \right].$$

The direction vector is still computed from Equation (2.4.9).

3. COMPUTER PROGRAM

A computer program called OPTDYN was written in FORTRAN IV language to implement the algorithm described in Section 2. The program runs in single precision on a CDC 6400 computer. This section describes the logic of the computer program and gives instructions for adding the user-supplied subroutines to solve a particular problem.

3.1 Computer Program Logic

The program flow diagram, giving the calling sequence of different subroutines is given in Figure 2. The program is divided into a base program and user-supplied section. The user-supplied section specifies the problem to be solved. The base program calls the user subroutines as needed. The program is structured in such a way that a user need not understand the base program thoroughly in order to solve his particular problem. However, enough information is given in the following pages to make the base program easier to understand and modify if desired.

A brief description of the functions of each subroutine in the base program is given below.

1. OPTDYN:

This is the main program. It calls the subroutines OPDATA and COPFED. The dimensions of arrays needed are set in this program and in QP. The minimum required dimensions of the arrays are given in the listing of the program in the form of comment cards.

2. OPDATA:

This subroutines reads and prints all input data needed in the program. The dimensions of the arrays set are checked with the input data and if they are not sufficient an error message is printed and execution is terminated.

3. COPFED:

This is the main optimization subroutine. Different steps of the algorithm presented in section 2 are identified by means of comment cards. The following subroutines are called, in order, by this subroutine: FUNCF, FUNCG, FUNCPH, QP and ARMIJO. If there are no conventional inequality and/or functional inequality constraints, the respective calls are skipped. A concise flow chart for this subroutine is given in Figure 3.

4. QP:

This subroutine formulates and solves the quadratic programming problem to compute the optimality function, ϑ , and the descent direction, h. It calls subroutines GRADF, AROW, EACTIV, GRADG, GRADPH, WOLFE and ANGLE. A concise flow chart for this subroutine is given in Figure 4.

5. EACTIV:

This subroutine determines the ε - active constraints. For conventional inequality constraints it sets up a vector NEPTG, whose i^{th} entry is zero if the i^{th} constraint is not active, and one if it is active. For functional constraints, it determines the local maxima of the ε - active intervals and sets up a matrix NEPTF whose i^{th} row corresponds to the i^{th} functional constraint and contains the discretization number of the local maxima of ε - active intervals. This information is used in subroutine QP, which makes calls to the gradient evaluation subroutines GRADG and GRADPH only if there is some constraint which is active. Information in array NEPTF can also be used to save storage space required for gradients of functional constraints, with these gradients being saved only at the ε - active points.

6. AROW:

This is a small subroutine which fills in the gradients scaled by their infinity norms into the rows of " A " matrix. The gradients of the cost function is entered in the first row of this matrix. Gradients of active conventional constraints are entered starting from the second row. Gradients of active functional constraints are entered in after the conventional constraint gradients. This subroutine also determines the maximum of all the infinity norms of the gradients.

7. WOLFE:

This is a standard quadratic programming problem solver.

8. ANGLE:

This subroutine computes angles between the direction vector given by QP and the cost function and active constraint gradients. This information can be employed by the user to choose a proper value for the so-called "push-off" factors. By a proper choice of these factors the problem can be scaled in such a way that the user can emphasize any particular constraint or cost in the direction finding process.

9. ARMIJO:

This subroutine computes step length along the usable feasible direction given by QP. An Armijo step size rule is used, as explained in section 2. It calls subroutines FUNCF, FUNCG and FUNCPH. If there are no conventional and/or functional constraints, the corresponding calls are skipped. A concise flow chart of this subroutine is given in Figure 5.

10. ERROR:

Prints input data error messages.

11. TIMLOG:

Prints solution time log at the end of the computer run.

3.2 User-Supplied Subroutines

The subroutines which define a specific problem are separated from the base program and are grouped under user-supplied subroutines. The calling sequence and functions of these subroutines are given below. Note that all the variables identified as input are set in the base program and should not be changed in the user subroutines.

1. FUNCF:

This subroutine evaluates the cost function f^o . It is called from the base program as follows:

CALL FUNCF (N, Z, F, NFUNCF)

where the arguments have the following meaning:

N number of optimization variables, (input);

Z vector containing current values of optimization variables , (input);

F value of the objective function f^o , (output);

NFUNCF a counter, which counts the number of times this subroutine is called, (input);

2. GRADF:

This subroutine evaluates the gradients of the objective function. The calling sequence for this subroutine is:

CALL GRADF (N, Z, GRAD)

where the arguments have the following meaning:

N number of optimization variables, (input);

Z vector containing current values of optimization variables , (input);

GRAD vector containing gradients of objective function , (output). The i^{th} entry in this vector should contain the partial derivative of the objective function with respect to the i^{th} optimization variable.

3. FUNCG:

This subroutine evaluates conventional inequality constraint functions (functions "g"). It is called from the base program as follows:

CALL FUNCG (N, JP, Z, G, PSI, NFUNCG)

where the arguments have the following meaning:

N number of optimization variables, (input);

JP number of constraints of this type, (input);

Z vector containing current values of optimization variables , (input);

G vector of functions "g",having dimension "JP", (output). These functions could be arranged in any order, but the corresponding gradients must follow the same order in subroutine GRADG;

PSI function ψ . At input it is initialized to its proper value by the main program. The maximum of functions g is computed and PSI is set equal to the greater of its input value or the maximum g function value at output. This should be achieved by adding the following FORTRAN statements, just before RETURN.

```
DO 100 I = 1, JP
100 IF (G(I) .GT. PSI) PSI = G(I)
```

NFUNCG a counter which is set equal to the number of the current call to this subroutine, (input).

4. GRADG:

This subroutine evaluates the gradients of conventional inequality constraints (functions g). The calling sequence for this subroutine is:

CALL GRADG (N, J, Z, GRAD)

where the arguments have the following meaning:

N number of optimization variables, (input);

J serial number of the constraint function for which the gradient is to be evaluated. A separate call is made for evaluation of gradient of each function, (input);

Z vector containing current values of optimization variables , (input);

GRAD vector containing gradient of J^{th} , g constraint with respect to the optimization variables. The dimension of this vector is "N". The i^{th} entry in this vector should contain the partial derivative of the J^{th} conventional constraint function with respect to the i^{th} optimization variable, (output).

5. FUNCPH:

This subroutine evaluates dynamic inequality constraint functions (functions φ). It is called from the base program as follows:

CALL FUNCPH (N, NJQ, JQ, Z, WO, WC, DELTAW, NQ, PHI, PSI, NFUNCP)

where the arguments have the following meaning:

N number of optimization variables, (input);

NJQ row dimension of matrix PHI in the main program, (input);

JQ number of constraints of this type, (input);

Z vector containing current values of optimization variables , (input);

WO initial value of the interval over which the functional constraint is to be evaluated, (input);

WC final value of the interval over which the functional constraint is to be evaluated, (input);

NQ number of discretization points, (input);

DELTAW discretization interval, defined as:

$$DELTAW = (WC - WO) / NQ;$$

PHI matrix containing values of functions φ. The i^{th} row of this matrix contains values of i^{th} functional constraint at specified intervals, (output);

PSI function ψ . At input it is initialized to its proper value by the main program. The maximum of functions φ is computed and PSI is set equal to the greater of its input value or the maximum φ function value at output. This should be achieved by adding the following FORTRAN statements, just before RETURN.

```
      DO 100 L = 1, JQ
      DO 100 K = 1, NQ
      IF (PHI(L,K) .GT. PSI) PSI = PHI(L,K)
  100 CONTINUE
```

NFUNCP a counter which is set equal to the number of the current call to this subroutine, (input).

6. GRADPH:

This subroutine evaluates gradients of dynamic inequality constraint functions (functions φ). It is called from the base program as follows:

```
CALL GRADPH (N,NJQ,NACTIV,JQ,WO,WC,DELTAW,NQ,NEPTF,L,Z,K,GRAD,IGRAD)
```

where the arguments have the following meaning:

N number of optimization variables, (input);

NJQ row dimension of matrix NEPTF, (input);

NACTIV column dimension of matrix NEPTF, (input).

JQ number of functional constraints , (input);

WO initial value of the interval over which the functional constraint is to be evaluated, (input);

WC final value of the interval over which the functional constraint is to be evaluated, (input);

NQ number of discretization points, (input);

DELTAW discretization interval, defined as:

DELTAW = (WC - WO) / NQ;

NEPTF matrix of points at which the ε - active intervals have local maxima, as explained earlier, (input);

L serial number of the current functional constraint. A separate call is made for evaluation of gradient of each ε - active point, (input);

Z vector containing current values of optimization variables , (input);

K current discretization point at which the gradient is desired, (input);

GRAD vector containing gradient of φ (L , K). The i^{th} entry in this vector should contain the partial derivative of the L^{th} functional constraint at the K^{th} discretization point with respect to the i^{th} optimization variable, (output);

IGRAD a counter, which is equal to the number of calls to this subroutine in the *current* iteration. At the beginning of every iteration, this is set equal to one, (input).

3.3 Explanation of Variables in Common Blocks

Data are organized in a number of common blocks to be shared by different subroutines. Different common blocks and their constituents are listed below.

1. COMMON /TAPES/NIN, NOU

This block is initialized in the main program.

NIN input tape unit. Its value is initialized to 1.

NOU output tape unit. Its value is initialized to 2.

2. COMMON /DIMNSN/NZ, NJQ, NJP, NQMAX, NACTIV

The data in this block are set in the main program. Change to appropriate values whenever the dimensions are changed.

NZ maximum number of optimization variables for which dimensions are set.

NJQ maximum number of functional constraints for which dimensions are set.

NJP maximum number of conventional inequality constraints for which the dimensions are set.

NQMAX maximum number of discretization points for which dimensions are set.

NACTIV maximum number of rows in the ε - active matrix " A " in QP. This is set to 10. If this requirement is exceeded, the program will print the dimension needed. Any change in the value of NACTIV will require changing the dimensions in the main program and QP. Sometimes reducing ε - band width might drop some of the constraints and set dimensions might be enough.

3. COMMON /OPTDAT/ E0, MAXITN, NCUT, ITRSTP, ITER, SCALE

The data in this block are read from unit NIN in subroutine OPDATA.

E0 initial ε - band width, ε_0 .

MAXITN maximum number of iterations specified. Program will stop either when MAXITN is reached or an optimum is achieved.

NCUT maximum number of iterations in the solution of the quadratic programming problem.

ITRSTP maximum number of iterations allowed in step length calculations.

ITER iteration number at start of this run. This is used only for labeling the output. The iteration number printed with the output starts from ITER and is incremented by one in subsequent iterations.

SCALE a scaling factor for ε - active constraints. This is used in computing push-off factors. (η in section 2.4).

4. COMMON /ONE/ JP, JQ, N

The data in this common block are read from unit NIN in subroutine OPDATA.

JP number of conventional inequality constraints.

JQ number of functional inequality constraints.

N number of optimization variables.

5. COMMON /TWO/ ALPHA, BETA, STPMAX, OLDSTP, ICOUNT

This common block contains data which are used in subroutine ARMIJO for step length calculations.

ALPHA parameter α , input in OPDATA.

BETA parameter β , input in OPDATA

STPMAX maximum step length parameter M, input in OPDATA.

OLDSTP step length at the last iteration. Initially input in OPDATA, later on updated at the end of ARMIJO.

ICOUNT a counter used to monitor the size of the step length. If the step length is less than a certain specified tolerance (1.0E-10) for 10 iterations, the execution is terminated. It is initialized to zero in OPDATA and is updated in ARMIJO.

6. COMMON /THREE/ TOL, TOLER(4), DELTA, MU1, MU2

These convergence tolerance parameters are set in OPDATA and used in ARMIJO and COPFED.

TOL tolerance parameter, set to 1.0E-10.

TOLER tolerance parameters set to 1.0E-10.

DELTA parameter δ .

MU1 real parameter μ_1 .

MU2 real parameter μ_2 .

7. COMMON /FIVE/ W0, WC, Q, DELTAW, QMAX

These values are read from unit NIN in subroutine OPDATA.

W0 initial value of the interval for functional constraints.

WC final value of the interval.

Q integer variable equal to the initial number of discretization points.

DELTAW discretization interval, defined as

$$DELTAW = (WC - W0) / Q.$$

QMAX integer variable equal to the maximum number of discretization points.

8. COMMON /TIMES/ TCONST, TQPT, TARMJT, TTOT

Common block containing elapsed CPU times in different phases of the program. This is initialized and updated in COPFED. Final values are printed in TIMLOG.

TCONST CPU time used in constraint function evaluations.

TQPT CPU time used in direction finding subproblem. This includes time spent in gradient evaluations.

TARMJT CPU time used in step length calculations.

TTOT total time used in a particular run.

9. COMMON /NUMFUN/ NFUNCF, NFUNCG, NFUNCP

This common block contains the number of function evaluations. The variables are initialized in COPFED and are updated in COPFED and ARMIJO. Final values are printed in TIMLOG.

NFUNCF number of objective function evaluations.

NFUNCG number of g function evaluations.

NFUNCP number of φ function evaluations.

10. COMMON /NUMGRD/ NGRADF, NGRADG, NGRADP

This common block contains the number of gradient evaluations. The variables are initialized in COPFED and are updated in QP. Final values are printed in TIMLOG.

NGRADF number of gradient evaluations of objective function.

NGRADG number of gradient evaluations of g constraint functions.

NGRADP number of gradient evaluations of φ constraint functions.

11. COMMON /WORK/ WORK(32)

This is a temporary storage area and can be used in any subroutine. Since it may be used to store some different quantities in another subroutine, it should not be used to transfer data between two subroutines.

4. SAMPLE APPLICATIONS

This section presents a number of example problems to introduce the user to some of the applications of the program. Problems from different fields are selected to show the wide range of applications of the program. Values of convergence parameters used for different problems are given. Although these may not represent the best choice for other applications, they may be a good starting point for problems in which no experience has been acquired.

4.1 A Constrained Minimization Test Problem

The following nonlinear programming problem is solved to test the algorithm and show the user the structure of the user-supplied subroutines. The problem is taken from reference [10].

$$\min_{\mathbf{z}}\{f^{o}(\mathbf{z}) \mid g^{j}(\mathbf{z}) \leqq 0 \quad j=1,\ldots,3\}$$

where

$$\begin{aligned}
f^{o}(\mathbf{z}) &= z_1^2 + z_2^2 + 2z_3^2 + z_4^2 - 5z_1 - 5z_2 - 21z_3 + 7z_4 \\
g^{1}(\mathbf{z}) &= z_1^2 + z_2^2 + z_3^2 + z_4^2 + z_1 - z_2 + z_3 - z_4 - 8 \\
g^{2}(\mathbf{z}) &= z_1^2 + 2z_2^2 + z_3^2 + 2z_4^2 - z_1 - z_4 - 10 \\
g^{3}(\mathbf{z}) &= 2z_1^2 + z_2^2 + z_3^2 + 2z_1 - 2z_2 - z_4 - 5
\end{aligned}$$

The optimal solution given in the reference is

$$\begin{aligned}
\mathbf{z}^{*} &= [0\ ,\ 1\ ,\ 2\ ,\ -1]^{T} \\
f^{o}(\mathbf{z}^{*}) &= -44
\end{aligned}$$

The gradients of the functions are:

$$\begin{aligned}
\nabla f^{o}(\mathbf{z}) &= [2z_1-5 \quad 2z_2-5 \quad 4z_3-21 \quad 2z_4+7]^{T} \\
\nabla g^{1}(\mathbf{z}) &= [2z_1+1 \quad 2z_2-1 \quad 2z_3+1 \quad 2z_4-1]^{T} \\
\nabla g^{2}(\mathbf{z}) &= [2z_1-1 \quad 4z_2 \quad 2z_3 \quad 4z_4-1]^{T} \\
\nabla g^{3}(\mathbf{z}) &= [4z_1+2 \quad 2z_2-1 \quad 2z_3 \quad -1]^{T}
\end{aligned}$$

A listing of the user-supplied subroutines for this problem is given in Appendix C. The following parameters values were used:

$$\mu_1 = 1.0 \quad \mu_2 = 0.01 \quad \delta = 0.001$$

$$\varepsilon_0 = 0.02 \quad \gamma = 2.0 \quad M = 15.0$$
$$\alpha = 0.2 \quad \beta = 0.3 \quad push-off\ factor = 1.0$$

Initial values of variables = $[\ 0\ ,\ 0\ ,\ 0\ ,\ 0\ ]^T$.

The results of the computations are tabulated in Table 1.

4.2 Design of a PID Controller

The control system is shown in Figure 6. The problem is to choose variables z_1, z_2 and z_3 such that the square of the error associated with a unit step input is minimized.

$$f^\circ(\mathbf{z}) = \int_0^\infty e^2(t,\mathbf{z})\, dt$$

The problem can be transformed into the following form, (see references [3,11]).

$$f^\circ(\mathbf{z}) = \frac{z_2\,(\ 122 + 17z_1 - 5z_2 + 6z_3 + z_1 z_3\ ) - 36z_1 + 180z_3 + 1224}{z_2\,(\ 408 + 56z_1 - 50z_2 + 60z_3 + 10z_1 z_3 - 2z_1^2\ )}$$

The following constraint is introduced to ensure closed-loop stability:

$$\varphi(\mathbf{z},\omega) = \mathrm{Im}\ T(\mathbf{z},\omega) - 3.33\ [\mathrm{Re}\ T(\mathbf{z},\omega)]^2 + 1.0$$

where

$$T(\mathbf{z},\omega) = 1 + H(\mathbf{z},j\omega)G(j\omega)$$
$$\omega \in \Omega = [10^{-6}\ ,\ 30]$$
$$0 \leqq z_1 \leqq 100.0$$
$$0.1 \leqq z_2 \leqq 100.0$$
$$0 \leqq z_3 \leqq 100.0$$

A listing of the user-supplied subroutines for this problem is given in Appendix D. The following parameters values were used:

$$\mu_1 = 0.001 \quad \mu_2 = 0.01 \quad \delta = 0.001$$
$$\varepsilon_0 = 0.2 \quad \gamma = 2.0 \quad M = 15.0$$
$$\alpha = 0.2 \quad \beta = 0.3 \quad push-off\ factor = 0.0$$
$$q = 128 \quad q_{\max} = 256$$

Initial values of variables = $[\ 1\ ,\ 1\ ,\ 1\ ]^T$.

The results of the computations are tabulated in Table 2.

4.3 Design of an Earthquake Isolation System

This problem is formulated and solved in detail in reference [1]. The problem consists of minimizing the sum of squares of story shears at the bottom floor level of the frame shown in Figure 7. The maximum displacement at the bottom floor is constrained to be less than 4.0 inches.

Following [1], the problem can be expressed as:

$$\min_{\mathbf{z}} \; z_4$$

subject to

$$\max_{t \in T} \left[\sum_{j=1}^{3} \left\{ K_j \left[u_j(\mathbf{z},t) - u_{j+1}(\mathbf{z},t) \right] \right\}^2 \right] \leq z_4$$

$$\max_{t \in T} \; [u_4(\mathbf{z},t)]^2 \leq 16.0$$

$$z_j > 0 \quad j = 1,4$$

where K_1, K_2 and K_3 are story stiffnesses and u_1, u_2, u_3 and u_4 are floor displacements. The displacements are computed by integrating the equations of motion for the frame. See reference [1] for details of derivation and solution of these equations of motion. The following parameter values were used:

$$\mu_1 = 1.0 \quad \mu_2 = 0.01 \quad \delta = 0.001$$

$$\varepsilon_0 = 0.025 \quad \gamma = 2.0 \quad M = 15.0$$

$$\alpha = 0.2 \quad \beta = 0.3 \quad push\text{-}off\ factor = 0.0$$

$$q = 1500 \quad q_{\max} = 1500$$

$$t_0 = 0 \quad t_f = 15.0$$

The initial values for the optimization variables were as follows

$$\mathbf{z} = [5.0 \, , \, 0.11 \, , \, 0.064 \, , \, 35.0]^T$$

The optimal values were:

$$\mathbf{z} = [4.2773 \, , \, 1.7529 \, , \, 0.005768 \, , \, 9.1509]^T$$

The results are tabulated in Table 3 and the objective function is plotted against the number of iterations in Figure 8.

Iteration	Z_1	Z_2	Z_3	Z_4	$F°(Z)$
0	0	0	0	0	0.0
5	.4474	.4474	1.774	-.6264	-39.026
10	.2973	.5712	1.918	-.7213	-41.379
15	.1876	.6605	1.992	-.7950	-42.603
20	.0649	.7572	2.028	-.8862	-43.313
25	.0140	.8086	2.043	-.9479	-43.754
30	-.150	.8573	2.033	-.9885	-43.847
35	-.0127	.9043	2.022	-.9947	-43.896
40	.0069	.9262	2.018	-.9754	-43.912
45	.0101	.9423	2.014	-.9739	-43.927
50	.00114	.9554	2.011	-.9847	-43.94
55	-.0047	.9780	2.003	-.9985	-43.942
60	.00062	.9840	2.003	-.9922	-43.956
70	.0013	.9919	2.002	-.9954	-43.99
Optimal Solution	0	1	2	-1	-44

Table 1 Solution of the Constrained Minimization Test Problem

Iteration	z_1	z_2	z_3	$F^\circ(Z)$
0	1.0	1.0	1.0	3.1307
5	21.0564	20.6782	27.2632	0.1951
10	16.7827	38.3781	34.4224	0.1755
15	17.1995	41.5172	34.4064	0.1748
20	17.1011	43.6862	34.5343	0.1747
25	16.9038	44.7145	34.6861	0.1746
30	16.7268	44.9996	34.8023	0.1746
35	16.7404	45.2958	34.8037	0.1746

Table 2 Solution of the PID Controller Problem

Iteration	Z_1	Z_2	Z_3	$Z_4 = f^\circ(Z)$
0	5.0	0.11	.064	35.0
5	5.000032	0.1303	.0246	23.9471
10	4.9765	0.1841	.0524	21.9413
15	4.9146	0.3549	.0578	20.3408
20	4.5008	1.3299	.1714	18.0123
25	4.4059	1.5248	.1928	13.1106
30	4.3026	1.7248	.0412	9.804
35	4.2886	1.7409	.0155	9.2033
41	4.2773	1.7529	.00577	9.1509

Table 3 Solution of the Earthquake Isolation System Problem

REFERENCES

1. Bhatti, M. A., Pister, K. S. and Polak, E., "Optimal Design of an Earthquake Isolation System," *Report No. UCB/EERC- 78/22*, Earthquake Engineering Research Center, University of California, Berkeley, October 1978.

2. Polak, E., Pister, K.S. and Ray, D. "Optimal Design of Framed Structures Subjected to Earthquakes," *Engineering Optimization*, Vol.2, 1976.

3. Polak, E. and Mayne, D. Q., "An Algorithm for Optimization Problems with Functional Inequality Constraints," *IEEE Transactions on Automatic Control*, Vol. AC-21, No. 2, April 1976.

4. Trahan, R. "Optimization Algorithms for Computer-Aided Design Problems", *Ph.D. Dissertation*, Department of Electrical Engineering and Computer Science, U. C. Berkeley, 1978.

5. Gonzaga, G., Polak, E. and Trahan, R., "An Improved Algorithm for Optimization Problems with Functional Inequality Constraints," *Memorandum No. UCB/ERL M78/56*, Electronics Research Laboratory, University of California, Berkeley, September 1977.

6. Blankenship, J.W. and Falk, J.E. "Infinitely Constrained Optimization Problems", *Journal of Optimization Theory and Applications*, Vol.19, No.2, June 1976.

7. Polak, E., **Computational Methods in Optimization,** Academic Press, New York, 1971.

8. Luenberger, D. G., **Introduction to Linear and Nonlinear Programming,** Addison-Wesley, 1973.

9. Armijo, L., "Minimization of Functions Having Continuous Partial Derivatives" *Pacific Jr. Math. 16*, 1-3 (1966).

10. Rosen, J.B. and Suzuki, S., "Construction of Nonlinear Programming Test Problems", *Communications of Association for Computing Machinery*, Vol.8, pp. 113, 1965.

11. Chen, C.T., **Analysis and Synthesis of Linear Control Systems**, Holt, Rinehart and Winston, New York, 1975.

NOTATION

$\mathbb{R}^n$	Denotes the euclidean space of ordered n-tuples of real numbers. When an n-tuplet is a vector in $\mathbb{R}^n$, it is always treated as a column vector.
$<.,.>$	Scalar Product in $\mathbb{R}^n$ defined by $<\mathbf{x},\mathbf{y}> = \sum_{i=1}^{n} x_i\, y_i$.
$\|\|.\|\|_2$	Euclidean norm defined by $\|\|\mathbf{x}\|\|_2 = \sqrt{\mathbf{x}^T\, \mathbf{x}}$.
$\|\|.\|\|_\infty$	Maximum norm in $\mathbb{R}^n$, defined by $\|\|\mathbf{x}\|\|_\infty = \max_{i \in \mathbb{R}^n} \|x_i\|$.
$\mathbf{z}$	Bold letters signify a vector or matrix quantity.
$\mathbf{z}^T$	Transpose of $\mathbf{z}$.
$\mathbf{z}^{-1}$	Inverse of matrix $\mathbf{z}$.
$\|x\|$	Absolute value of x.
$A \bigcup B$	Union of two sets A and B.
$\{x \mid p\}$	Set of points x having property p.
$x \in A$	x belongs to A.
$x \notin A$	x does not belong to A.
(a,b)	Open interval.
[a,b]	Closed interval.
(a,b]	Semi-open or semi-closed interval.
f(.) or f	Denotes a function, with the dot standing for undesignated variable; f(z) denotes the value of f(.) at point z. Domain A and range B of function f(.) is indicated by f: A $\rightarrow$ B.
$\nabla f(\mathbf{z})$	Denotes the gradient of f at $\mathbf{z}$. The gradient is treated as a column vector. If f is a function of more than one variable, the variable with respect to which the gradient is evaluated is shown as a subscript to

the gradient symbol, e.g. $\nabla_{\mathbf{z}} f(\mathbf{z},t)$ indicates gradient with respect to $\mathbf{z}$ of a function of $\mathbf{z}$ and t.

$\overline{T}_{q,\varepsilon}(\mathbf{z}) = \{2,8,9,10,11,12\}$; times at which φ^j is active.

$I^j_{q,\varepsilon,1}(\mathbf{z}) = \{2\}$; 1^{st} active interval.

$I^j_{q,\varepsilon,2}(\mathbf{z}) = \{8,9,10,11,12\}$; 2^{nd} active interval.

$t^j_{q,\varepsilon,1}(\mathbf{z}) = \{2\}$; left local maxima in 1^{st} interval.

$t^j_{q,\varepsilon,2}(\mathbf{z}) = \{10\}$; left local maxima in 2^{nd} interval.

$T^j_{q,\varepsilon}(\mathbf{z}) = \{2,10\}$; ε - active points included in the direction finding process for j^{th} dynamic constraint.

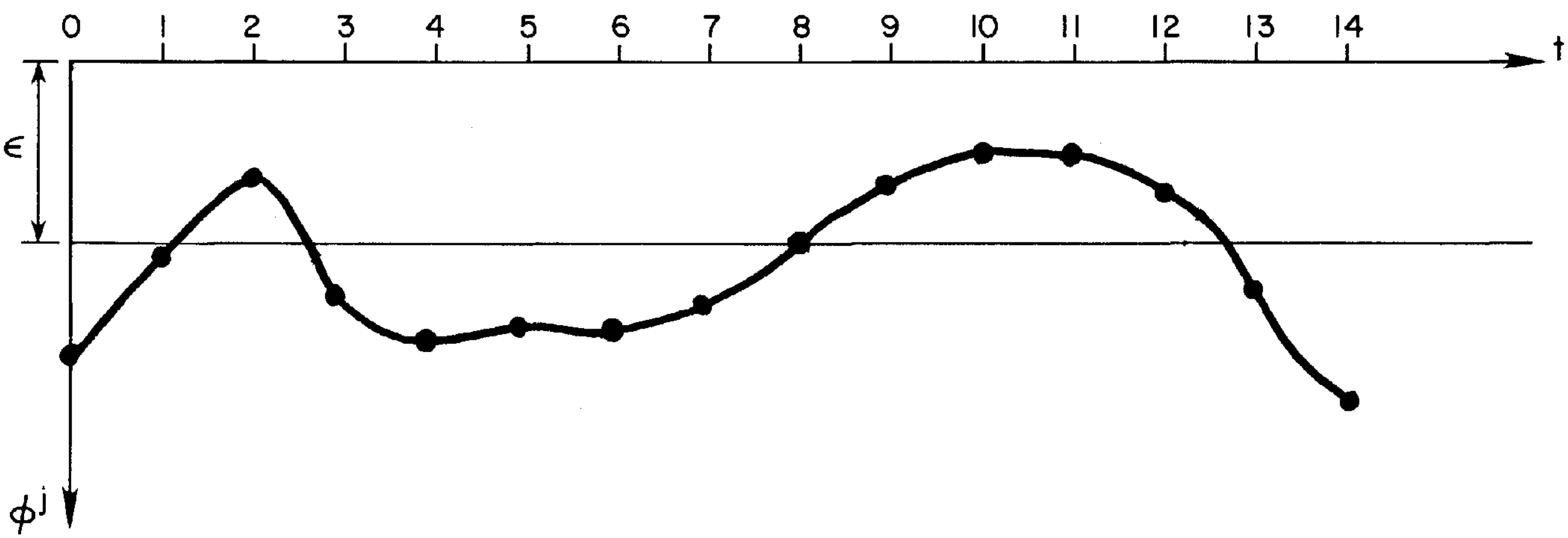


Figure 1 Illustration of ε - Active Points for Dynamic Constraints

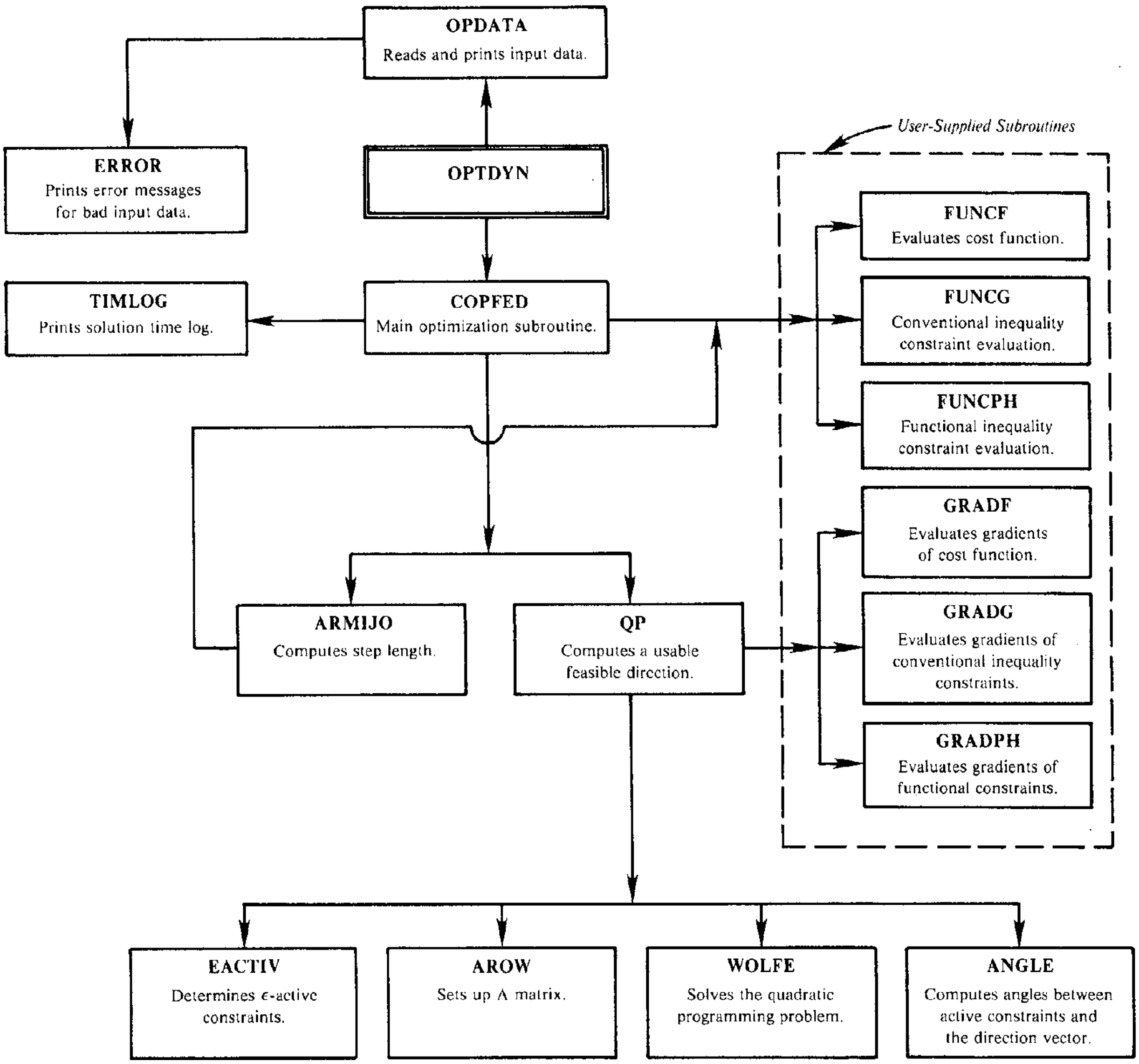


Figure 2 Program Flow Diagram

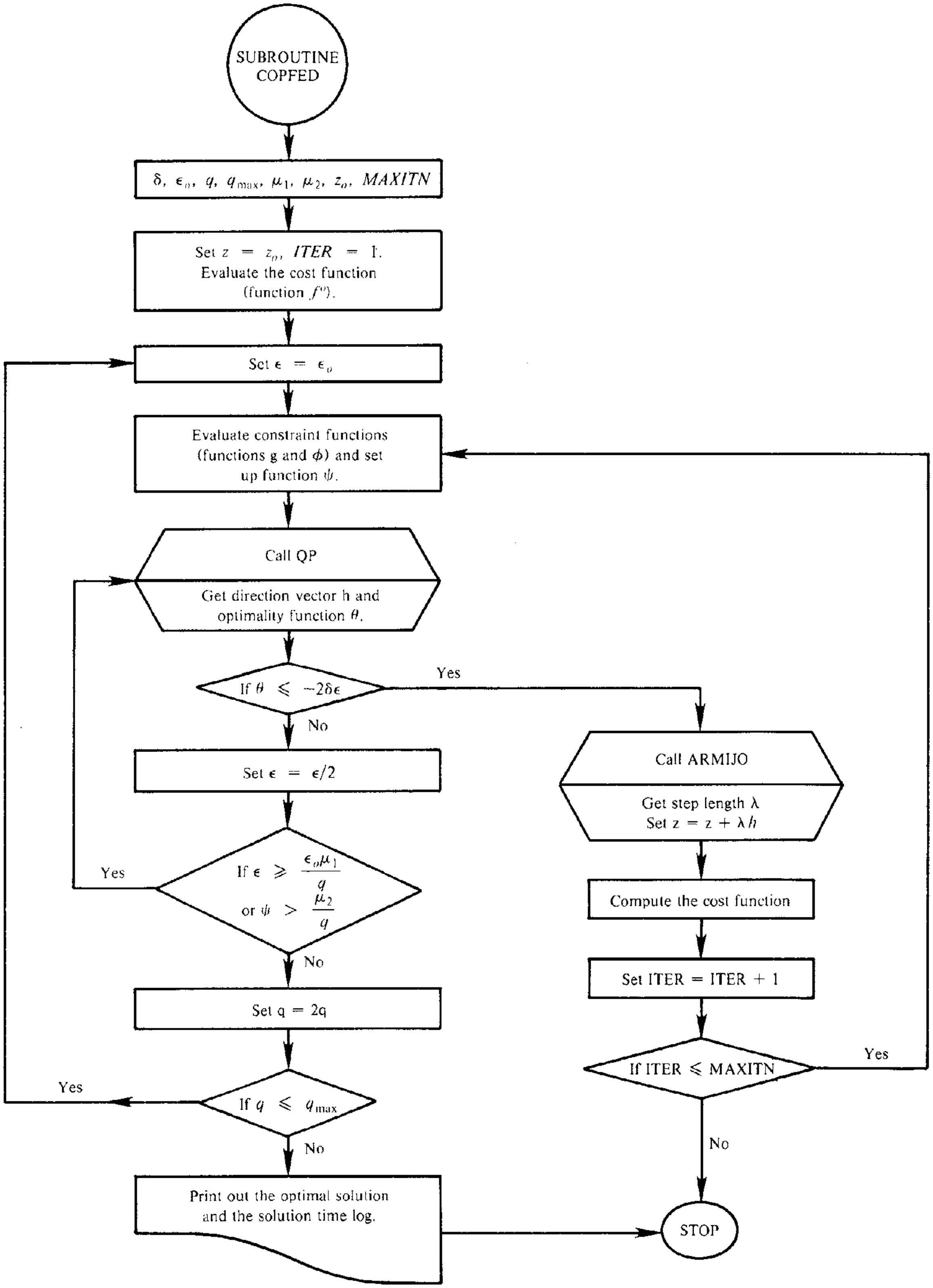


Figure 3 Concise Flow Chart of Subroutine COPFED

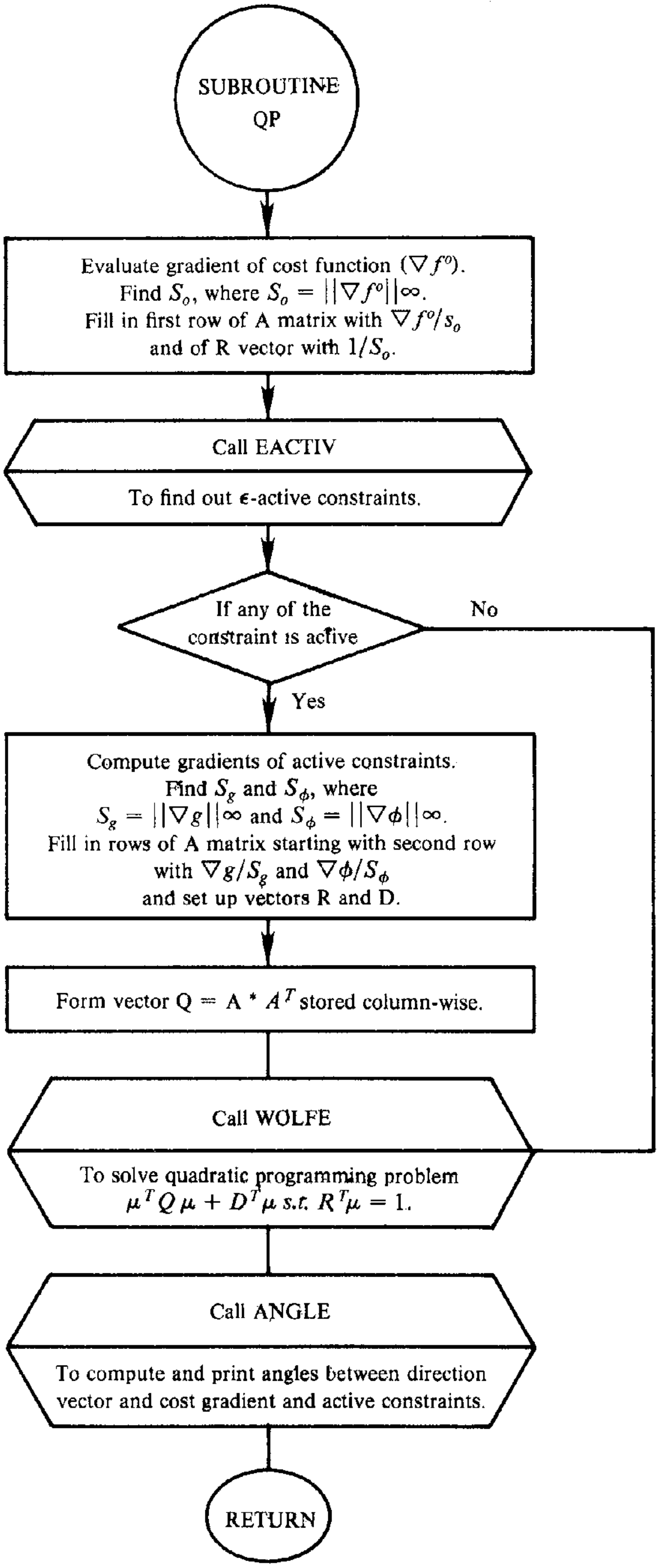


Figure 4 Concise Flow Chart of Subroutine QP

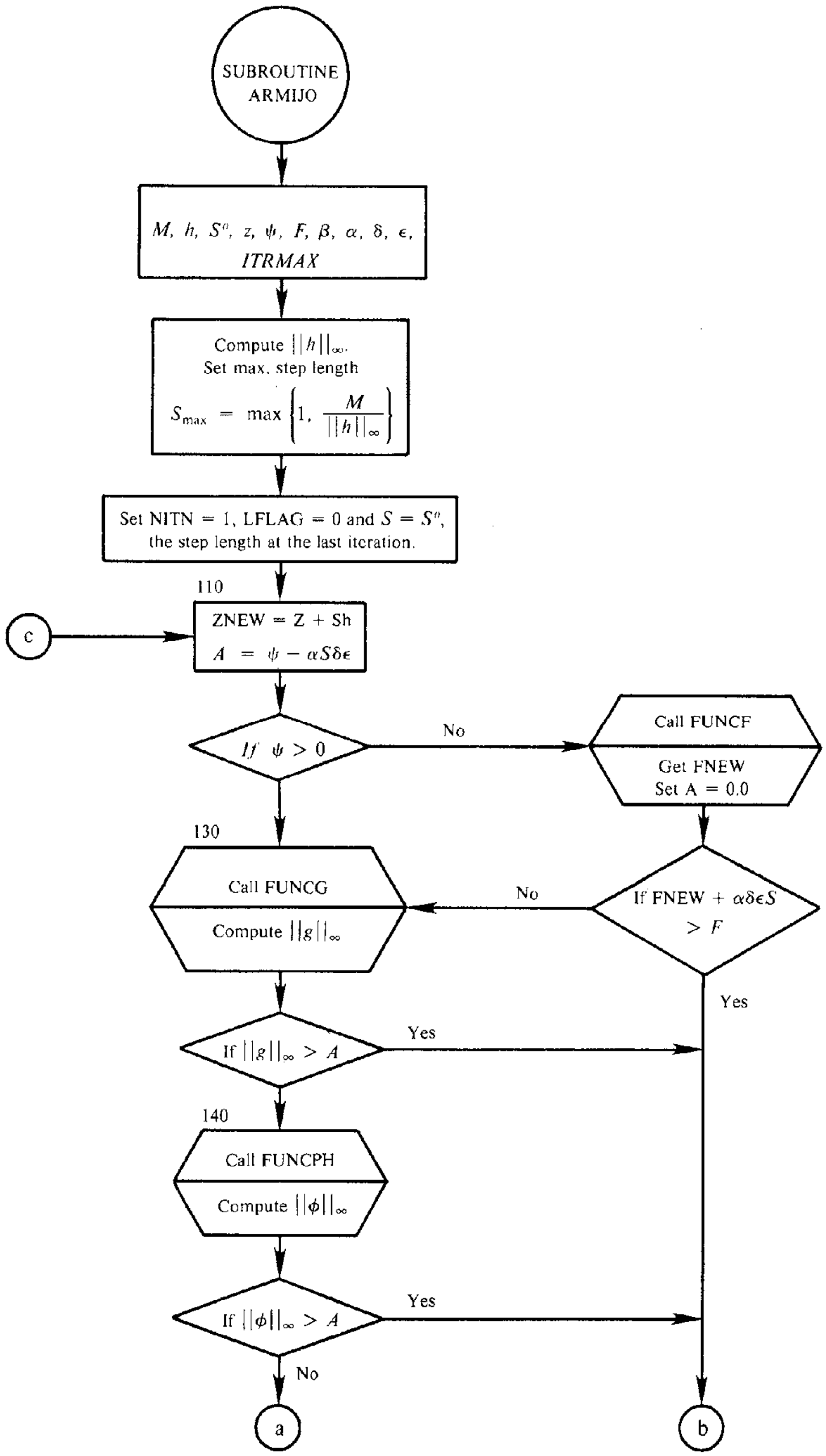


Figure 5 Concise Flow Chart of Subroutine ARMIJO

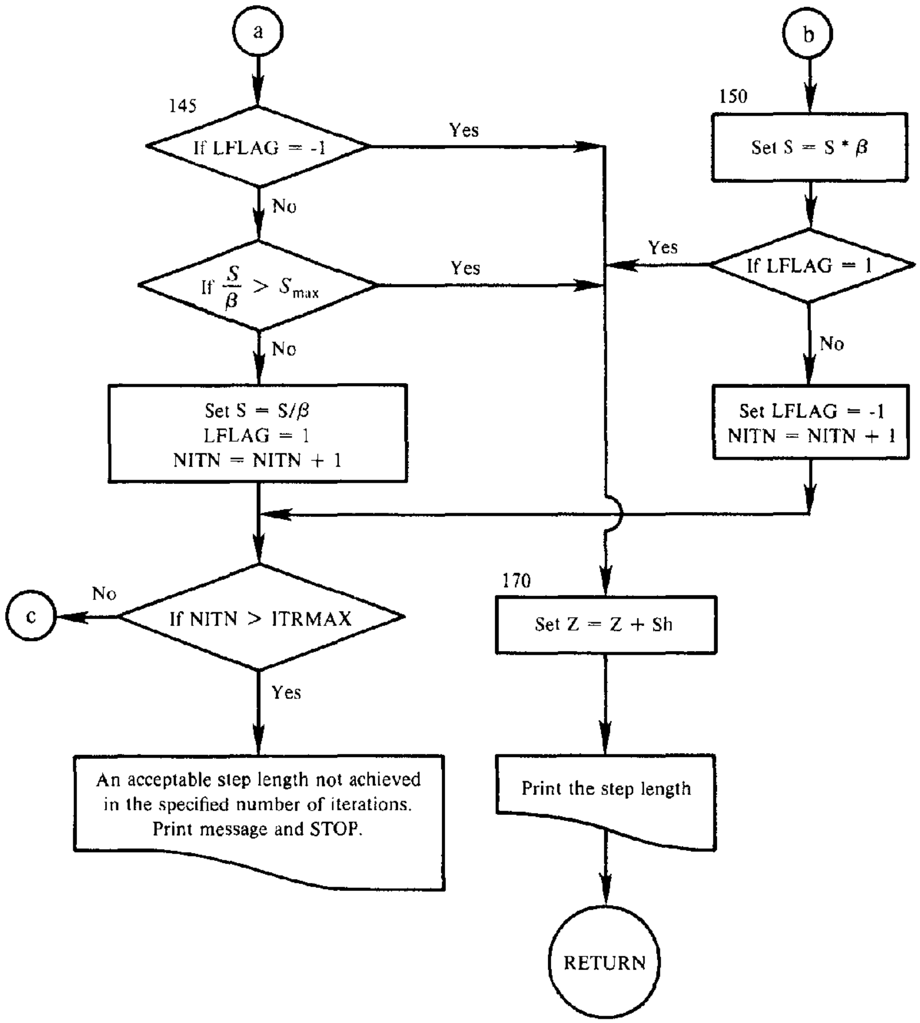
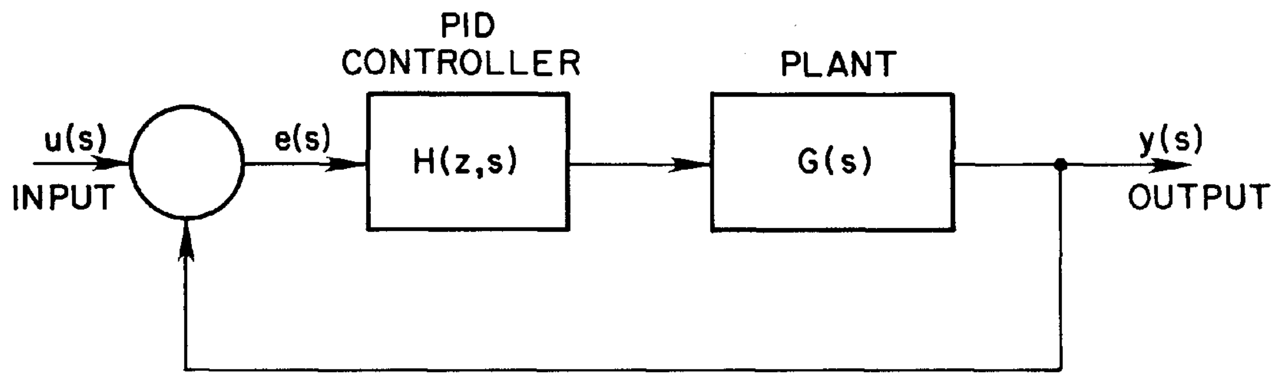


Figure 5 (Continued)



$$H(z,s) = z_1 + z_2/s + z_3\, s$$

$$G(s) = \frac{1}{(s+3)(s^2+2s+2)}$$

Figure 6 Control System to be Optimized

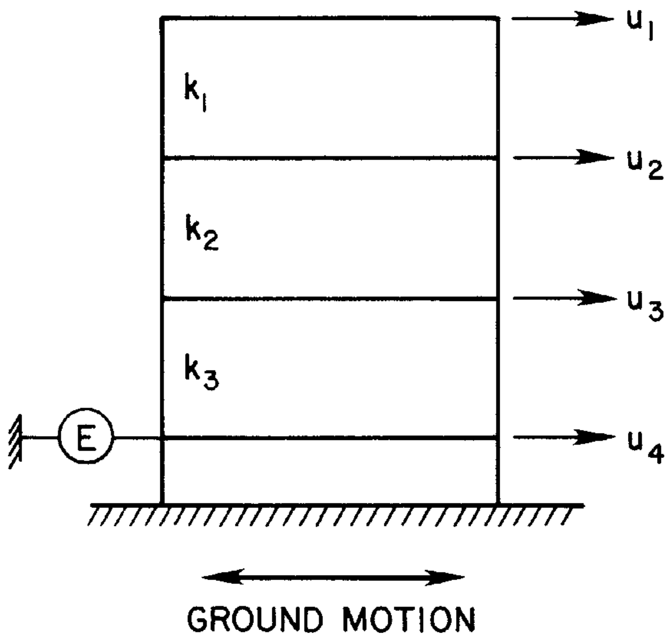


Figure 7 Design of Device "E" for Structural System Shown

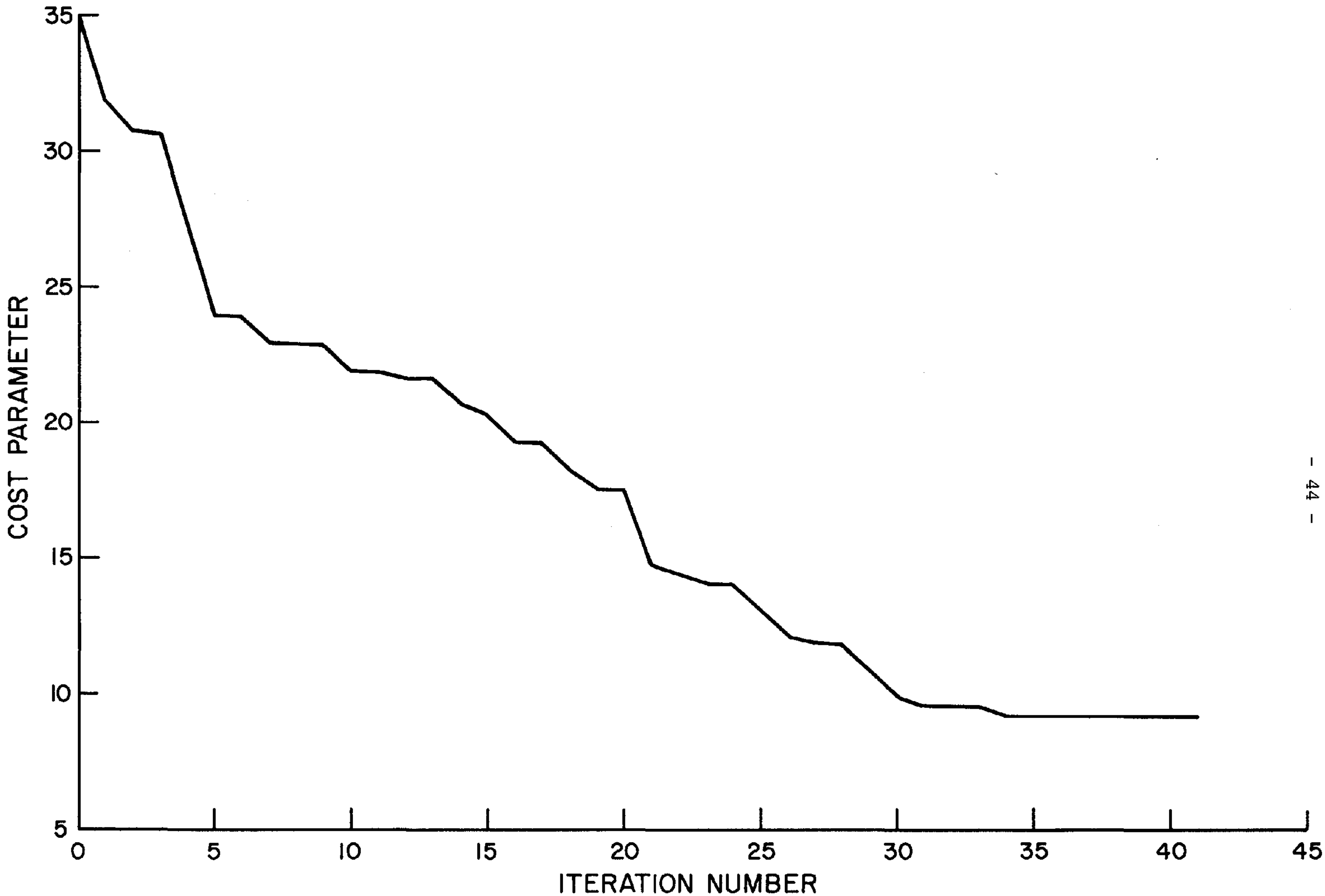


Figure 8 Cost Parameter Versus Number of Iterations

APPENDIX A - OPTDYN User's Guide

The base program requires the following input data.

1. Problem Heading (20 A 4) - one card

COLUMNS	NOTE	VARIABLE	DESCRIPTION OF DATA ENTRY
1-80		HED	Problem heading to be printed with output.

2. Control Information (4 I5) - one card

COLUMNS	NOTE	VARIABLE	DESCRIPTION OF DATA ENTRY
1-5	(1)	MAXITN	Maximum number of iterations allowed.
6-10	(2)	ITER	Iteration number at start of this run. Leave blank if this is the first run.
11-15		NCUT	Maximum number of simplex iterations in solving the quadratic programming problem for direction finding.
16-20		ITRSTP	Maximum number of iterations allowed in step length calculations.

3. Convergence Tolerance Parameters (8 F 10.0) - one card

COLUMNS	NOTE	VARIABLE	DESCRIPTION OF DATA ENTRY
1-10		MU1	Parameter μ_1 used in tolerance test on ε .
11-20		MU2	Parameter μ_2 used in step 4 of the algorithm.
21-30		DELTA	Parameter δ used in step 2 (convergence check) and step 6 (step length calculations).
31-40		E0	ε_0 , initial value of ε .
41-50		GAMMA	Parameter γ , used in QP.

4. Problem size (3 I 5) - one card

COLUMNS	NOTE	VARIABLE	DESCRIPTION OF DATA ENTRY
1-5		JP	Number of conventional inequality constraints (functions 'g').
6-10		JQ	Number of dynamic constraints (functions φ).
11-15		N	Number of optimization variables.

5. Armijo Parameters (8 F 10.0) - one card

COLUMNS	NOTE	VARIABLE	DESCRIPTION OF DATA ENTRY
1-10		STPMAX	Parameter controlling maximum value of step length at any iteration.
11-20		ALPHA	Parameter α .
21-30		BETA	Parameter β.
31-40	(3)	OLDSTP	Initial value for the step length.

6. Functional Constraint Parameters (2 I 5 , 2 F 10.0) - one card

COLUMNS	NOTE	VARIABLE	DESCRIPTION OF DATA ENTRY
1-5		NQ	Initial number of discretization points.
6-10		NQMAX	Maximum number of discretization points.
11-20	(4)	WO	t_0 defining the interval of interest , $[t_0, t_f]$
21-30		WC	t_f defining the interval of interest , $[t_0, t_f]$.

7. Scaling Factors (2 F 10.0) - one card

COLUMNS	NOTE	VARIABLE	DESCRIPTION OF DATA ENTRY
1-10	(5)	SCALE	Scale factor, η , used in scaling QP.
11-20		PUSHF	Scale factor for cost function.

8. Push-off Factors for Conventional Constraints (8 F 10.0)

As many cards as needed to specify push-off factors for all conventional inequality constraint functions

9. Push-off Factors for Dynamic Constraints (8 F 10.0)

As many cards as needed to specify push-off factors for all dynamic constraints.

10. Initial Values of Variables (8 F 10.0)

As many cards as needed to specify initial values for N optimization variables.

NOTES

(1) The program will stop normally if either the number of iterations reaches MAXITN or the optimal solution is achieved.

(2) ITER is used only to label the output. In a number of practical situations it is not possible to let the program run for too many iterations. The process can be restarted with the latest values of the optimization variables, ε and q with ITER equal to the number of the next iteration. the output will then be labeled starting from ITER and incrementing it by one, after each subsequent iteration.

(3) The step length calculations start by assuming an initial trial value equal to OLDSTP. If a good estimate is available, it will accelerate the step length computation process.

(4) If there are no functional constraints, supply a blank card.

(5) The "push-off" factors are used to force the direction vector away from or toward a constraint. Some experience is needed before arriving at suitable values. The angles between the direction vector and objective function gradient and active constraint gradients should be used as guidelines.

APPENDIX B - Listing of the Program

```
*DECK OPTDYN
      PROGRAM OPTDYN (INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT)
C     ****************************************************************
C     A GENERAL PURPOSE OPTIMIZATION PROGRAM FOR PROBLEMS WITH OR
C     WITHOUT DYNAMIC CONSTRAINTS.
C     THE PROGRAM SOLVES PROBLEMS OF THE TYPE
C        MINIMIZE FØ(Z) SUBJECT TO  1. F(Z) = MAX. PHI (Z,T)  .LT. Ø
C                                              (OVER T)
C                                   2. G(Z)                  .LT. Ø
C     SOLUTION ALGORITHM IS GIVEN IN EARTHQUAKE ENGINEERING RESEARCH
C     CENTER,S REPORT UCB/EERC-79/16....JULY 1979.
C
C
C     NOTE ON THE DIMENSIONS OF THE ARRAYS
C     ------------------------------------
C
C     THE MINIMUM REQUIRED DIMENSIONS OF THE ARRAYS ARE:
C         Z(NZ), G(NJP), H(NZ), PHI(NJQ,NQMAX), ZNEW(NZ),
C         GRAD(NZ), NEPTG(NACTIV), NEPTF(NJQ,NACTIV),
C         AQP(NACTIV,NZ), PUSHG(NJP), PUSHPH(NJQ)
C
C      WHERE
C     NZ     = MAX. NO. OF ELEMENTS IN VECTOR Z
C     NJQ    = MAX. NO. OF FUNCTIONAL CONSTRAINTS (FUNCTIONS PHI)
C     NJP    = MAX. NO. OF CONVENTIONAL INEQUALITY CONSTRAINTS.
C     NQMAX = MAX. NO. OF DISCRETIZATION POINTS FOR FUNCTIONAL
C             CONSTRAINTS.
C     NACTIV= MAX. NO. OF ROWS IN THE "A" MATRIX FOR DIRECTION FINDING.
C
C-----THE DIMENSIONS ARE SET FOR
C     NZ = 1Ø , NJQ = 5 , NJP = 1Ø , NQMAX = 1ØØØ , NACTIV = 1Ø
C
C
C     TO CHANGE THE DIMENSION REQUIREMENTS, CHANGE DIMENSIONS
C     OF ARRAYS IN THE MAIN PROGRAM AND IN THE SUBROUTINE QP.
C     THE MINIMUM REQUIRED DIMENSIONS OF ARRAYS USED IN QP ARE GIVEN
C     IN SUBROUTINE QP AND THEY NEED TO BE CHANGED ONLY IF
C     "NACTIV" IS CHANGED.
C
C
C     PROGRAMMED BY.....M. A. BHATTI                MAY 1Ø,1979
C     ****************************************************************
      COMMON /TAPES / NIN,NOU
      COMMON /DIMNSN/ NZ,NJQ,NJP,NQMAX,NACTIV
      COMMON /WORK  / WORK(32)
C
      INTEGER Q,QMAX
      DATA NIN,NOU /1,2/
C
      DIMENSION Z(1Ø),G(1Ø),H(1Ø),PHI(5,1ØØØ),ZNEW(1Ø),GRAD(1Ø),
     1          NEPTG(1Ø),NEPTF(5,1Ø), AQP(1Ø,1Ø), PUSHG(1Ø), PUSHPH(5)
C
      NZ     = 1Ø
      NJQ    = 5
      NJP    = 1Ø
      NQMAX  = 1ØØØ
      NACTIV = 1Ø
C
C-----READ INPUT DATA FOR OPTIMIZATION PART
C
      CALL OPDATA (Z,PUSHF,PUSHG,PUSHPH)
C
C-----CALL MAIN OPTIMIZATION SUBROUTINE
C
  1ØØ CALL COPFED (NJQ,NACTIV,Z,ZNEW,G,H,PHI,GRAD,NEPTG,NEPTF,
     1             AQP,PUSHF,PUSHG,PUSHPH)
C
      END
```

```
*DECK OPDATA
      SUBROUTINE OPDATA (Z,PUSHF,PUSHG,PUSHPH)
C     ****************************************************************
C-----READ AND PRINT DATA FOR OPTIMIZATION PART
C
C     SUBROUTINES NEEDED
C
C        MPRINT
C        ERROR
C
C     OUTPUT VARIABLES
C
C        Z      = VECTOR OF OPTIMIZATION VARIABLES.
C        PUSHF  = PUSH-OFF FACTOR FOR OBJECTIVE FUNCTION.
C        PUSHG  = VECTOR OF PUSH-OFF FACTORS FOR G FUNCTIONS.
C        PUSHPH = VECTOR OF PUSH-OFF FACTORS FOR PHI FUNCTIONS.
C
C     ****************************************************************
C
      COMMON /TAPES / NIN,NOU
      COMMON /DIMNSN/ NZ,NJQ,NJP,NQMAX,NACTIV
      COMMON /OPTDAT/ EØ,MAXITN,NCUT,ITRSTP,ITER,SCALE
      COMMON /ONE   / JP,JQ,N
      COMMON /TWO   / ALPHA,BETA,STPMAX,OLDSTP,ICOUNT
      COMMON /THREE / TOL,TOLER(4),DELTA,MU1,MU2,GAMMA
      COMMON /FIVE  / WØ,WC,Q,DELTAW,QMAX
      COMMON /WORK  / HED(2Ø),WORK(12)
C
      DIMENSION Z(1), PUSHG(1), PUSHPH(1)
      INTEGER Q,QMAX
      REAL MU1,MU2
C
      READ (NIN,1ØØØ) HED
      READ (NIN,1Ø1Ø) MAXITN,ITER,NCUT,ITRSTP
      IF (ITER .LE. Ø) ITER=1
      READ (NIN,1Ø2Ø) MU1    ,MU2    ,DELTA ,EØ     ,GAMMA
      READ (NIN,1Ø1Ø) JP     ,JQ     ,N
      READ (NIN,1Ø2Ø) STPMAX,ALPHA  ,BETA   ,OLDSTP
      READ (NIN,1Ø3Ø) Q      ,QMAX   ,WO     ,WC
      READ (NIN,1Ø2Ø) SCALE  ,PUSHF
C
C-----DIMENSION CHECKS
C
      IF (N .GT. NZ) CALL ERROR(1)
      IF (JQ .GT. NJQ) CALL ERROR(2)
      IF (JP .GT. NJP) CALL ERROR(4)
      IF (Q .GT. NQMAX) CALL ERROR(3)
      IF (QMAX .GT. NQMAX) CALL ERROR(3)
C
      READ (NIN,1Ø2Ø) (PUSHG(I) , I=1,JP)
      READ (NIN,1Ø2Ø) (PUSHPH(I), I=1,JQ)
      READ (NIN,1Ø2Ø) (Z(I) , I=1,N)
C
C-----PRINT OUT DATA JUST READ
C
      WRITE (NOU,2ØØØ) HED
      WRITE (NOU,2Ø1Ø) MAXITN,ITER,NCUT,ITRSTP,
     1                 MU1    ,MU2    ,DELTA ,EØ     ,GAMMA ,
     2                 JP     ,JQ     ,N
      WRITE (NOU,2Ø2Ø) STPMAX,ALPHA  ,BETA   ,OLDSTP,
     4                 WØ     ,WC     ,Q      ,QMAX   ,
     5                 SCALE  ,PUSHF
      CALL MPRINT (PUSHG,1,JP,3ØHPUSH FACTORS FOR G FUNCTIONS          )
      CALL MPRINT (PUSHPH,1,JQ,3ØHPUSH FACTORS FOR PHI FUNCTS.         )
      CALL MPRINT (Z,1,N,3ØH INITIAL VALUES OF PARAMETERS              )
C
      MAXITN = MAXITN + ITER - 1
C
C     SET TOLERANCES
C
      TOL=1.ØE-1Ø
      TOLER(1)=TOL
      TOLER(2)=TOL
```

```
      TOLER(3)=TOL
      TOLER(4)=TOL
      ICOUNT = Ø
C
  5ØØ RETURN
C
 1ØØØ FORMAT (2ØA4)
 1Ø1Ø FORMAT (5I5)
 1Ø2Ø FORMAT (8F1Ø.Ø)
 1Ø3Ø FORMAT (2I5,2F1Ø.Ø)
 2ØØØ FORMAT (1H1/81(1H*)/1X,2ØA4/81(1H*))
 2Ø1Ø FORMAT (//3ØX,32HINPUT DATA FOR OPTIMIZATION PART                 /
     1              3ØX,33H---------------------------------              ///
     2            5X,43HMAXIMUM NO. OF ITERATIONS-----------------=,I5     /
     3            5X,43HITERATION NUMBER AT START OF THIS RUN-----=,I5     /
     4            5X,43HNO. OF SIMPLEX ITERATIONS IN QP-----------=,I5    /
     5            5X,43HMAX.NO.OF ITERATIONS IN STEP LENGTH CALC.-=,I5   //
     6            5X,43HTHE TOLERANCE PARAMETERS ARE--------------         /
     7            5X,43H    MU1---------------------------------=,E1Ø.4 /
     8            5X,43H    MU2---------------------------------=,E1Ø.4 /
     9            5X,43H    DELTA-------------------------------=,E1Ø.4 /
     A            5X,43H    EØ----------------------------------=,E1Ø.4 /
     B            5X,43H    GAMMA-------------------------------=,E1Ø.4//
     C            5X,43HNUMBER OF CONVENTIAL CONSTRAINTS----------=,I5     /
     D            5X,43HNUMBER OF FUNCTIONAL CONSTRAINTS----------=,I5     /
     E            5X,43HDIMENSION OF PARAMETER VECTOR ''Z''-------=,I5    //
     Z    )
 2Ø2Ø FORMAT (
     C            5X,43HTHE ARMIJO PARAMETERS ARE-----------------         /
     D            5X,43H    STEPMAX-----------------------------=,E1Ø.4 /
     E            5X,43H    ALPHA-------------------------------=,E1Ø.4 /

     F            5X,43H    BETA--------------------------------=,E1Ø.4 /
     G            5X,43H    OLDSTEP-----------------------------=,E1Ø.4//
     H            5X,43HFUNCTIONAL CONSTRAINTS PARAMETERS ARE------        /
     I            5X,43H    WØ----------------------------------=,F1Ø.4 /
     J            5X,43H    WC----------------------------------=,F1Ø.4 /
     K            5X,43H    INITIAL NO. OF DISCRETIZATION POINTS--=,I5     /
     L            5X,43H    MAXIMUM NO. OF DISCRETIZATION POINTS--=,I5    //
     M            5X,43HSCALE FACTOR FOR E-ACTIVE CONSTRAINTS-----=,F1Ø.4 /
     N            5X,43HPUSH FACTOR FOR OBJECTIVE FUNCTION--------=,F1Ø.4 /
     Z    )
C
      END
*DECK COPFED
      SUBROUTINE COPFED (NJQ,NACTIV,Z,ZNEW,G,H,PHI,GRAD,NEPTG,NEPTF,
     1                   AQP,PUSHF,PUSHG,PUSHPH)
C     ******************************************************************
C     MAIN SUBROUTINE FOR
C     CONSTRAINED OPTIMIZATION USING FEASIBLE DIRECTIONS METHOD.
C
C     SUBROUTINES NEEDED:
C        FUNCF
C        FUNCG
C        FUNCPH
C        QP
C        ARMIJO
C        TIMLOG
C        MPRINT
C
C     VARIABLES IN THE ARGUMENT LIST HAVE THE FOLLOWING MEANING:
C        NJQ    = ROW DIMENSION OF FUNCTIONAL CONSTRAINT ARRAYS.
C        NACTIV = ROW DIMENSION OF E-ACTIVE ARRAYS.
C        Z      = VECTOR OF OPTIMIZATION VARIABLES.
C        ZNEW   = TEMP. ARRAY USED TO STORE OPTIMIZATION VARIABLES
C                 DURING ARMIJO ITERATIONS.
C        G      = CONVENTIONAL INEQUALITY CONSTRAINT FUNCTIONS (G FUNCTIONS)
C        PHI    = FUNCTIONAL INEQUALITY CONSTARAINTS (FUNCTIONS PHI).
C        GRAD   = ARRAY STORING GRADIENTS OF FUNCTIONS. SAME ARRAY IS USED
C                 REPEATEDLY FOR ALL FUNCTIONS.
C        NEPTG  = ARRAY INDICATING E-ACTIVE G FUNCTINS. IF THE ITH. ENTRY
C                 IS 1 THEN THE ITH. CONSTRAINT IS ACTIVE.
C        NEPTF  = MATRIX INDICATING E-ACTIVE LOCAL MAXIMA FOR FUNCTIONAL
C                 CONSTRAINTS.
```

```
C         AQP     = MATRIX "A" IN THE DIRECTION FINDING PROCESS.
C         PUSHF   = PUSH-OFF FACTOR FOR COST FUNCTION ( FUNCTION F ).
C         PUSHG   = PUSH-OFF FACTOR FOR G FUNCTIONS.
C         PUSHPH  = PUSH-OFF FACTORS FOR PHI FUNCTIONS.
C
C     *****************************************************************
      COMMON /TAPES / NIN,NOU
      COMMON /OPTDAT/ EØ,MAXITN,NCUT,ITRSTP,ITER,SCALE
      COMMON /ONE   / JP,JQ,N
      COMMON /THREE / TOL,TOLER(4),DELTA,MU1,MU2,GAMMA
      COMMON /FIVE  / WØ,WC,Q,DELTAW,QMAX
      COMMON /TIMES / TCONST,TQPT,TARMJT,TTOT
      COMMON /NUMFUN/ NFUNCF,NFUNCG,NFUNCP
      COMMON /NUMGRD/ NGRADF,NGRADG,NGRADP
C
      INTEGER Q,QMAX
      REAL MU1,MU2
      DIMENSION Z(1),G(1),H(1),PHI(NJQ,1),ZNEW(1),GRAD(1),NEPTG(1),
     1          NEPTF(NJQ,1), AQP(NACTIV,1) ,PUSHG(1), PUSHPH(1)
      DATA NFUNCF,NFUNCG,NFUNCP /3*Ø/
      DATA NGRADF,NGRADG,NGRADP /3*Ø/
C
C-----INITIALIZATION.
C
      TCONST = Ø.Ø
      TQPT   = Ø.Ø
      TARMJT = Ø.Ø
      TTOT   = Ø.Ø
C
C-----
C-----START OF THE MAIN ALGORITHM.
C-----
C
      EMU1Q = 1.ØE-5
      AMU2Q = 1.ØE-5
C
C-----FIRST STEP OF THE ALGORITHM
C
  1ØØ E = EØ
      IF (JQ .EQ. Ø) GO TO 11Ø
      EMU1Q = EØ*MU1/FLOAT(Q)
      AMU2Q = MU2/FLOAT(Q)
      DELTAW = (WC-WØ) / FLOAT(Q)
C
  11Ø NFUNCF = NFUNCF + 1
      CALL FUNCF (N,Z,F,NFUNCF)
      WRITE (NOU,2Ø8Ø) F
C
      WRITE (NOU,2Ø3Ø) ITER,E,Q
      CALL SECOND (T1)
C
C     SET UP CONSTRAINTS FUNCTIONS
C
      PSI=Ø.Ø
      IF (JP .EQ. Ø) GO TO 12Ø
      NFUNCG = NFUNCG + 1
      CALL FUNCG (N,JP,Z,G,PSI,NFUNCG)
C
  12Ø IF (JQ .EQ. Ø) GO TO 14Ø
      NFUNCP = NFUNCP + 1
      CALL FUNCPH (N,NJQ,JQ,Z,WØ,WC,DELTAW,Q,PHI,PSI,NFUNCP)
  14Ø CALL SECOND (T2)
  15Ø WRITE (NOU,2Ø4Ø) PSI
C
C-----SECOND STEP OF THE ALGORITHM(DIRECTION FINDING PHASE)
C
      CALL QP (NJQ,NACTIV,G,E,PSI,Z,PHI,THETA,H,SCALE,NCUT,GAMMA,TOL,
     1         TOLER,GRAD,NEPTG,NEPTF, AQP,PUSHF,PUSHG,PUSHPH)
      CALL SECOND(T3)
C
C-----THIRD STEP OF THE ALGORITHM
C
      IF (THETA .LE. (-2.Ø*DELTA*E)) GO TO 17Ø
```

```
C
C-----FOURTH STEP OF THE ALGORITHM
C
      E = E/2.Ø
      WRITE (NOU,211Ø) E
      IF ((E.GE.EMU1Q) .OR. (PSI.GT.AMU2Q)) GO TO 15Ø
C
C-----FIFTH STEP OF THE ALGORITHM(STOP RULE)
C
      IF (JQ .EQ. Ø) GO TO 16Ø
      Q = Q * 2
      IF (Q.LE.QMAX) GO TO 1ØØ
C
  16Ø NFUNCF = NFUNCF + 1
      CALL FUNCF (N, Z, F, NFUNCF)
      WRITE (NOU,21ØØ)
      WRITE (NOU,2Ø5Ø) F
      CALL MPRINT (Z,1,N,3ØHOPTIMAL PARAMETERS                              )
      CALL TIMLOG
C
C-----SIXTH STEP OF THE ALGORITHM (STEP LENGTH CALCULATIONS)
C
  17Ø CALL SECOND (T4)
      CALL ARMIJO (E,PSI,H,Z,ZNEW,ITRSTP,F,DELTA,TOL,PHI,NJQ,G)
      CALL SECOND (T5)
C
      TARMJO = T5 - T4
      TQP    = T3 - T2
      TCONSF = T2 - T1
      TTOTAL = T5 - T1
      TCONST = TCONST + TCONSF
      TQPT   = TQPT   + TQP
      TARMJT = TARMJT + TARMJO
      TTOT   = TTOT   + TTOTAL
C
      WRITE (NOU,2Ø9Ø)
      CALL MPRINT (Z,1,N,3ØHNEW PARAMETERS                                  )
      WRITE (NOU,2Ø7Ø) TTOTAL,TCONSF,TQP,TARMJO
C
      ITER=ITER+1
      IF (ITER .LE. MAXITN) GO TO 11Ø
C
      WRITE (NOU,21ØØ)
      WRITE (NOU,2Ø6Ø)
      CALL TIMLOG
C
C
 2Ø3Ø FORMAT (/1ØØ(1H*)/5X,17HITERATION NUMBER=,I5/28(1H*)//
     1                    5X,9HEPSILON =,E14.6,5X3HQ =,I5)
 2Ø4Ø FORMAT (/5X,4HPSI=,E14.6)
 2Ø5Ø FORMAT (///5X,45HCONGRATULATIONS, HERE IS THE OPTIMAL SOLUTION //
     1             5X,25HOBJECTIVE FUNCTION VALUE=,E14.6)
 2Ø6Ø FORMAT (/5X,52HOPTIMUM NOT ACHIEVED WITHIN THE SPECIFIED NUMBER OF
     1              ,32HITERATIONS--EXECUTION TERMINATED        /)
 2Ø7Ø FORMAT (/5X,46HTOTAL CPU TIME TAKEN IN THIS ITERATION (SEC.)=,
     *            F1Ø.4/
     1            5X,33H(CONSTRAINT FUNCTION EVALUATION =,F1Ø.4/
     2            5X,33H DIRECTION FINDING SUBPROBLEM    =,F1Ø.4/
     3            5X,33H STEP LENGTH CALCULATIONS        =,F1Ø.4,1H))
 2Ø8Ø FORMAT (/5X,26HOBJECTIVE FUNCTION VALUE =,E14.6)
 2Ø9Ø FORMAT (//5X,36HRESULTS AT THE END OF THIS ITERATION/5X,36(1H-))
 21ØØ FORMAT (/1ØØ(1H*))
 211Ø FORMAT (/5X,29HEPSILON IS REDUCED TO ....... E14.6)
      END
```

```
*DECK QP
      SUBROUTINE QP (NJQ,NACTIV,G,E,PSI,Z,PHI,THETA,H,SCALE,NCUT,GAMMA,
     1               TOL,TOLER,GRAD,NEPTG,NEPTF,A,PUSHF,PUSHG,PUSHPH)
C     ******************************************************************
C
C        THIS SUBROUTINE DETERMINES THE EPSILON ACTIVE CONSTRAINTS
C        THEN FILLS IN THE MATRICES ASSOCIATED WITH THE DIRECTION
C        FINDING QP.  THE QP IS SCALED BY NORMALIZING THE MATRIX OF
C        GRADIENTS.  THE OUTPUT QUANTITIES ARE THETA AND H.
C
C        THE SUBROUTINES CALLED BY THIS ONE ARE:
C        1. GRADG
C        2. AROW -- FILLS IN ONE ROW OF THE GRADIENT MATRIX
C        3. GRADPH
C        4. GRADF
C        5. WOLFE -- STANDARD QP SOLVER
C        6. EACTIV
C        7. ANGLE
C
C      VARIABLES IN THE ARGUMENT LIST HAVE THE FOLLOWING MEANING:
C         NJQ    = ROW DIMENSION OF FUNCTIONAL CONSTRAINT ARRAYS.
C         NACTIV = DIMENSION OF E-ACTIVE ARRAYS.
C         G      = ARRAY CONTAINING G CONSTRAINT FUNCTIONS.
C         E      = CURRENT VALUE OF EPSILON.
C         PSI    = FUNCTION PSI.
C         Z      = CURRENT VALUES OF OPTIMIZATION VARIABLES.
C         PHI    = MATRIX OF FUNCTIONAL CONSTRAINTS.
C         THETA  = FUNCTION THETA.
C         H      = DIRECTION VECTOR.
C         SCALE  = SCALE FACTOR FOR ACTIVE CONSTRAINTS ( PARAMETER ETA).
C         NCUT   = MAXIMUM NO. OF ITERATIONS ALLOWED IN QP.
C         GAMMA  = PARAMETER GAMMA.
C         TOL    = TOLERANCE PARAMETER.
C         TOLER  = TOLERANCE PARAMETERS USED IN QP.
C         GRAD   = ARRAY CONTAINING FUNCTION GRADIENTS.
C         NEPTG  = ARRAY CONTAING INFORMATION ON E-ACTIVE G FUNCTIONS.
C                  ITH. ENTRY IS 1 IF THE ITH. CONSTRAINT IS ACTIVE.
C         NEPTF  = MATRIX CONTAINING INFORMATION ON E-ACTIVE PHI FUNCTIONS.
C                  ITH. ROW CONTAINS MESH POINT NUMBERS AT WHICH ITH.
C                  FUNCTIONAL CONSTRAINT IS ACTIVE.
C         A      = MATRIX "A" IN THE DIRECTION FINDING PROCESS.
C         PUSHF  = PUSH-OFF FACTOR FOR COST FUNCTION.
C         PUSHG  = PUSH-OFF FACTORS FOR G FUNCTIONS.
C         PUSHPH = PUSH-OFF FACTORS FOR PHI FUNCTIONS.
C
C     ******************************************************************
C
      COMMON /ONE   / JP,JQ,N
      COMMON /FIVE  / WØ,WC,QQ,DELTAW,QMAX
      COMMON /TAPES / NIN,NOU
      COMMON /NUMGRD/ NGRADF,NGRADG,NGRADP
C
C     THE MINIMUM REQUIRED DIMENSIONS OF THE ARRAYS ARE:
C         S(NACTIV), D(NACTIV), Q(NACTIV*NACTIV) ,R(NACTIV),
C         MUBAR(NACTIV), KOUT(7), AA((NACTIV+2)*(3*NACTIV+2)),
C         B(NACTIV+2), JH(NACTIV+2), X(NACTIV+2), PP(NACTIV+2),
C         YY(NACTIV+2), KB(3*NACTIV+2), EE((NACTIV+2)*(NACTIV+2)),
C         INFIX(8), ERR(8), PRODCT(NACTIV), ATHETA(NACTIV),
C         ENORM(NACTIV)
C
      DIMENSION G(1),Z(1),GRAD(1),S(1Ø),A(NACTIV,1),D(1Ø),PHI(NJQ,1),
     1          Q(1ØØ),R(1Ø),MUBAR(1Ø),
     2          KOUT(7),AA(384),B(12),JH(12),X(12),PP(12),YY(12),
     3          KB(32),EE(144),INFIX(8),H(1),ERR(8),NEPTF(NJQ,1),
     4           NEPTG(1),TOLER(1),PRODCT(1Ø), ATHETA(1Ø), ENORM(1Ø),
     5           PUSHG(1), PUSHPH(1)
C
      REAL MU1,MU2
      REAL MUBAR
      INTEGER QQ,QMAX
      PUSHMX = 5Ø.Ø
C
      WRITE (NOU,2ØØØ)
```

```
      NR     = 1
      SHAT   = Ø.Ø
      DIFF = PSI - E
C
C     COMPUTE THE GRADIENT OF THE COST FUNCTION AND FILL IN THE FIRST
C     ROW OF A MATRIX WITH GRAD F / S(1).
C
      CALL GRADF (N,Z,GRAD)
      NGRADF = NGRADF + 1
      WRITE (NOU,2Ø6Ø) (GRAD(II),II=1,N)
      CALL AROW (S(1),SHAT,GRAD,N,TOL,A,NR,NACTIV)
      R(NR) = PUSHF * (1.Ø / S(NR) - 1.Ø)
      IF ((JP.EQ.Ø) .AND. (JQ.EQ.Ø)) GO TO 15Ø
C
C     DETERMINE E-ACTIVE CONSTRAINTS.
C
      CALL EACTIV (NJQ,NACTIV,NR,G,PHI,DIFF,NEPTF,NEPTG,IACTIV,
     1             NGACTV,Z)
C
C     COMPUTE GRADIENTS OF THE E-ACTIVE CONVENTIONAL CONSTRAINTS AND
C     FILL IN THE NR TH. ROW OF A MATRIX WITH GRAD G / S(NR)
C
      IF (JP .EQ. Ø) GO TO 11Ø
      IF (NGACTV .EQ. Ø) GO TO 11Ø
C
      DO 1ØØ I=1,JP
      IF (NEPTG(I) .EQ. Ø) GO TO 1ØØ
      NR = NR + 1
      CALL GRADG (N,I,Z,GRAD)
      NGRADG = NGRADG + 1
      WRITE (NOU,2Ø2Ø) I,(GRAD(II),II=1,N)
      CALL AROW (S(NR),SHAT,GRAD,N,TOL,A,NR,NACTIV)
      R(NR) = PUSHG(I) + (SCALE*((1.Ø+(G(I)-PSI)/E)**2))
  1ØØ CONTINUE
C     COMPUTE GRADIENTS OF E-ACTIVE FUNCTIONAL CONSTRAINTS AND FILL
C     IN NR TH. ROW OF A MATRIX WITH GRAD PHI / S(NR)
C
  11Ø IF (JQ .EQ. Ø) GO TO 15Ø
      IF (IACTIV .EQ. Ø) GO TO 15Ø
      IGRAD = 1
      DO 14Ø L=1,JQ
      NCC= 1
C
  13Ø NEPTFN = NEPTF(L,NCC)
      IF (NEPTFN .EQ. Ø) GO TO 14Ø
      K = NEPTFN
      CALL GRADPH (N,NJQ,NACTIV,JQ,WØ,WC,DELTAW,QQ,NEPTF,L,Z,K,GRAD,
     1             IGRAD)
      IGRAD = IGRAD + 1
      NGRADP = NGRADP + 1
      NR = NR + 1
      WRITE (NOU,2Ø3Ø) L,K,(GRAD(II),II=1,N)
      CALL AROW (S(NR),SHAT,GRAD,N,TOL,A,NR,NACTIV)
      R(NR) = PUSHPH(L) + SCALE*((1.Ø+(PHI(L,K)-PSI)/E)**2)
      NCC = NCC + 1
      GO TO 13Ø
  14Ø CONTINUE
C
C     SET UP THE QUADRATIC PROGRAMMING PROBLEM AS
C        MIN. (MU'*Q*MU + D'*MU) S.T. R'*MU = C , MU GE. Ø.
C
C     FORM VECTOR Q = A*A'...STORED COLUMN WISE.
C
  15Ø DO 17Ø J=1,NR
      DO 17Ø I=1,NR
      M = I + (J-1)*NR
      Q(M) = Ø.Ø
      DO 16Ø K=1,N
  16Ø Q(M) = Q(M) + A(I,K)*A(J,K)
  17Ø CONTINUE
C
C
```

```
C      FORM VECTOR D.
C
       DO 18Ø I=1,NR
  18Ø  D(I) = Ø.Ø
       D(1) = GAMMA * PSI / S(1)
C
C      FORM VECTOR R.
C
       DO 19Ø I=1,NR
  19Ø  IF (R(I) .GT. PUSHMX) R(I) = PUSHMX
       C = 1.Ø
C
       WRITE (NOU,2Ø5Ø) SHAT
       CALL MPRINT (R,1,NR,3ØHR VECTOR                          )
C
       CALL WOLFE (NR,Q,D,NCUT,TOLER,MUBAR,THETA,SY,KO,KOUT,AA,B,JH,X,
      *PP,YY,KB,EE,INFIX,ERR,R,C)
C
       THETA = -THETA
       WRITE (NOU,2Ø1Ø) THETA,KO,SY
       CALL MPRINT (MUBAR,1,NR,3ØHMUBAR VECTOR                    )
       IF (KO .GT. 1Ø) GO TO 22Ø
       DO 21Ø I=1,N
       H(I)=Ø.Ø
       DO 2ØØ K=1,NR
  2ØØ  H(I) = H(I) - A(K,I)*MUBAR(K)
  21Ø  CONTINUE
C
       CALL MPRINT (H,1,N ,3ØHDIRECTION VECTOR                    )
C
       CALL ANGLE (A,S,NR,N,H,NACTIV,PRODCT, ATHETA, ENORM)
C
       RETURN
C
  22Ø  WRITE (NOU,2Ø4Ø) KO,SY
  6ØØ  CALL TIMLOG
 2ØØØ  FORMAT (/5X,28HDIRECTION FINDING SUBPROBLEM/5X,28(1H-)//)
 2Ø1Ø  FORMAT (/5X,11HQP SOLUTION/5X,6HTHETA=,E14.6,5X,3HKO=,I5,5X,3HSY=,
      1E14.6)
 2Ø2Ø  FORMAT (/5X,8HGRAD. G(,I2,4H) = ,5(E14.6,5X))
 2Ø3Ø  FORMAT (/5X,1ØHGRAD. PHI(,I2,1H,,I5,4H) = ,5(E14.6,5X))
 2Ø4Ø  FORMAT (//5X,21HTHE QP WAS NOT SOLVED/5X,3HKO=,I2,5X,3HSY=,E14.7)
 2Ø5Ø  FORMAT (/5X,7HSHAT = ,E14.6)
 2Ø6Ø  FORMAT (/5X,1ØHGRAD. F =  ,5(E14.6,5X))
       END
*DECK EACTIV
       SUBROUTINE EACTIV (NJQ,NACTIV,NROW,G,PHI,DIFF,NEPTF,NEPTG,IACTIV,
      1                   NGACTV,Z)
C      *****************************************************************
C      SUBROUTINE TO DETERMINE THE CONSTRAINTS WHICH ARE E-ACTIVE.
C      ARGUMENTS
C
C         NJQ     = DIMENSION OF FUNCTIONAL CONSTRAINT ARRAYS.
C         NACTIV  = DIMENSION OF E-ACTIVE ARRAYS.
C         NROW    = NUMBER OF ROWS ALLREADY FILLED IN THE "A" MATRIX.
C         G       = FUNCTIONS G.
C         PHI     = MATRIX OF FUNCTIONS PHI.
C         DIFF    = PSI - EPSILON.
C         NEPTF   = MATRIX CONTAINING INFORMATION ON E-ACTIVE PHI FUNCTIONS.
C                   ITH. ROW CONTAINS MESH POINT NUMBERS AT WHICH ITH.
C                   CONSTAINT IS ACTIVE.
C         NEPTG   = VECTOR CONTAINING INFORMATION ON E-ACTIVE G FUNCTIONS.
C         IACTIV  = A FLAG WITH THE FOLLOWING MEANING:
C                        Ø   IF NONE OF THE CONSTRAINTS ARE ACTIVE;
C                        1   IF ANY OF THE G OR PHI CONSTRAINTS ARE ACTIVE.
C         NGACTV  = NUMBER OF ACTIV G CONSTRAINTS.
C         Z       = VECTOR OF OPTIMIZATION VARIABLES.
C
C      *****************************************************************
C
       COMMON /TAPES / NIN,NOU
       COMMON /ONE   / JP,JQ,NN
       COMMON /FIVE  / WØ,WC,Q,DELTAW,QMAX
C
```

```
      DIMENSION PHI(NJQ,1),NEPTF(NJQ,1),NEPTG(1),G(1),Z(1)
      INTEGER Q,QMAX
C
      NROWS = NROW
      IF (JQ .EQ. Ø) GO TO 2ØØ
C
      DO 1ØØ L=1,JQ
      DO 1ØØ N=1,NACTIV
  1ØØ NEPTF(L,N) = Ø
C
C-----DETERMINE E-ACTIVE FUNCTIONAL CONSTRAINTS (LOCAL MAXIMA'S)
C-----AND SET UP MATRIX NEPTF WHOSE ITH. ROW CONTAINS THE LOCATION
C-----OF E-ACTIVE(LOCAL MAX.) POINT FOR THE ITH. FUNCTIONAL CONSTRAINT.
C
      NQ1 = Q - 1
      IACTIV = Ø
      DO 14Ø L=1,JQ
C
      N = Ø
      K = 1

      PHIK = PHI(L,K)
      PHIKP1 = PHI(L,K+1)
      IF ((PHIK.LT.DIFF) .OR. (PHIK.LT.PHIKP1)) GO TO 11Ø
      N = N + 1
      NEPTF(L,N) = K
      IACTIV = 1
      WRITE (NOU,2ØØØ) L,K,PHIK
C
  11Ø DO 12Ø K = 2,NQ1
      PHIKM1 = PHIK
      PHIK = PHIKP1
      PHIKP1 = PHI(L,K+1)
      IF ((PHIK.LT.DIFF) .OR. (PHIK.LE.PHIKM1) .OR. (PHIK.LT.PHIKP1))
     1 GO TO 12Ø
      N = N + 1
      NEPTF(L,N) = K
      IACTIV = 1
      WRITE (NOU,2ØØØ) L,K,PHIK

  12Ø CONTINUE
C
      PHIKM1 = PHIK
      PHIK = PHIKP1
      IF ((PHIK.LT.DIFF) .OR. (PHIK.LE.PHIKM1)) GO TO 13Ø
      N = N + 1
      NEPTF(L,N) = Q
      IACTIV = 1
      WRITE (NOU,2ØØØ) L,Q,PHIK
  13Ø NROWS = NROWS + N
  14Ø CONTINUE
C
C-----CHECK DIMENSION OF ARRAYS USED IN QP
C
      IF (NROWS .GT. NACTIV) GO TO 25Ø
C
C-----DETERMINE E-ACTIVE CONVENTIONAL CONSTRAINTS.
C
  2ØØ IF (JP .EQ. Ø) GO TO 5ØØ
C
      DO 21Ø I=1,JP
  21Ø NEPTG(I) = Ø
C
      NGACTV = Ø
C
      DO 22Ø I=1,JP
      IF (G(I) .LT. DIFF) GO TO 22Ø
      NGACTV = NGACTV + 1
      NEPTG(I) = 1
  22Ø CONTINUE
C
```

```
C-----DIMENSION CHECK FOR E-ACTIVE POINTS ARRAYS
C
      NROWS = NROWS + NGACTV
      IF (NROWS .GT. NACTIV) GO TO 25Ø
C
  5ØØ RETURN
C
  25Ø WRITE (NOU,2Ø3Ø) NROWS
      CALL TIMLOG
C
C
 2ØØØ FORMAT (/5X,4HPHI(,I4,1H,,I4,2H)=,E14.6)
 2Ø3Ø FORMAT (/5X,47HERROR--DIMENSION OF ARRAYS REQUIRED BY WOLFE IS
     1,9HTOO SHORT/
     25X,33HEITHER INCREASE THE DIMENSION TO        ,I5/
     35X,22HOR,REDUCE EPSILON BAND       )
      END
*DECK AROW
      SUBROUTINE AROW (SL,SHAT,GRAD,N,TOL,A,LL,NACTIV)
C     ****************************************************************
C     *                                                              *
C     *   THIS SUBROUTINE STORES THE SCALED GRADIENT IN THE A MATRIX  *
C     *   AND THE SCALED FUNCTION DIFFERENCE IN THE VECTOR D.         *
C     *                                                              *
C     *     INPUT VARIABLES                                          *
C     *   GRAD  =  GRADIENT TO BE STORED                             *
C     *   N     =  DIMENSION OF Z                                    *
C     *   TOL   =  ZERO TOLERANCE                                    *
C     *   LL    =  ROW INDEX OF A MATRIX AND D TO STORE GRAD AND DIFF *
C     *   NACTIV=  ROW DIMENSION OF "A" MATRIX                       *
C     *     OUTPUT VARIABLES                                         *
C     *   SL    =  INFINITY NORM OF GRAD                             *
C     *   SHAT  =  MAX OVER SL                                       *
C     *   A     =  MATRIX OF GRADIENTS                               *
C     *                                                              *
C     ****************************************************************
      DIMENSION GRAD(1),A(NACTIV,1)
C
      SL=ABS(GRAD(1))
      DO 1ØØ J=2,N
      GRADJ = ABS(GRAD(J))
  1ØØ IF (GRADJ .GT. SL) SL = GRADJ
C
      IF (SL.LT.TOL) SL=1.Ø
      DO 11Ø J=1,N
      GRADSL = GRAD(J) /SL
      IF (ABS(GRADSL) .LT. TOL) GRADSL = Ø.Ø
  11Ø A(LL,J) = GRADSL
      IF (SL.GT.SHAT) SHAT=SL
C
      RETURN
      END
*DECK ANGLE
      SUBROUTINE ANGLE (A,S,NR,N,H,NACTIV, PRODCT, THETA, ENORM)
C
C     ****************************************************************
C
C     THIS SUBROUTINE COMPUTES ANGLE BETWEEN ACTIVE CONSTRAINT GRADIENTS
C     AND THE DIRECTION VECTOR GIVEN BY QP.
C     INPUT VARIABLES:
C         A       = MATRIX OF SCALED GRADIENTS OF COST AND ACTIVE CONSTRAINTS
C                   FIRST ROW OF THIS MATRIX ALWAYS CONTAINS COST GRADIENT.
C         S       = VECTOR CONTAINING SCALING FACTORS BY WHICH THE GRADIENTS
C                   WERE DIVIDED IN MATRIX "A".
C         NR      = NUMBER OF NONZERO ROWS IN A MATRIX.
C         N       = NUMBER OF OPTIMIZATION VARIABLES.
C         H       = DIRECTION VECTOR.
C         NACTIV  = ROW DIMENSION OF A MATRIX.
C     OUTPUT VARIABLES:
C         PRODCT  = ARRAY CONTAINIG INNER PRODUCT OF EACH ROW OF A MATRIX
C                   WITH THE DIRECTION VECTOR.
C         THETA   = VECTOR CONTAINIG ANGLES BETWEEN GRADIENTS AND DIRECTION
C                   VECTOR.
```

```
C         ENORM    = ARRAY CONTAINING ROW NORM OF A MATRIX.
C
C     *****************************************************************
C
      DIMENSION A(NACTIV,1),S(1),H(1), PRODCT(1), THETA(1), ENORM(1)
      COMMON /TAPES / NIN,NOU
C
C-----MULTIPLY ACTIVE CONSTRAINT GRADIENTS BY SCALING FACTOR BY WHICH
C-----THEY WERE DIVIDED WHILE SETTING UP QP
      DO 1ØØ I=1,NR
      DO 1ØØ J=1,N
  1ØØ A(I,J)=S(I)*A(I,J)
C----- COMPUTE NORM OF EACH ROW OF A MATRIX
      DO 11Ø I=1,NR
      ENORM(I)=Ø.Ø
      DO 12Ø J=1,N
  12Ø ENORM(I)=ENORM(I)+A(I,J)*A(I,J)
      ENORM(I)=SQRT(ENORM(I))
  11Ø CONTINUE
C-----COMPUTE NORM OF DIRECTION VECTOR
      HNORM=Ø.Ø
      DO 13Ø I=1,N
  13Ø HNORM=HNORM+H(I)*H(I)
      HNORM=SQRT(HNORM)
C-----MULTIPLY NORM OF EACH ROW OF A MATRIX BY THE NORM OF DIRECTION
C-----VECTOR
      DO 14Ø I=1,NR
      ENORM(I)=ENORM(I)*HNORM
      IF (ENORM(I) .EQ. Ø.Ø ) GO TO 18Ø
  14Ø CONTINUE
C-----COMPUTE INNER PRODUCT OF EACH ROW OF A MATRIX WITH H VECTOR
      DO 15Ø I=1,NR
      PRODCT(I)=Ø.Ø
      DO 16Ø J=1,N
  16Ø PRODCT(I)=PRODCT(I)+A(I,J)*H(J)
  15Ø CONTINUE
C-----DIVIDE THE INNER PRODUCT BY PRODUCT OF NORMS AND TAKE THE
C-----ARC COSINE TO GET THE DESIRED ANGLE
      PI = 4.Ø * ATAN(1.Ø)
      FACT=18Ø.Ø/PI
      DO 17Ø I=1,NR
      FACTOR = PRODCT(I) / ENORM(I)
      SIGN = 1.Ø
      IF (FACTOR .LT. Ø.Ø) SIGN = -1.Ø
      TOL = ABS(FACTOR)
      IF ((TOL.GT.1.Ø) .AND. (TOL.LE.1.ØØØ1)) FACTOR = SIGN
      THETA(I) = ACOS(FACTOR)
      THETA(I)=THETA(I)*FACT
  17Ø CONTINUE
      WRITE (NOU,2ØØØ) THETA(1)
      IF (NR .EQ. 1) GO TO 5ØØ
      WRITE (NOU,2Ø1Ø)
      WRITE (NOU,2Ø2Ø) (THETA(J),J=2,NR)
  5ØØ RETURN
C
  18Ø CALL MPRINT (H,1,N,(3ØH H VECTOR                              ))
      CALL MPRINT (S,1,NR,(3ØH S VECTOR                             ))
      WRITE (NOU,2Ø3Ø)
      CALL TIMLOG
C
 2ØØØ FORMAT (/5X,46HANGLE BETWEEN DIRECTION VECTOR AND COST GRAD.=
     1           ,E14.6)
 2Ø1Ø FORMAT (/5X,47HANGLES BETWEEN DIRECTION VECTOR AND CONSTRAINT
     1          ,4HGRAD)
 2Ø2Ø FORMAT (5X,8F1Ø.2/)
 2Ø3Ø FORMAT (//5X,47HABNORMAL STOP--ROW NORM OF A MATRIX OR NORM OF
     1               38HDIRECTION VECTOR Ø IN SUBROUTINE ANGLE      )
      END
```

```
*DECK ARMIJO
      SUBROUTINE ARMIJO   (E,PSI,H,Z,ZNEW,ITRMAX,F,DELTA,TOL,PHI,NJQ,
     1                     G)
C    *****************************************************************
C
C        THIS SUBROUTINE CALCULATES A STEP LENGTH USING THE
C        ARMIJO TEST.
C    THE VARIABLES IN THE ARGUMENT LIST HAVE THE FOLLOWING MEANING:
C        E        = CURRENT VALUE OF EPSILON.
C        PSI      = FUNCTION PSI.
C        H        = DIRECTION VECTOR.
C        Z        = CURRENT VALUES OF OPTIMIZATION VARIABLES.
C        ZNEW     = INTERMEDIATE VALUES OF OPTIMIZATION VARIABLES DURING
C                   ARMIJO ITERATIONS.
C        ITRMAX   = MAXIMUM NUMBER OF ITERATIONS ALLOWED IN ARMIJO.
C        F        = COST FUNCTION.
C        DELTA    = ARMIJO VARIABLE DELTA.
C        TOL      = TOLERANCE FOR CHANGE IN OPTIMIZATION VARIABLES BETWEEN
C                   ITERATIONS.
C        PHI      = CONSTRAINT FUNCTION PHI.
C        NJQ      = ROW DIMENSION OF ARRAY PHI.
C        G        = G CONSTRAINT FUNCTIONS.
C
C    *****************************************************************
C
      COMMON /ONE   / JP,JQ,N
      COMMON /TWO   / ALPHA,BETA,STPMAX,OLDSTP,ICOUNT
      COMMON /FIVE  / WØ,WC,Q,DELTAW,QMAX
      COMMON /TAPES / NIN,NOU
      COMMON /NUMFUN/ NFUNCF,NFUNCG,NFUNCP
C
      DIMENSION ZNEW(1),Z(1),H(1),G(1),PHI(NJQ,1)
      INTEGER Q,QMAX
C
      WRITE (NOU,2ØØØ)
      NITN = 1
C
C        CALCULATE THE INFINITY NORM OF H
C
      HNORM = ABS(H(1))
C
      DO 1ØØ I=2,N
      HI = ABS(H(I))
      IF (HI .GT. HNORM) HNORM = HI
  1ØØ CONTINUE
      SMAX = STPMAX / HNORM
      SMAX = AMAX1 (1.Ø,SMAX)
      S    = OLDSTP
      LFLAG=Ø
      ALEDT = ALPHA * E * DELTA
C
  11Ø DO 12Ø I=1,N
  12Ø ZNEW(I) = Z(I) + S*H(I)
C
      B = S * ALEDT
      A = PSI - B
C
C     IF PSI GT. Ø , IGNORE COST FUNCTION.
C
      IF (PSI .GT. Ø.Ø) GO TO 13Ø
      A = Ø.Ø
      NFUNCF = NFUNCF + 1
      CALL FUNCF (N,ZNEW,FNEW,NFUNCF)
      IF ((FNEW+B) .GT. F) GO TO 15Ø
C
C
  13Ø IF (JP .EQ. Ø) GO TO 14Ø
C
      GNORM = A
      NFUNCG = NFUNCG + 1
      CALL FUNCG (N,JP,ZNEW,G,GNORM,NFUNCG)
      IF (GNORM .GT. A) GO TO 15Ø
C
```

```
  14Ø IF (JQ .EQ. Ø) GO TO 145
      PHNORM = A
      NFUNCP = NFUNCP + 1
      CALL FUNCPH (N,NJQ,JQ,ZNEW,WØ,WC,DELTAW,Q,PHI,PHNORM,NFUNCP)
      IF (PHNORM .GT. A) GO TO 15Ø
C
  145 IF (LFLAG .EQ. -1) GO TO 16Ø
C
      IF ((S/BETA) .GT. SMAX) GO TO 16Ø
      S=S/BETA
      LFLAG=1
      NITN = NITN + 1
      IF (NITN .GT. ITRMAX) GO TO 18Ø
C
      GO TO 11Ø
C
  15Ø S = S * BETA
C
      IF (LFLAG .EQ. 1) GO TO 16Ø
      LFLAG=-1
      NITN = NITN + 1
      IF (NITN .GT. ITRMAX) GO TO 18Ø
C
      GO TO 11Ø
C
  16Ø IF (S .LT. TOL) S = TOL
      IF (S .GT. SMAX) S = SMAX
C
      DO 17Ø I=1,N
  17Ø Z(I) = Z(I) + S*H(I)
C
      WRITE (NOU,2Ø1Ø) NITN,S
      IF ((S.EQ.TOL) .AND. (OLDSTP.EQ.TOL)) ICOUNT = ICOUNT + 1
      IF (ICOUNT .GE. 1Ø) GO TO 19Ø
      OLDSTP = S
      RETURN
C
  18Ø WRITE (NOU,2Ø2Ø) ITRMAX
      GO TO 55Ø
  19Ø WRITE (NOU,2Ø3Ø) ICOUNT
C
  55Ø CALL TIMLOG
C
 2ØØØ FORMAT (//5X,24HSTEP LENGTH CALCULATIONS/5X, 24(1H-))
 2Ø1Ø FORMAT (/5X,2ØHNO. OF ITERATIONS = I2/
     1          5X,2ØHSTEP LENGTH       = E14.6)
 2Ø2Ø FORMAT (/5X,36HNO. OF ITERATIONS IN ARMIJO EXCEEDS ,I2)
 2Ø3Ø FORMAT (//5X,48HPROGRAM STOP--STEP LENGTH TOO SMALL FOR THE LAST
     1              ,I5,1ØHITERATIONS)
C
      END
*DECK TIMLOG
      SUBROUTINE TIMLOG
C
C     *****************************************************************
C     PRINTS SOLUTION TIME LOG.
C     *****************************************************************
C
      COMMON /TAPES / NIN,NOU
      COMMON /TIMES / TCONST,TQPT,TARMJT,TTOT
      COMMON /NUMFUN/ NFUNCF,NFUNCG,NFUNCP
      COMMON /NUMGRD/ NGRADF,NGRADG,NGRADP
C
      WRITE (NOU,2ØØØ) TCONST,TQPT,TARMJT,TTOT
      WRITE (NOU,2Ø1Ø) NFUNCF,NFUNCG,NFUNCP
      WRITE (NOU,2Ø2Ø) NGRADF,NGRADG,NGRADP
      CALL EXIT
C
 2ØØØ FORMAT (/5X,17HSOLUTION TIME LOG/5X,17(1H-)//
     1  5X,45HTIME SPENT IN CONSTRAINT FUNCTION EVALUATION=,F1Ø.4/
     2  5X,45HTIME SPENT IN DIRECTION FINDING SUBPROBLEM..=,F1Ø.4/
     3  5X,45HTIME SPENT IN STEP LENGTH CALCULATIONS......=,F1Ø.4/
     4  5X,45H          TOTAL TIME SPENT (SECONDS)........=,F1Ø.4)
```

```
 2Ø1Ø FORMAT (//5X,45HNUMBER OF COST FUNCTION EVALUATIONS.........=,I5/
     1  5X,45HNUMBER OF G    FUNCTION EVALUATIONS.........=,I5/
     3  5X,45HNUMBER OF PHI  FUNCTION EVALUATIONS.........=,I5)
 2Ø2Ø FORMAT (//5X,45HNUMBER OF COST GRADIENT EVALUATIONS.........=,I5/
     1  5X,45HNUMBER OF G    GRADIENT EVALUATIONS.........=,I5/
     3  5X,45HNUMBER OF PHI  GRADIENT EVALUATIONS.........=,I5)
C
      END
*DECK ERROR
      SUBROUTINE ERROR(I)
C     *****************************************************************
C     PRINTS ERROR MESSAGES
C     *****************************************************************
      COMMON /TAPES/ NIN,NOU
C
      GO TO (1ØØ,11Ø,12Ø,13Ø) , I
  1ØØ WRITE(NOU,2ØØØ)
      GO TO 5ØØ
  11Ø WRITE(NOU,2Ø1Ø)
      GO TO 5ØØ
  12Ø WRITE(NOU,2Ø2Ø)
      GO TO 5ØØ
  13Ø WRITE (NOU,2Ø3Ø)
C
  5ØØ STOP
 2ØØØ FORMAT (/5X,4ØHERROR--DIMENSION OF ARRAY Z IS TOO SHORT)
 2Ø1Ø FORMAT (/5X,49HERROR--NO. OF FUNCTIONAL CONSTRAINTS EXCEEDS MAX.)
 2Ø2Ø FORMAT (/5X,48HERROR--NO. OF DISCRETIZATION POINTS EXCEEDS MAX.)
 2Ø3Ø FORMAT (/5X,48HERROR--NO.OF INEQUALITY CONSTRAINTS EXCEEDS MAX.)
      END
*DECK MPRINT
      SUBROUTINE MPRINT (A,NRA,NCA,TITLE)
C     *****************************************************************
C     PRINTS MATRICES AND ARRAYS
C     *****************************************************************
      DIMENSION A(NRA,1),TITLE (3)
      COMMON /TAPES / NIN,NOU
      WRITE (NOU,1ØØ) TITLE
      DO 11Ø NC=1,NCA,8
      NCC=NC+7
      IF (NCC.GT.NCA) NCC=NCA
      WRITE (NOU,12Ø) (N,N=NC,NCC)
      DO 13Ø NR=1,NRA
  13Ø WRITE (NOU,14Ø) NR,(A(NR,N),N=NC,NCC)
  11Ø CONTINUE
C
  1ØØ FORMAT(  /5X,3A1Ø)
  12Ø FORMAT(  8X,8I14)
  14Ø FORMAT(I4,4X,8E14.7)
C
      RETURN
      END
```

```
*DECK WOLFE
      SUBROUTINE WOLFE (N,Q,D,NCUT,TOL,Z,PHI,SY,KO,KOUT,A,B,JH,X,P,Y,     WOLF
     * KB,E,INFIX,ERR,R,C)
C                                                                        WOLF
C     WOLFE SOLVES  QUADRATIC MINIMIZATION PROBLEM                       WOLF
C                           PHI = MIN ( ZQZ/2 + DZ )                     WOLF
C                                 SUBJECT TO     Z(I).GE.Ø.Ø  FOR I=1,N  WOLF
C                                         AND    R * Z = C
C     REQUIRED ARRAYS   Q(N*N),D(N),Z(N),A((N+2)*(3*N+2)),B(N+2),        WOLF
C                       JH(N+2),X(N+2),P(N+2),Y(N+2),KB(3*N+2),          WOLF
C                       E((N+2)*(N+2)),R(N)
C     INPUT QUANTITIES    N = DIMENSION OF Z VECTOR                      WOLF
C                         Q = SECOND ORDER COEFFICIENT MATRIX            WOLF
C                                (STORE COLUMNWISE)                      WOLF
C                         D = FIRST ORDER COEFFICIENT VECTOR             WOLF
C                       NCUT= THE MAXIMUM NUMBER OF ITERATIONS           WOLF
C                        TOL= TOLERANCES FOR SIMPLEX ALGORITHM
C     OUTPUT QUANTITIES   Z = SOLUTION VECTOR                            WOLF
C                            (THIS IS NOT THE Z OF THE MAIN PROGRAM)
C                        PHI= MINIMUM FUNCTION VALUE                     WOLF
C                        KO = OUTPUT CONDITION INDICATER FOR SIMPLEX     WOLF
C                            FOR SIMPLEX ALGORITHM                       WOLF
C                              3 - FEASIBLE AND OPTIMAL                  WOLF
C                              4 - NO FEASIBLE SOLUTION                  WOLF
C                              5 - NO PIVOT, INFINITE SOLUTION           WOLF
C                              6 - ITERATION LIMIT ( NCUT ) EXCEEDED     WOLF
C                            FOR WOLFE ALGORITHM                         WOLF
C                             1Ø - INITIALIZATION FOR SIMPLEX FAILED     WOLF
C                             2Ø - SOLN DOES NOT SATISFY OPTIMAL COND'N  WOLF
C                             3Ø - BOTH OF 1Ø AND 2Ø HAPPENED            WOLF
C     IT WAS OUR EXPERIENCE THAT KO .GE. 1Ø  IS CAUSED BY THE IMBALANCE  WOLF
C     OF THE ENTRIES OF MATRIX Q AND VECTOR D                            WOLF
C                        SY = SUM OF ABSOLUTE VALUE OF Y(I)              WOLF
C     THIS SY WILL BE A MEASURE FOR VIOLATION OF OPTIMALITY CONDITION    WOLF
C     NOTE  ACCORDING TO WOLFE  SY = Ø.Ø    IS  OPTIMALITY CONDITON      WOLF
C                                                                        WOLF
C                                                                        WOLF
      DIMENSION Q(1),D(1),Z(1),A(1),B(1),INFIX(8),TOL(4),KOUT(7),        WOLF
     * ERR(8),JH(1),X(1),P(1),Y(1),KB(1),E(1),R(1)                       WOLF
C     SET INTEGERS FOR SIMPLEX ALGORITHM                                 WOLF
      NS=3*N+2                                                           WOLF
      MS=N+2                                                             WOLF
      NMS=NS*MS                                                          WOLF
      NY=2*N+3                                                           WOLF
      INFIX(1)=1                                                         WOLF
      INFIX(2)=NS                                                        WOLF
      INFIX(3)=MS                                                        WOLF
      INFIX(4)=MS                                                        WOLF
      INFIX(5)=2                                                         WOLF
      INFIX(6)=1                                                         WOLF
      INFIX(7)=NCUT                                                      WOLF
      INFIX(8)=Ø                                                         WOLF
C     SET MATRIX A                                                       WOLF
      DO 1Ø I=1,NMS                                                      WOLF
   1Ø A(I)=Ø.Ø                                                           WOLF
      L=1                                                                WOLF
      DO 12 J=1,N                                                        WOLF
      I=(J-1)*MS+2                                                       WOLF
      DO 11 K=1,N                                                        WOLF
      A(I)=Q(L)                                                          WOLF
      I=I+1                                                              WOLF
   11 L=L+1                                                              WOLF
   12 A(I)=R(J)
      I=I+2                                                              WOLF
      DO 13 K=1,N                                                        WOLF
      A(I)=-R(K)
   13 I=I+1                                                              WOLF
      I=I+2                                                              WOLF
      DO 14 K=1,N                                                        WOLF
      A(I)=-1.Ø                                                          WOLF
   14 I=I+MS+1                                                           WOLF
      I=I-MS+2                                                           WOLF
```

```
      DO 15 K=1,N                                                        WOLF
      A(I)=R(K)
   15 I=I+1                                                              WOLF
      I=I+1                                                              WOLF
      DO 16 K=1,N                                                        WOLF
      A(I)=1.0                                                           WOLF
   16 I=I+MS                                                             WOLF
      DO 19 K=1,N                                                        WOLF
      DELTA=-D(K)-Q(K)                                                   WOLF
      I=(2*N+K+1)*MS+K+1                                                 WOLF
      A(I)=SIGN(1.0,DELTA)
   19 CONTINUE                                                           WOLF
C     SET VECTORS                                                        WOLF
      B(1)=0.0                                                           WOLF

      DO 20 K=1,N                                                        WOLF
   20 B(K+1)=-D(K)                                                       WOLF
      B(MS)=C
      PRM = 0.0                                                          WOLF
      DO 21 I=1,MS                                                       WOLF
   21 JH(I)=1                                                            WOLF
      DO 22 J=1,NS                                                       WOLF
   22 KB(J)=0                                                            WOLF
      KB(1)=1                                                            WOLF
      DO 23 J=NY,NS                                                      WOLF
   23 KB(J)=1                                                            WOLF
C     USE SIMPLEX ALGORITHM WITH ADDITIONAL REQUIREMENT                  WOLF
C     CORRECTION IS MADE ONLY IN SUBROUTINE  MIN                         WOLF
      CALL SMPLX (INFIX,A,B,TOL,PRM,KOUT,ERR,JH,X,P,Y,KB,E )             WOLF
      KO = KOUT(1)                                                       WOLF
C     GET SUMY = SUM OF ( ABS(X(KB(J))),J=NY,NS ), WHICH SHOULD BE ZERO WOLF
      SUMY=0.0                                                           WOLF
      DO 25 J=NY,NS                                                      WOLF
      KBJ=KB(J)                                                          WOLF
      IF(KBJ) 25,25,24                                                   WOLF
   24 SUMY=SUMY+ABS (X(KBJ))                                             WOLF
   25 CONTINUE                                                           WOLF
      SY = SUMY                                                          WOLF
C     CHECK IF SUMY = 0.0, WHICH IS OPTIMAL CONDITION FOR WOLFE          WOLF
      IF (ABS(SUMY) .LE. TOL(1)) GO TO 2
    1 KO = KO + 20                                                       WOLF
C     GET Z VECTOR FROM X AND KB                                         WOLF
    2 DO 28 J=1,N                                                        WOLF
      KBJ=KB(J)                                                          WOLF
      IF(KBJ) 26,26,27                                                   WOLF
   26 Z(J)=0.0                                                           WOLF
      GOTO 28                                                            WOLF
   27 Z(J)=X(KBJ)                                                        WOLF
   28 CONTINUE                                                           WOLF
C     GET PHI = MIN VALUE                                                WOLF
      PHI = 0.0                                                          WOLF
      L=0                                                                WOLF
      DO 31 J=1,N                                                        WOLF
      SUM=D(J)                                                           WOLF
      DO 30 I=1,N                                                        WOLF
      L=L+1                                                              WOLF
   30 SUM = SUM + Z(I)*Q(L)*0.5                                          WOLF
   31 PHI = PHI + SUM*Z(J)                                               WOLF
      RETURN                                                             WOLF
      END                                                                WOLF
C
      SUBROUTINE SMPLX  (INFIX,A,B,TOL,PRM,KOUT,ERS,JH,X,P,Y,KB,E)       MSUB2001
CBOSS    MASTER SUBROUTINE OF  RS MSUB,  VERSION 2.                      MSUB2002
C                                                                        MSUB2004
      DIMENSION INFIX(8),A(1),B(1),TOL(4),KOUT(7),ERS(8),JH(1),X(1),     MSUB2005
     1 P(1),Y(1),KB(1),E(1),ZZ(4), IOFIX(16) , TERR(8)                   MSUB2006
C                                                                        MSUB2007
      EQUIVALENCE  (INFLG ,IOFIX(1) ), (N , IOFIX(2) ) ,
     1            (ME,IOFIX(3) ),  (M,IOFIX(4)), (MF,IOFIX(5)),          MSUB2009
     2  (MC, IOFIX(6) ),  ( NCUT, IOFIX(7) ) , ( NVER, IOFIX(8) ),       MSUB2010
     3 ( K, IOFIX(9) ), (ITER, IOFIX(10) ), (INVC , IOFIX(11) )
      EQUIVALENCE (NUMVR, IOFIX(12) ), ( NUMPV, IOFIX(13) ) ,
     4  (INFS, IOFIX(14)  ) , ( JT,  IOFIX(15) ) ,( LA  , IOFIX(16) ),   MSUB2013
     5 (ZZ(1),TPIV), (ZZ(2),TZERO),(ZZ(3),TCOST),(ZZ(4),TECOL)           MSUB2014
C                                                                        MSUB2015
```

```
C                         MOVE INPUTS  ...  ZERO OUTPUTS                   MSUB2Ø16
      DO 134Ø  I= 1, 8                                                     MSUB2Ø17
       TERR(I) = Ø.Ø                                                       MSUB2Ø18
      IOFIX(I+8)  =  Ø                                                     MSUB2Ø19
 134Ø  IOFIX(I)  =  INFIX(I)                                               MSUB2Ø2Ø
      DO 13Ø8  I = 1 , 4                                                   MSUB2Ø21
 13Ø8 ZZ(I)  =  TOL(I)                                                     MSUB2Ø22
      PMIX = PRM                                                           MSUB2Ø23
      TCOST  =  -  ABS   (TCOST)
      M2 = M**2                                                            MSUB2Ø25
      INFS = 1                                                             MSUB2Ø26
      LA   =   Ø                                                           MSUB2Ø27
C                    CHECK FOR ILLEGAL INPUT                               MSUB2Ø28
      IF (N)  13Ø4, 13Ø4, 1371                                             MSUB2Ø29
 1371  IF (M - MF ) 13Ø4, 13Ø4, 1372                                       MSUB2Ø3Ø
 1372  IF (MF - MC)  13Ø4, 13Ø4, 1373                                      MSUB2Ø31
 1373  IF ( MC ) 13Ø4 , 13Ø4,  1374                                        MSUB2Ø32
 1374  IF (ME - M ) 13Ø4, 1375, 1375                                       MSUB2Ø33
 13Ø4 K  =  7                                                              MSUB2Ø34
      GO TO 1392                                                           MSUB2Ø35
 1375 IF(INFLG-(INFLG/4)* 4   -1 )  14ØØ,  132Ø,  1ØØ                      MSUB2Ø36
 14ØØ  CALL       NEW (M,N, JH, KB, A, B, MF, ME )                         MSUB2Ø37
 132Ø  CALL       VER ( A, B, JH, X, E, KB, Y, N, ME, M, MF, INVC,         MSUB2Ø38
     1                   NUMVR, NUMPV, INFS, LA, TPIV, TECOL, M2 )         MSUB2Ø39
C                           PERFORM ONE ITERATION                          MSUB2Ø4Ø
  1ØØ  CALL       XCK ( M, MF, JH, X, TZERO, JIN )                         MSUB2Ø41
C                                                                          MSUB2Ø42
C          CHECK CHANGE OF PHASE.. GO BACK TO INVERT IF GONE INFEAS.       MSUB2Ø43
      IF (INFS - JIN ) 132Ø,  5ØØ,  2ØØ                                    MSUB2Ø44
C                 BECOME FEASIBLE                                          MSUB2Ø45
  2ØØ INFS = Ø                                                             MSUB2Ø46
  2Ø1 PMIX  =  Ø.Ø                                                         MSUB2Ø47
  5ØØ  CALL       GET ( M, MC, MF, JH, X, P, E, INFS, PMIX )               MSUB2Ø48
       CALL       MIN ( JT, N, M, A, P, KB, ME, TCOST )                    MSUB2Ø49
      JM  =  JT                                                            MSUB2Ø5Ø
      J = JM                                                               MSUB2Ø51
        IF  (JM)    2Ø3, 2Ø3, 222                                          MSUB2Ø52
C          ALL COSTS NON-NEGATIVE...  K = 3 OR 4                           MSUB2Ø53
  2Ø3 K = 3 + INFS                                                         MSUB2Ø54
      GO TO 257                                                            MSUB2Ø55
C                           NORMAL CYCLE                                   MSUB2Ø56
  222  CALL       JMY ( J, A, E, M, Y, ME )                                MSUB2Ø57
       CALL        ROW  ( IR, M, MF, JH, X, Y, TPIV )                      MSUB2Ø58
C                           TEST PIVOT                                     MSUB2Ø59
  2Ø6   IF( IR )   2Ø7, 2Ø7, 21Ø                                           MSUB2Ø6Ø
C                           NO PIVOT                                       MSUB2Ø61
  2Ø7 K = 5                                                                MSUB2Ø62
  257 IF (PMIX)   2Ø1, 4ØØ, 2Ø1                                            MSUB2Ø63
C                     ITERATION LIMIT FOR CUT OFF                          MSUB2Ø64
  21Ø IF (ITER -NCUT )    2Ø8,  16Ø,  16Ø                                  MSUB2Ø65
C                           PIVOT FOUND                                    MSUB2Ø66
  2Ø8  CALL       PIV  ( IR, Y, M, E, X, NUMPV, TECOL )                    MSUB2Ø67
  221 JOLD  = JH(IR)                                                       MSUB2Ø68
      IF (JOLD) 213, 213, 214                                              MSUB2Ø69
  214 KB(JOLD) = Ø                                                         MSUB2Ø7Ø
  213 KB(JM)   = IR                                                        MSUB2Ø71
      JH(IR)   = JM                                                        MSUB2Ø72
      LA  =  Ø                                                             MSUB2Ø73
      ITER   = ITER   +1                                                   MSUB2Ø74
      INVC   = INVC   +1                                                   MSUB2Ø75
C                     INVERSION FREQUENCY                                  MSUB2Ø76
      IF (INVC - NVER  )  1ØØ, 132Ø, 1ØØ                                   MSUB2Ø77
C                                 CUT OFF ... TOO MANY ITERATIONS          MSUB2Ø78
  16Ø K = 6                                                                MSUB2Ø79
  4ØØ  CALL       ERR ( M, A, B, TERR, JH, X, P, Y, ME, LA )               MSUB2Ø8Ø
      IF (LA)  193, 191,  193                                              MSUB2Ø81
  191 LA  =  4                                                             MSUB2Ø82
      IF (INFLG   -  4  )  132Ø,  193,  193
  193 IF  (K-5)    1392,  194,  1392                                       MSUB2Ø84
  194  CALL       JMY ( J, A, E, M, Y, ME )                                MSUB2Ø85
C                 SET EXIT VALUES                                          MSUB2Ø86
 1392   DO  13Ø9  I=  1, 8                                                 MSUB2Ø87
 13Ø9 ERS(I)         =   TERR(I)                                           MSUB2Ø88
```

```
       DO 1329  I = 1, 7                                                MSUB2Ø89
 1329 KOUT(I)  =  IOFIX(I+8)                                            MSUB2Ø9Ø
      RETURN                                                            MSUB2Ø91
C                                                                       MSUB2Ø92
      END                                                               MSUB2Ø93
C
      SUBROUTINE DEL (  JM, DT, M, A, P, ME )                           MSUB2Ø96
CDELS             DELTA-JAV.  PRICES OUT ONE MATRIX COLUMN              MSUB2Ø95
      DIMENSION A(1), P(1)                                              MSUB2Ø97
C                                                                       MSUB2Ø98
  3ØØ DT = Ø.                                                           MSUB2Ø99
      KDEL = (JM - 1) * ME                                              MSUB21ØØ
C                                                                       MSUB21Ø1
  3Ø1 DO  3Ø3  IDEL = 1, M                                              MSUB21Ø2
      KDEL = KDEL + 1                                                   MSUB21Ø3
      IF ( A(KDEL))3Ø4, 3Ø3, 3Ø4                                        MSUB21Ø4
  3Ø4 IF ( P(IDEL) )  3Ø2, 3Ø3, 3Ø2                                     MSUB21Ø5
  3Ø2 DT = DT + P(IDEL) * A(KDEL)                                       MSUB21Ø6
  3Ø3 CONTINUE                                                          MSUB21Ø7
C                                                                       MSUB21Ø8
  399 RETURN                                                            MSUB21Ø9
      END                                                               MSUB211Ø
C
      SUBROUTINE ERR ( M, A, B, TERR, JH, X, P, Y, ME, LA )             MSUB2113
CERRS             ERROR CHECK.  COMPARES AX WITH B, PA WITH ZERO        MSUB2112
      DIMENSION JH(1), A(1), B(1), X(1), P(1), Y(1), TERR(8)            MSUB2114
C                                STORE AX-B AT Y                        MSUB2115
      DO 4Ø1 I = 1, M                                                   MSUB2116
  4Ø1 Y(I) =-B(I)                                                       MSUB2117
      DO 4Ø2 I = 1, M                                                   MSUB2118
      JA = JH(I)                                                        MSUB2119
      IF (JA) 4Ø3, 4Ø2, 4Ø3                                             MSUB212Ø
  4Ø3 IA =ME* (JA-1)                                                    MSUB2121
      DO 4Ø5 IT = 1, M                                                  MSUB2122
      IA = IA + 1                                                       MSUB2123
      IF(A(IA) )  415, 4Ø5, 415                                         MSUB2124
  415 Y(IT) =Y(IT) +X(I) * A(IA)                                        MSUB2125
  4Ø5 CONTINUE                                                          MSUB2126
  4Ø2 CONTINUE                                                          MSUB2127
C                                FIND SUM AND MAXIMUM OF ERRORS         MSUB2128
      DO 481  I = 1, M                                                  MSUB2129
      YI  = Y(I)                                                        MSUB213Ø
      IF ( JH(I) )  472, 471, 472                                       MSUB2131
  471  YI = YI + X(I)                                                   MSUB2132
  472 TERR(LA+1) = TERR(LA+1) + ABS (YI)
      IF  ( ABS  (TERR(LA+2))- ABS  (   YI   )  )   482, 481, 481
  482 TERR(LA+2) = YI                                                   MSUB2135
  481   CONTINUE                                                        MSUB2136
C                                STORE P TIMES BASIS AT DT              MSUB2137
      DO 411  I = 1, M                                                  MSUB2138
      JM = JH(I)                                                        MSUB2139
      IF ( JM     )  3ØØ , 411 , 3ØØ                                    MSUB214Ø
  3ØØ  CALL       DEL (  JM, DT, M, A, P, ME )                          MSUB2141
  41Ø TERR(LA+3)  =   TERR(LA +3)  +  ABS (DT)
      IF (ABS (TERR(LA+4))   -   ABS (DT)  )  413, 411, 411
  413  TERR(LA+4)  =  DT                                                MSUB2144
  411 CONTINUE                                                          MSUB2145
      RETURN                                                            MSUB2146
      END                                                               MSUB2147
C
      SUBROUTINE GET ( M, MC, MF, JH, X, P, E, INFS, PMIX )             MSUB2151
CGETS       GET PRICES                                                  MSUB215Ø
      DIMENSION JH(1), X(1), P(1), E(1)                                 MSUB2152
C                                                                       MSUB2153
  5ØØ MMM = MC                                                          MSUB2154
C                          PRIMAL PRICES                                MSUB2155
  5Ø2 DO 5Ø3 J = 1, M                                                   MSUB2156
      P(J) = E(MMM)                                                     MSUB2157
  5Ø3 MMM =MMM + M                                                      MSUB2158
      IF ( INFS )  5Ø1, 599, 5Ø1                                        MSUB2159
C                          COMPOSITE PRICES                             MSUB216Ø
  5Ø1 DO 5Ø4  J = 1, M                                                  MSUB2161
  5Ø4 P(J) = P(J)* PMIX                                                 MSUB2162
```

```
      DO 5Ø5  I = MF, M                                                 MSUB2163
      MMM =I                                                            MSUB2164
      IF  ( X(I) )  5Ø6, 5Ø7, 5Ø7                                       MSUB2165
  5Ø6 DO 5Ø8  J = 1, M                                                  MSUB2166
      P(J) =  P(J) + E(MMM)                                             MSUB2167
  5Ø8 MMM = MMM + M                                                     MSUB2168
      GO TO 5Ø5                                                         MSUB2169
  5Ø7 IF (JH(I))  5Ø5, 5Ø9, 5Ø5                                         MSUB217Ø
  5Ø9 DO 51Ø  J = 1, M                                                  MSUB2171
      P(J) = P(J) - E(MMM)                                              MSUB2172
  51Ø MMM =  MMM +M                                                     MSUB2173
  5Ø5   CONTINUE                                                        MSUB2174
C                                                                       MSUB2175
  599 RETURN                                                            MSUB2176
      END                                                               MSUB2177
C
      SUBROUTINE JMY (JT, A, E, M, Y, ME )                              MSUB218Ø
CJMYS            J MULTIPLY.  BASIS INVERSE * COLUMN J                  MSUB2179
      DIMENSION A(1), E(1), Y(1)                                        MSUB2181
C                                                                       MSUB2182
  6ØØ DO 61Ø  I= 1,M                                                    MSUB2183
  61Ø Y(I) =Ø.                                                          MSUB2184
      LP   =  JT*ME - ME                                                MSUB2185
      LL = Ø                                                            MSUB2186
      DO 6Ø5   I= 1,M                                                   MSUB2187
      LP = LP + 1                                                       MSUB2188
      IF   (A(LP))   6Ø1, 6Ø2, 6Ø1                                      MSUB2189
  6Ø1 DO 6Ø6  J  =  1,M                                                 MSUB219Ø
      LL = LL + 1                                                       MSUB2191
  6Ø6 Y(J) = Y(J) + A(LP) * E(LL)                                       MSUB2192
      GO TO 6Ø5                                                         MSUB2193
  6Ø2 LL = LL + M                                                       MSUB2194
  6Ø5 CONTINUE                                                          MSUB2195
  699 RETURN                                                            MSUB2196
      END                                                               MSUB2197
      SUBROUTINE MIN ( JT, N, M, A, P, KB, ME, TCOST )                  MSUB22ØØ
C     THIS SUBROUTINE IS FOR THE MODIFIED (QP) PROGRAM
CMINS            MIN D-J.  SELECTS COLUMN TO ENTER BASIS                MSUB2199
      DIMENSION A(1), P(1), KB(1)                                       MSUB22Ø1
C                                                                       MSUB22Ø2
  7ØØ JT = Ø                                                            MSUB22Ø3
      DA = TCOST                                                        MSUB22Ø4
C                                                                       MSUB22Ø5
  7Ø1 DO  7Ø2  JM = 1, N                                                MSUB22Ø6
C                            SKIP COLUMNS IN BASIS                      MSUB22Ø7
  7Ø3 IF ( KB(JM) )  7Ø2, 3ØØ, 7Ø2                                      MSUB22Ø8
  3ØØ  CALL      DEL (  JM, DT, M, A, P, ME )                           MSUB22Ø9
  7Ø5 IF  ( DT - DA )  7Ø8, 7Ø2, 7Ø2                                    MSUB221Ø
C     CHECK IF JM VIOLATES ADDITIONAL REQUIREMENT OF USABILITY,         WOLF
C               I.E., ZE(I+N+1)*ZE(I) = Ø.Ø   FOR I=1,(N+1)             WOLF
  7Ø8 NP=M-1                                                            WOLF
      IF(JM-NP) 71Ø,71Ø,712                                             WOLF
  71Ø JMNP=JM+NP                                                        WOLF
      GOTO 714                                                          WOLF
  712 JMNP=JM-NP                                                        WOLF
  714 IF(KB(JMNP)) 7Ø2,716,7Ø2                                          WOLF
C     JM MAYBE ADMITTED                                                 WOLF
  716 DA=DT                                                             WOLF
      JT=JM                                                             WOLF
C     END OF CORRECTION FOR WOLFE                                       WOLF
  7Ø2    CONTINUE                                                       MSUB2213
      RETURN                                                            MSUB2214
      END                                                               MSUB2215
      SUBROUTINE NEW (M,N, JH, KB, A, B, MF, ME )                       MSUB2218
CNEWS             STARTS PHASE ONE                                      MSUB2217
      DIMENSION JH(1), KB(1), A(1), B(1)                                MSUB2219
C                            INITIATE                                   MSUB222Ø
 14ØØ   DO 14Ø1 I = 1, M                                                MSUB2221
 14Ø1 JH(I) = Ø                                                         MSUB2222
C                 INSTALL SINGLETONS                                    MSUB2223
       KT   =   Ø                                                       MSUB2224
        DO 14Ø2   J = 1, N                                              MSUB2225
      KB(J) = Ø                                                         MSUB2226
      KTA =   KT + MF                                                   MSUB2227
      KTB =   KT + M                                                    MSUB2228
```

```
C                                    TALLY ENTRIES IN CONSTRAINTS          MSUB2229
      KQ  = Ø                                                              MSUB223Ø
      DO  14Ø3 L = KTA,KTB                                                 MSUB2231
      IF (A(L)) 14Ø4, 14Ø3, 14Ø4                                           MSUB2232
 14Ø4 KQ = KQ+1                                                            MSUB2233
      LQ = L                                                               MSUB2234
 14Ø3   CONTINUE                                                           MSUB2235
C                                    CHECK WHETHER J IS CANDIDATE          MSUB2236
            IF (KQ - 1) 14Ø2, 14Ø5, 14Ø2                                   MSUB2237
 14Ø5 IQ = LQ- KT                                                          MSUB2238
          IF ( JH(IQ) ) 14Ø2, 14Ø6, 14Ø2                                   MSUB2239
 14Ø6 IF (A(LQ)*B(IQ)) 14Ø2, 14Ø7, 14Ø7                                    MSUB224Ø
C                                    J IS CANDIDATE. INSTALL               MSUB2241
 14Ø7 JH(IQ) = J                                                           MSUB2242
      KB(J)  = IQ                                                          MSUB2243
 14Ø2 KT = KT + ME                                                         MSUB2244
      RETURN                                                               MSUB2245
      END                                                                  MSUB2246
C
      SUBROUTINE PIV  ( IR, Y, M, E, X, NUMPV, TECOL )                     MSUB2249
CPIVS           PIVOT.  PIVOTS ON GIVEN ROW                                MSUB2248
      DIMENSION Y(1), E(1), X(1)                                           MSUB225Ø
C                              LEAVE TRANSFORMED COLUMN IN Y(I)            MSUB2251
C                                                                          MSUB2252
  9ØØ NUMPV  =  NUMPV  +  1                                                MSUB2253
C                                                                          MSUB2254
      T2 = -Y(IR)                                                          MSUB2255
      Y(IR) = -1.                                                          MSUB2256
      LL = Ø                                                               MSUB2257
C                                    TRANSFORM INVERSE                     MSUB2258
  9Ø3 DO 9Ø4  JP= 1, M                                                     MSUB2259
      L = LL + IR                                                          MSUB226Ø
      IF ( ABS ( E(L) )  - TECOL)    914, 914, 9Ø5
  914 LL = LL + M                                                          MSUB2262
      GO TO 9Ø4                                                            MSUB2263
  9Ø5 T3 =  E(L) / T2                                                      MSUB2264
      E(L) =Ø.                                                             MSUB2265
      DO 9Ø6  I  = 1, M                                                    MSUB2266
      LL= LL +1                                                            MSUB2267
  9Ø6 E(LL) = E(LL) +T3* Y(I)                                              MSUB2268
  9Ø4 CONTINUE                                                             MSUB2269
C                                    TRANSFORM X                           MSUB227Ø
       T3 = X(IR) / T2                                                     MSUB2271
      X(IR) = Ø.                                                           MSUB2272
      DO 9Ø8  I  = 1, M                                                    MSUB2273
  9Ø8 X(I) = X(I) +T3* Y(I)                                                MSUB2274
C                              RESTORE Y(IR)                               MSUB2275
      Y(IR) = -T2                                                          MSUB2276
C                                                                          MSUB2277
  999 RETURN                                                               MSUB2278
      END                                                                  MSUB2279
C
      SUBROUTINE  ROW  ( IR, M, MF, JH, X, Y, TPIV )                       MSUB2282
CROWS       ROW SELECTION--COMPOSITE                                       MSUB2281
      DIMENSION JH(1), X(1), Y(1)                                          MSUB2283
C                                                                          MSUB2284
C AMONG EQS. WITH X=Ø, FIND MAX ABS(Y) AMONG ARTIFICIALS, OR, IF NONE,     MSUB2285
C  GET MAX POSITIVE Y(I) AMONG REALS.                                      MSUB2286
 1ØØØ IR = Ø                                                               MSUB2287
      AA = Ø.Ø                                                             MSUB2288
      IA  =  Ø                                                             MSUB2289
      DO 1Ø5Ø  I = MF,  M                                                  MSUB229Ø
      IF ( X(I) )  1Ø5Ø, 1Ø41, 1Ø5Ø                                        MSUB2291
 1Ø41 YI   =  ABS  ( Y(I) )
       IF   (   YI   -   TPIV    )   1Ø5Ø, 1Ø5Ø, 1Ø42                      MSUB2293
 1Ø42 IF   ( JH(I) )  1Ø43, 1Ø44, 1Ø43                                     MSUB2294
 1Ø43 IF   (IA)    1Ø5Ø, 1Ø48, 1Ø5Ø                                        MSUB2295
 1Ø48  IF   ( Y(I) )  1Ø5Ø, 1Ø5Ø, 1Ø45                                     MSUB2296
 1Ø44 IF    (IA)   1Ø45, 1Ø46, 1Ø45                                        MSUB2297
 1Ø45 IF    (  YI     -  AA    )    1Ø5Ø, 1Ø5Ø, 1Ø47                       MSUB2298
 1Ø46     IA = 1                                                           MSUB2299
 1Ø47 AA   =   YI                                                          MSUB23ØØ
      IR   =   I                                                           MSUB23Ø1
 1Ø5Ø CONTINUE                                                             MSUB23Ø2
```

```
      IF (IR)1Ø99,1ØØ1,1Ø99                                              MSUB23Ø3
 1ØØ1 AA = 1.ØE+2Ø                                                       MSUB23Ø4
C               FIND MIN. PIVOT AMONG POSITIVE EQUATIONS                 MSUB23Ø5
      DO 1Ø1Ø IT = MF , M                                                MSUB23Ø6
      IF  ( Y(IT)  -  TPIV )  1Ø1Ø, 1Ø1Ø, 1ØØ2                           MSUB23Ø7
 1ØØ2 IF  ( X(IT) ) 1Ø1Ø, 1Ø1Ø, 1ØØ3                                     MSUB23Ø8
 1ØØ3 XY = X(IT) / Y(IT)                                                 MSUB23Ø9
      IF ( XY - AA )  1ØØ4, 1ØØ5, 1Ø1Ø                                   MSUB231Ø
 1ØØ5 IF ( JH(IT))  1Ø1Ø, 1ØØ4, 1Ø1Ø                                     MSUB2311
 1ØØ4 AA = XY                                                            MSUB2312
      IR = IT                                                            MSUB2313
 1Ø1Ø CONTINUE                                                           MSUB2314
C  FIND PIVOT AMONG NEGATIVE EQUATIONS, IN WHICH X/Y IS LESS THAN THE    MSUB2315
C MINIMUM X/Y IN THE POSITIVE EQUATIONS, THAT HAS THE LARGEST ABSF(Y)    MSUB2316
 1Ø16 BB = - TPIV                                                        MSUB2317
      DO 1Ø3Ø I  = MF , M                                                MSUB2318
      IF (X(I))   1Ø12, 1Ø3Ø, 1Ø3Ø                                       MSUB2319
 1Ø12  IF  ( Y(I)    -   BB  )   1Ø22, 1Ø3Ø, 1Ø3Ø                        MSUB232Ø
 1Ø22 IF  (   Y(I) * AA  -  X(I)    )   1Ø24, 1Ø24, 1Ø3Ø                 MSUB2321
 1Ø24 BB  =  Y(I)                                                        MSUB2322
      IR  =  I                                                           MSUB2323
 1Ø3Ø CONTINUE                                                           MSUB2324
 1Ø99 RETURN                                                             MSUB2325
      END                                                                MSUB2326
C
      SUBROUTINE VER ( A, B, JH, X, E, KB, Y, N, ME, M, MF, INVC,        MSUB2329
     1                 NUMVR, NUMPV, INFS, LA, TPIV, TECOL, M2 )         MSUB233Ø
CVERS              FORMS INVERSE FROM  KB                                MSUB2328
      DIMENSION A(1), B(1), JH(1), X(1), E(1), KB(1), Y(1)               MSUB2331
C                                                                        MSUB2332
C                  INITIATE                                              MSUB2333
      IF (LA)  1121, 1121, 1122                                          MSUB2334
 1121 INVC = Ø                                                           MSUB2335
 1122 NUMVR   =  NUMVR  +1                                               MSUB2336
       DO 11Ø1  I = 1, M2                                                MSUB2337
 11Ø1 E(I)=Ø.                                                            MSUB2338
      MM=1                                                               MSUB2339
       DO 1113  I = 1, M                                                 MSUB234Ø
      E(MM) =1.Ø                                                         MSUB2341
      X(I) = B(I)                                                        MSUB2342
 1113 MM = MM + M + 1                                                    MSUB2343
        DO 111Ø  I = MF, M                                               MSUB2344
      IF (JH(I)) 1111, 111Ø, 1111                                        MSUB2345
 1111 JH(I) = 12345                                                      MSUB2346
 111Ø CONTINUE                                                           MSUB2347
      INFS = 1                                                           MSUB2348
C                  FORM INVERSE                                          MSUB2349
        DO 11Ø2  J = 1, N                                                MSUB235Ø
      IF ( KB(J) )  6ØØ , 11Ø2 , 6ØØ                                     MSUB2351
  6ØØ  CALL       JMY ( J, A, E, M, Y, ME )                              MSUB2352
C                        CHOOSE PIVOT                                    MSUB2353
 1114 TY = Ø.                                                            MSUB2354
        DO 11Ø4  I = MF, M                                               MSUB2355
      IF (JH(I)  - 12345 )  11Ø4, 11Ø5, 11Ø4                             MSUB2356
 11Ø5 IF ( ABS ( Y(I) ) -  TY )  11Ø4, 11Ø4, 11Ø6
 11Ø6 IR   = I                                                           MSUB2358
      TY =  ABS  ( Y(I) )
 11Ø4   CONTINUE                                                         MSUB236Ø
C                        TEST PIVOT                                      MSUB2361
      IF  (TY - TPIV )    11Ø7, 11Ø8, 11Ø8                               MSUB2362
C        BAD  PIVOT,  ROW IR,  COLUMN J                                  MSUB2363
 11Ø7 KB(J) = Ø                                                          MSUB2364
      GO TO 11Ø2                                                         MSUB2365
C                        PIVOT                                           MSUB2366
 11Ø8 JH(IR) = J                                                         MSUB2367
      KB(J)  = IR                                                        MSUB2368
  9ØØ  CALL       PIV  ( IR, Y, M, E, X, NUMPV, TECOL )                  MSUB2369
 11Ø2     CONTINUE                                                       MSUB237Ø
C                  RESET ARTIFICIALS                                     MSUB2371
        DO 11Ø9  I = 1, M                                                MSUB2372
      IF ( JH(I) - 12345   )  11Ø9, 1112, 11Ø9                           MSUB2373
 1112 JH(I) = Ø                                                          MSUB2374
 11Ø9   CONTINUE                                                         MSUB2375
      RETURN                                                             MSUB2376
      END                                                                MSUB2377
```

```
C
      SUBROUTINE XCK ( M, MF, JH, X, TZERO, JIN )                   MSUB238Ø
CXCKS        X CHECKER                                               MSUB2379
      DIMENSION JH(1), X(1)                                          MSUB2381
C                                                                    MSUB2382
C            RESET X AND CHECK FOR INFEASIBILITIES                   MSUB2383
 1212   JIN  =  Ø                                                    MSUB2384
      DO 12Ø1 I = MF, M                                              MSUB2385
      IF ( ABS  ( X(I) )  - TZERO)  12Ø2, 12Ø3, 12Ø3
 12Ø2 X(I) = Ø.Ø                                                     MSUB2387
      GO TO 12Ø1                                                     MSUB2388
 12Ø3 IF ( X(I) )  12Ø6, 12Ø1, 12Ø5                                  MSUB2389
 12Ø5 IF ( JH(I) )  12Ø1, 12Ø6, 12Ø1                                 MSUB239Ø
 12Ø6   JIN = 1                                                      MSUB2391
 12Ø1   CONTINUE                                                     MSUB2392
      RETURN                                                         MSUB2393
      END                                                            MSUB2394
```

APPENDIX C - User-Supplied Subroutines for Constrained Minimization Test Problem

```
      SUBROUTINE FUNCF (N, Z, F, NFUNCF)
C     ****************************************************************
C     COST FUNCTION EVALUATION.
C     ****************************************************************
C
      DIMENSION Z(1)
C
      Z1 = Z(1)
      Z2 = Z(2)
      Z3 = Z(3)
      Z4 = Z(4)
C
      F = Z1*Z1 + Z2*Z2 + 2.Ø*Z3*Z3 + Z4*Z4 - 5.Ø*Z1 - 5.Ø*Z2 -
     1    21.Ø*Z3 + 7.Ø*Z4
C
      RETURN
      END
      SUBROUTINE GRADF (N, Z, GRAD)
C     ******************************************************************
C     EVALUATES GRADIENT OF COST FUNCTION.
C     ******************************************************************
C
      DIMENSION Z(1), GRAD(1)
C
      Z1 = Z(1)
      Z2 = Z(2)
      Z3 = Z(3)
      Z4 = Z(4)
C
      GRAD(1) = 2.Ø * Z1 - 5.Ø
      GRAD(2) = 2.Ø * Z2 - 5.Ø
      GRAD(3) = 4.Ø * Z3 -21.Ø
      GRAD(4) = 2.Ø * Z4 + 7.Ø
C
      RETURN
      END
      SUBROUTINE FUNCG (N, JP, Z, G, PSI, NFUNCG)
C     *******************************************************************
C     EVALUATES CONVENTIONAL INEQUALITY CONSTRAINTS ( FUNCTION G )
C     ****************************************************************
C
      DIMENSION Z(1), G(1)
C
      Z1 = Z(1)
      Z2 = Z(2)
      Z3 = Z(3)
      Z4 = Z(4)
C
      G(1) = Z1*Z1 + Z2*Z2 + Z3*Z3 + Z4*Z4 + Z1 - Z2 + Z3 - Z4 - 8.Ø
      G(2) = Z1*Z1 + 2.Ø*Z2*Z2     + Z3*Z3 + 2.Ø*Z4*Z4 - Z1 - Z4 - 1Ø.Ø
      G(3) = 2.Ø*Z1*Z1 + Z2*Z2     + Z3*Z3 + 2.Ø*Z1 - 2.Ø*Z2 - Z4 - 5.Ø
C
      DO 1ØØ I=1,JP
1ØØ   IF (G(I) .GT. PSI) PSI = G(I)
C
      RETURN
      END
      SUBROUTINE GRADG (N, J, Z, GRAD)
C     ****************************************************************
C     EVALUATES GRADIENTS OF G FUNCTIONS
C     ****************************************************************
C
      DIMENSION Z(1), GRAD(1)
C
      GO TO (1, 2, 3) , J
C
1     GRAD(1) = 2.Ø * Z(1) + 1.Ø
      GRAD(2) = 2.Ø * Z(2) - 1.Ø
      GRAD(3) = 2.Ø * Z(3) + 1.Ø
      GRAD(4) = 2.Ø * Z(4) - 1.Ø
      RETURN
```

```
C
2     GRAD(1) = 2.Ø * Z(1) - 1.Ø
      GRAD(2) = 4.Ø * Z(2)
      GRAD(3) = 2.Ø * Z(3)
      GRAD(4) = 4.Ø * Z(4) - 1.Ø
      RETURN
C
3     GRAD(1) = 4.Ø * Z(1) + 2.Ø
      GRAD(2) = 2.Ø * Z(2) - 1.Ø
      GRAD(3) = 2.Ø * Z(3)
      GRAD(4) = -1.Ø
      RETURN
C
      END
      SUBROUTINE FUNCPH (N, NJQ, JQ, Z, WØ, WC, DELTAW, NQ, PHI, PSI,
     1                   NFUNCP)
C     ****************************************************************
C     EVALUATES DYNAMIC CONSTRAINTS ( FUNCTIONS PHI )
C     ****************************************************************
C
      DIMENSION Z(1), PHI(NJQ,1)
C
      RETURN
      END
      SUBROUTINE GRADPH (N, NJQ, NACTIV, JQ, WØ, WC, DELTAW, NQ, NEPTF,
     1                   L, Z, K, GRAD, IGRAD)
C     ****************************************************************
C     EVALUATES GRADIENTS OF PHI CONSTRAINT FUNCTIONS
C     ****************************************************************
C
      DIMENSION Z(1), GRAD(1), NEPTF(NJQ,1)
C
      RETURN
      END
```

APPENDIX D - User-Supplied Subroutines for PID Controller Problem

```
      SUBROUTINE FUNCF (N, Z, F, NFUNCF)
C     ****************************************************************
C     COST FUNCTION EVALUATION.
C     ****************************************************************
C
      DIMENSION Z(1)
C
      Z1 = Z(1)
      Z2 = Z(2)
      Z3 = Z(3)
C
      DENOM = Z2 * (4Ø8.Ø + 56.Ø * Z1 - 5Ø.Ø * Z2 + 6Ø.Ø * Z3 +
     1        1Ø.Ø * Z1 * Z3 - 2.Ø * Z1 * Z1)
      ANUM = Z2 * (122.Ø + 17.Ø * Z1 - 5.Ø * Z2 + 6.Ø * Z3 + Z1 * Z3) +
     1       18Ø.Ø * Z3 - 36.Ø * Z1 + 1224.Ø
      F = ANUM / DENOM
C
      RETURN
      END
      SUBROUTINE GRADF (N, Z, GRAD)
C     ******************************************************************
C     EVALUATES GRADIENT OF COST FUNCTION.
C     ******************************************************************
C
      DIMENSION Z(1), GRAD(1)
C
      Z1 = Z(1)
      Z2 = Z(2)
      Z3 = Z(3)
C
      DENOM = Z2 * (4Ø8.Ø + 56.Ø * Z1 - 5Ø.Ø * Z2 + 6Ø.Ø * Z3 +
     1        1Ø.Ø * Z1 * Z3 - 2.Ø * Z1 * Z1)
      ANUM = Z2 * (122.Ø + 17.Ø * Z1 - 5.Ø * Z2 + 6.Ø * Z3 + Z1 * Z3) +
     1       18Ø.Ø * Z3 - 36.Ø * Z1 + 1224.Ø
      GRAD(1) = (17.Ø * Z2 + Z2 * Z3 -36.Ø) / DENOM -
     1          (56.Ø * Z2 + 1Ø.Ø * Z2 * Z3 - 4.Ø * Z1 * Z2) * ANUM /
     2          (DENOM * DENOM)
      GRAD(2) = (122.Ø + 17.Ø * Z1 - 1Ø.Ø * Z2 + 6.Ø * Z3 + Z1 * Z3) / DENOM
     1          - (4Ø8.Ø + 56.Ø * Z1 - 1ØØ.Ø * Z2 + 6Ø * Z3 + 1Ø.Ø * Z1 * Z3
     2          -  2.Ø * Z1 * Z1) * ANUM / (DENOM * DENOM)
      GRAD(3) = (6.Ø * Z2 + Z1 * Z2 + 18Ø.Ø) / DENOM -
     1          (6Ø.Ø * Z2 + 1Ø.Ø * Z1 * Z2) * ANUM / (DENOM * DENOM)
      RETURN
      END
      SUBROUTINE FUNCG (N, JP, Z, G, PSI, NFUNCG)
C     *******************************************************************
C     EVALUATES CONVENTIONAL INEQUALITY CONSTRAINTS ( FUNCTION G )
C     ****************************************************************
C
      DIMENSION Z(1), G(1)
C
      G(1) = -Z(1)
      G(2) = -Z(2) + Ø.1
      G(3) = -Z(3)
      G(4) = Z(1) - 1ØØ.Ø
      G(5) = Z(2) - 1ØØ.Ø
      G(6) = Z(3) - 1ØØ.Ø
C
      DO 1ØØ I=1,JP
1ØØ   IF (G(I) .GT. PSI) PSI = G(I)
C
      RETURN
      END
      SUBROUTINE GRADG (N, J, Z, GRAD)
C     ****************************************************************
C     EVALUATES GRADIENTS OF G FUNCTIONS
C     ****************************************************************
C
      DIMENSION Z(1), GRAD(1)
C
      GO TO (1, 2, 3, 4, 5, 6) , J
C
1     GRAD(1) = -1.Ø
      GRAD(2) = Ø.Ø
      GRAD(3) = Ø.Ø
      RETURN
```

```
C
2       GRAD(1) = 0.0
        GRAD(2) = -1.0
        GRAD(3) = 0.0
        RETURN
C
3       GRAD(1) = 0.0
        GRAD(2) = 0.0
        GRAD(3) = -1.0
        RETURN
C
4       GRAD(1) = 1.0
        GRAD(2) = 0.0
        GRAD(3) = 0.0
        RETURN
C
5       GRAD(1) = 0.0
        GRAD(2) = 1.0
        GRAD(3) = 0.0
        RETURN
C
6       GRAD(1) = 0.0
        GRAD(2) = 0.0
        GRAD(3) = 1.0
        RETURN
C
        END
        SUBROUTINE FUNCPH (N, NJQ, JQ, Z, W0, WC, DELTAW, NQ, PHI, PSI,
      1                   NFUNCP)
C       ****************************************************************
C       EVALUATES DYNAMIC CONSTRAINTS ( FUNCTIONS PHI )
C       ****************************************************************
C
        DIMENSION Z(1), PHI(NJQ,1)
C
        W  = W0
        W2 = W * W
        DO 100 I=1,NQ
        B = ((W2 + 9.0) * W2 + 4.0) * W2 + 36.0
        AR= (((W2 + 9.0 - Z(3)) * W2 + 4.0 + 8.0 * Z(3) - 5.0 * Z(1) +
      1     Z(2)) * W2 + 36.0 + 6.0 * Z(1) - 8.0 * Z(2)) / B
        AI= (((Z(1) - 5.0 * Z(3)) * W2 + 6.0 * Z(3) + 5.0 * Z(2) -8.0 *
      1     Z(1)) * W - (6.0 * Z(2)) / W) / B
        PHI(1,I) = AI - 3.33*AR*AR + 1.0
        W = W + DELTAW
100     CONTINUE
C
        DO 110 L=1,JQ
        DO 110 K=1,NQ
        IF (PHI(L,K) .GT. PSI) PSI = PHI(L,K)
110     CONTINUE
C
        RETURN
        END
        SUBROUTINE GRADPH (N, NJQ, NACTIV, JQ, W0, WC, DELTAW, NQ, NEPTF,
      1                   L, Z, K, GRAD, IGRAD)
C       ****************************************************************
C       EVALUATES GRADIENTS OF PHI CONSTRAINT FUNCTIONS
C       ****************************************************************
C
        DIMENSION Z(1), GRAD(1), NEPTF(NJQ,1)
C
        W = (K-1) * DELTAW + W0
        W2 = W * W
        B  = ((W2 + 9.0) * W2 + 4.0) * W2 + 36.0
        AR= (((W2 + 9.0 - Z(3)) * W2 + 4.0 + 8.0 * Z(3) - 5.0 * Z(1) +
      1     Z(2)) * W2 + 36.0 + 6.0 * Z(1) - 8.0 * Z(2)) / B
        GRAD(1) = (((W2 - 8.0) * W) / B) - 6.66 * AR * ((-5.0 * W2 +
      1          6.0) / B)
        GRAD(2) = ((5.0 * W2 - 6.0) / ( W * B )) - 6.66 * AR * ((W2 -
      1          8.0) / B)
        GRAD(3) = (((-5.0*W2 + 6.0) * W)/B) - 6.66*AR * (((-W2+8.0) *
      1          W2) / B)
C
        RETURN
        END
```

EARTHQUAKE ENGINEERING RESEARCH CENTER REPORTS

NOTE: Numbers in parenthesis are Accession Numbers assigned by the National Technical Information Service; these are followed by a price code. Copies of the reports may be ordered from the National Technical Information Service, 5285 Port Royal Road, Springfield, Virginia, 22161. Accession Numbers should be quoted on orders for reports (PB --- ---) and remittance must accompany each order. Reports without this information were not available at time of printing. Upon request, EERC will mail inquirers this information when it becomes available.

EERC 67-1 "Feasibility Study Large-Scale Earthquake Simulator Facility," by J. Penzien, J.G. Bouwkamp, R.W. Clough and D. Rea - 1967 (PB 187 905)A07

EERC 68-1 Unassigned

EERC 68-2 "Inelastic Behavior of Beam-to-Column Subassemblages Under Repeated Loading," by V.V. Bertero - 1968 (PB 184 888)A05

EERC 68-3 "A Graphical Method for Solving the Wave Reflection-Refraction Problem," by H.D. McNiven and Y. Mengi - 1968 (PB 187 943)A03

EERC 68-4 "Dynamic Properties of McKinley School Buildings," by D. Rea, J.G. Bouwkamp and R.W. Clough - 1968 (PB 187 902)A07

EERC 68-5 "Characteristics of Rock Motions During Earthquakes," by H.B. Seed, I.M. Idriss and F.W. Kiefer - 1968 (PB 188 338)A03

EERC 69-1 "Earthquake Engineering Research at Berkeley," - 1969 (PB 187 906)A11

EERC 69-2 "Nonlinear Seismic Response of Earth Structures," by M. Dibaj and J. Penzien - 1969 (PB 187 904)A08

EERC 69-3 "Probabilistic Study of the Behavior of Structures During Earthquakes," by R. Ruiz and J. Penzien - 1969 (PB 187 886)A06

EERC 69-4 "Numerical Solution of Boundary Value Problems in Structural Mechanics by Reduction to an Initial Value Formulation," by N. Distefano and J. Schujman - 1969 (PB 187 942)A02

EERC 69-5 "Dynamic Programming and the Solution of the Biharmonic Equation," by N. Distefano - 1969 (PB 187 941)A03

EERC 69-6 "Stochastic Analysis of Offshore Tower Structures," by A.K. Malhotra and J. Penzien - 1969 (PB 187 903)A09

EERC 69-7 "Rock Motion Accelerograms for High Magnitude Earthquakes," by H.B. Seed and I.M. Idriss - 1969 (PB 187 940)A02

EERC 69-8 "Structural Dynamics Testing Facilities at the University of California, Berkeley," by R.M. Stephen, J.G. Bouwkamp, R.W. Clough and J. Penzien - 1969 (PB 189 111)A04

EERC 69-9 "Seismic Response of Soil Deposits Underlain by Sloping Rock Boundaries," by H. Dezfulian and H.B. Seed 1969 (PB 189 114)A03

EERC 69-10 "Dynamic Stress Analysis of Axisymmetric Structures Under Arbitrary Loading," by S. Ghosh and E.L. Wilson 1969 (PB 189 026)A10

EERC 69-11 "Seismic Behavior of Multistory Frames Designed by Different Philosophies," by J.C. Anderson and V. V. Bertero - 1969 (PB 190 662)A10

EERC 69-12 "Stiffness Degradation of Reinforcing Concrete Members Subjected to Cyclic Flexural Moments," by V.V. Bertero, B. Bresler and H. Ming Liao - 1969 (PB 202 942)A07

EERC 69-13 "Response of Non-Uniform Soil Deposits to Travelling Seismic Waves," by H. Dezfulian and H.B. Seed - 1969 (PB 191 023)A03

EERC 69-14 "Damping Capacity of a Model Steel Structure," by D. Rea, R.W. Clough and J.G. Bouwkamp - 1969 (PB 190 663)A06

EERC 69-15 "Influence of Local Soil Conditions on Building Damage Potential during Earthquakes," by H.B. Seed and I.M. Idriss - 1969 (PB 191 036)A03

EERC 69-16 "The Behavior of Sands Under Seismic Loading Conditions," by M.L. Silver and H.B. Seed - 1969 (AD 714 982)A07

EERC 70-1 "Earthquake Response of Gravity Dams," by A.K. Chopra - 1970 (AD 709 640)A03

EERC 70-2 "Relationships between Soil Conditions and Building Damage in the Caracas Earthquake of July 29, 1967," by H.B. Seed, I.M. Idriss and H. Dezfulian - 1970 (PB 195 762)A05

EERC 70-3 "Cyclic Loading of Full Size Steel Connections," by E.P. Popov and R.M. Stephen - 1970 (PB 213 545)A04

EERC 70-4 "Seismic Analysis of the Charaima Building, Caraballeda, Venezuela," by Subcommittee of the SEAONC Research Committee: V.V. Bertero, P.F. Fratessa, S.A. Mahin, J.H. Sexton, A.C. Scordelis, E.L. Wilson, L.A. Wyllie, H.B. Seed and J. Penzien, Chairman - 1970 (PB 201 455)A06

EERC 70-5 "A Computer Program for Earthquake Analysis of Dams," by A.K. Chopra and P. Chakrabarti - 1970 (AD 723 994)A05

EERC 70-6 "The Propagation of Love Waves Across Non-Horizontally Layered Structures," by J. Lysmer and L.A. Drake 1970 (PB 197 896)A03

EERC 70-7 "Influence of Base Rock Characteristics on Ground Response," by J. Lysmer, H.B. Seed and P.B. Schnabel 1970 (PB 197 897)A03

EERC 70-8 "Applicability of Laboratory Test Procedures for Measuring Soil Liquefaction Characteristics under Cyclic Loading," by H.B. Seed and W.H. Peacock - 1970 (PB 198 016)A03

EERC 70-9 "A Simplified Procedure for Evaluating Soil Liquefaction Potential," by H.B. Seed and I.M. Idriss - 1970 (PB 198 009)A03

EERC 70-10 "Soil Moduli and Damping Factors for Dynamic Response Analysis," by H.B. Seed and I.M. Idriss - 1970 (PB 197 869)A03

EERC 71-1 "Koyna Earthquake of December 11, 1967 and the Performance of Koyna Dam," by A.K. Chopra and P. Chakrabarti 1971 (AD 731 496)A06

EERC 71-2 "Preliminary In-Situ Measurements of Anelastic Absorption in Soils Using a Prototype Earthquake Simulator," by R.D. Borcherdt and P.W. Rodgers - 1971 (PB 201 454)A03

EERC 71-3 "Static and Dynamic Analysis of Inelastic Frame Structures," by F.L. Porter and G.H. Powell - 1971 (PB 210 135)A06

EERC 71-4 "Research Needs in Limit Design of Reinforced Concrete Structures," by V.V. Bertero - 1971 (PB 202 943)A04

EERC 71-5 "Dynamic Behavior of a High-Rise Diagonally Braced Steel Building," by D. Rea, A.A. Shah and J.G. Bouwkamp 1971 (PB 203 584)A06

EERC 71-6 "Dynamic Stress Analysis of Porous Elastic Solids Saturated with Compressible Fluids," by J. Ghaboussi and E. L. Wilson - 1971 (PB 211 396)A06

EERC 71-7 "Inelastic Behavior of Steel Beam-to-Column Subassemblages," by H. Krawinkler, V.V. Bertero and E.P. Popov 1971 (PB 211 335)A14

EERC 71-8 "Modification of Seismograph Records for Effects of Local Soil Conditions," by P. Schnabel, H.B. Seed and J. Lysmer - 1971 (PB 214 450)A03

EERC 72-1 "Static and Earthquake Analysis of Three Dimensional Frame and Shear Wall Buildings," by E.L. Wilson and H.H. Dovey - 1972 (PB 212 904)A05

EERC 72-2 "Accelerations in Rock for Earthquakes in the Western United States," by P.B. Schnabel and H.B. Seed - 1972 (PB 213 100)A03

EERC 72-3 "Elastic-Plastic Earthquake Response of Soil-Building Systems," by T. Minami - 1972 (PB 214 868)A08

EERC 72-4 "Stochastic Inelastic Response of Offshore Towers to Strong Motion Earthquakes," by M.K. Kaul - 1972 (PB 215 713)A05

EERC 72-5 "Cyclic Behavior of Three Reinforced Concrete Flexural Members with High Shear," by E.P. Popov, V.V. Bertero and H. Krawinkler - 1972 (PB 214 555)A05

EERC 72-6 "Earthquake Response of Gravity Dams Including Reservoir Interaction Effects," by P. Chakrabarti and A.K. Chopra - 1972 (AD 762 330)A08

EERC 72-7 "Dynamic Properties of Pine Flat Dam," by D. Rea, C.Y. Liaw and A.K. Chopra - 1972 (AD 763 928)A05

EERC 72-8 "Three Dimensional Analysis of Building Systems," by E.L. Wilson and H.H. Dovey - 1972 (PB 222 438)A06

EERC 72-9 "Rate of Loading Effects on Uncracked and Repaired Reinforced Concrete Members," by S. Mahin, V.V. Bertero, D. Rea and M. Atalay - 1972 (PB 224 520)A08

EERC 72-10 "Computer Program for Static and Dynamic Analysis of Linear Structural Systems," by E.L. Wilson, K.-J. Bathe, J.E. Peterson and H.H.Dovey - 1972 (PB 220 437)A04

EERC 72-11 "Literature Survey - Seismic Effects on Highway Bridges," by T. Iwasaki, J. Penzien and R.W. Clough - 1972 (PB 215 613)A19

EERC 72-12 "SHAKE-A Computer Program for Earthquake Response Analysis of Horizontally Layered Sites," by P.B. Schnabel and J. Lysmer - 1972 (PB 220 207)A06

EERC 73-1 "Optimal Seismic Design of Multistory Frames," by V.V. Bertero and H. Kamil - 1973

EERC 73-2 "Analysis of the Slides in the San Fernando Dams During the Earthquake of February 9, 1971," by H.B. Seed, K.L. Lee, I.M. Idriss and F. Makdisi - 1973 (PB 223 402)A14

EERC 73-3 "Computer Aided Ultimate Load Design of Unbraced Multistory Steel Frames," by M.B. El-Hafez and G.H. Powell 1973 (PB 248 315)A09

EERC 73-4 "Experimental Investigation into the Seismic Behavior of Critical Regions of Reinforced Concrete Components as Influenced by Moment and Shear," by M. Celebi and J. Penzien - 1973 (PB 215 884)A09

EERC 73-5 "Hysteretic Behavior of Epoxy-Repaired Reinforced Concrete Beams," by M. Celebi and J. Penzien - 1973 (PB 239 568)A03

EERC 73-6 "General Purpose Computer Program for Inelastic Dynamic Response of Plane Structures," by A. Kanaan and G.H. Powell - 1973 (PB 221 260)A08

EERC 73-7 "A Computer Program for Earthquake Analysis of Gravity Dams Including Reservoir Interaction," by P. Chakrabarti and A.K. Chopra - 1973 (AD 766 271)A04

EERC 73-8 "Behavior of Reinforced Concrete Deep Beam-Column Subassemblages Under Cyclic Loads," by O. Küstü and J.G. Bouwkamp - 1973 (PB 246 117)A12

EERC 73-9 "Earthquake Analysis of Structure-Foundation Systems," by A.K. Vaish and A.K. Chopra - 1973 (AD 766 272)A07

EERC 73-10 "Deconvolution of Seismic Response for Linear Systems," by R.B. Reimer - 1973 (PB 227 179)A08

EERC 73-11 "SAP IV: A Structural Analysis Program for Static and Dynamic Response of Linear Systems," by K.-J. Bathe, E.L. Wilson and F.E. Peterson - 1973 (PB 221 967)A09

EERC 73-12 "Analytical Investigations of the Seismic Response of Long, Multiple Span Highway Bridges," by W.S. Tseng and J. Penzien - 1973 (PB 227 816)A10

EERC 73-13 "Earthquake Analysis of Multi-Story Buildings Including Foundation Interaction," by A.K. Chopra and J.A. Gutierrez - 1973 (PB 222 970)A03

EERC 73-14 "ADAP: A Computer Program for Static and Dynamic Analysis of Arch Dams," by R.W. Clough, J.M. Raphael and S. Mojtahedi - 1973 (PB 223 763)A09

EERC 73-15 "Cyclic Plastic Analysis of Structural Steel Joints," by R.B. Pinkney and R.W. Clough - 1973 (PB 226 843)A08

EERC 73-16 "QUAD-4: A Computer Program for Evaluating the Seismic Response of Soil Structures by Variable Damping Finite Element Procedures," by I.M. Idriss, J. Lysmer, R. Hwang and H.B. Seed - 1973 (PB 229 424)A05

EERC 73-17 "Dynamic Behavior of a Multi-Story Pyramid Shaped Building," by R.M. Stephen, J.P. Hollings and J.G. Bouwkamp - 1973 (PB 240 718)A06

EERC 73-18 "Effect of Different Types of Reinforcing on Seismic Behavior of Short Concrete Columns," by V.V. Bertero, J. Hollings, O. Küstü, R.M. Stephen and J.G. Bouwkamp - 1973

EERC 73-19 "Olive View Medical Center Materials Studies, Phase I," by B. Bresler and V.V. Bertero - 1973 (PB 235 986)A06

EERC 73-20 "Linear and Nonlinear Seismic Analysis Computer Programs for Long Multiple-Span Highway Bridges," by W.S. Tseng and J. Penzien - 1973

EERC 73-21 "Constitutive Models for Cyclic Plastic Deformation of Engineering Materials," by J.M. Kelly and P.P. Gillis 1973 (PB 226 024)A03

EERC 73-22 "DRAIN - 2D User's Guide," by G.H. Powell - 1973 (PB 227 016)A05

EERC 73-23 "Earthquake Engineering at Berkeley - 1973," (PB 226 033)A11

EERC 73-24 Unassigned

EERC 73-25 "Earthquake Response of Axisymmetric Tower Structures Surrounded by Water," by C.Y. Liaw and A.K. Chopra 1973 (AD 773 052)A09

EERC 73-26 "Investigation of the Failures of the Olive View Stairtowers During the San Fernando Earthquake and Their Implications on Seismic Design," by V.V. Bertero and R.G. Collins - 1973 (PB 235 106)A13

EERC 73-27 "Further Studies on Seismic Behavior of Steel Beam-Column Subassemblages," by V.V. Bertero, H. Krawinkler and E.P. Popov - 1973 (PB 234 172)A06

EERC 74-1 "Seismic Risk Analysis," by C.S. Oliveira - 1974 (PB 235 920)A06

EERC 74-2 "Settlement and Liquefaction of Sands Under Multi-Directional Shaking," by R. Pyke, C.K. Chan and H.B. Seed 1974

EERC 74-3 "Optimum Design of Earthquake Resistant Shear Buildings," by D. Ray, K.S. Pister and A.K. Chopra - 1974 (PB 231 172)A06

EERC 74-4 "LUSH - A Computer Program for Complex Response Analysis of Soil-Structure Systems," by J. Lysmer, T. Udaka, H.B. Seed and R. Hwang - 1974 (PB 236 796)A05

EERC 74-5 "Sensitivity Analysis for Hysteretic Dynamic Systems: Applications to Earthquake Engineering," by D. Ray 1974 (PB 233 213)A06

EERC 74-6 "Soil Structure Interaction Analyses for Evaluating Seismic Response," by H.B. Seed, J. Lysmer and R. Hwang 1974 (PB 236 519)A04

EERC 74-7 Unassigned

EERC 74-8 "Shaking Table Tests of a Steel Frame - A Progress Report," by R.W. Clough and D. Tang - 1974 (PB 240 869)A03

EERC 74-9 "Hysteretic Behavior of Reinforced Concrete Flexural Members with Special Web Reinforcement," by V.V. Bertero, E.P. Popov and T.Y. Wang - 1974 (PB 236 797)A07

EERC 74-10 "Applications of Reliability-Based, Global Cost Optimization to Design of Earthquake Resistant Structures," by E. Vitiello and K.S. Pister - 1974 (PB 237 231)A06

EERC 74-11 "Liquefaction of Gravelly Soils Under Cyclic Loading Conditions," by R.T. Wong, H.B. Seed and C.K. Chan 1974 (PB 242 042)A03

EERC 74-12 "Site-Dependent Spectra for Earthquake-Resistant Design," by H.B. Seed, C. Ugas and J. Lysmer - 1974 (PB 240 953)A03

EERC 74-13 "Earthquake Simulator Study of a Reinforced Concrete Frame," by P. Hidalgo and R.W. Clough - 1974 (PB 241 944)A13

EERC 74-14 "Nonlinear Earthquake Response of Concrete Gravity Dams," by N. Pal - 1974 (AD/A 006 583)A06

EERC 74-15 "Modeling and Identification in Nonlinear Structural Dynamics - I. One Degree of Freedom Models," by N. Distefano and A. Rath - 1974 (PB 241 548)A06

EERC 75-1 "Determination of Seismic Design Criteria for the Dumbarton Bridge Replacement Structure, Vol. I: Description, Theory and Analytical Modeling of Bridge and Parameters," by F. Baron and S.-H. Pang - 1975 (PB 259 407)A15

EERC 75-2 "Determination of Seismic Design Criteria for the Dumbarton Bridge Replacement Structure, Vol. II: Numerical Studies and Establishment of Seismic Design Criteria," by F. Baron and S.-H. Pang - 1975 (PB 259 408)A11 (For set of EERC 75-1 and 75-2 (PB 259 406))

EERC 75-3 "Seismic Risk Analysis for a Site and a Metropolitan Area," by C.S. Oliveira - 1975 (PB 248 134)A09

EERC 75-4 "Analytical Investigations of Seismic Response of Short, Single or Multiple-Span Highway Bridges," by M.-C. Chen and J. Penzien - 1975 (PB 241 454)A09

EERC 75-5 "An Evaluation of Some Methods for Predicting Seismic Behavior of Reinforced Concrete Buildings," by S.A. Mahin and V.V. Bertero - 1975 (PB 246 306)A16

EERC 75-6 "Earthquake Simulator Study of a Steel Frame Structure, Vol. I: Experimental Results," by R.W. Clough and D.T. Tang - 1975 (PB 243 981)A13

EERC 75-7 "Dynamic Properties of San Bernardino Intake Tower," by D. Rea, C.-Y. Liaw and A.K. Chopra - 1975 (AD/A008 406) A05

EERC 75-8 "Seismic Studies of the Articulation for the Dumbarton Bridge Replacement Structure, Vol. I: Description, Theory and Analytical Modeling of Bridge Components," by F. Baron and R.E. Hamati - 1975 (PB 251 539)A07

EERC 75-9 "Seismic Studies of the Articulation for the Dumbarton Bridge Replacement Structure, Vol. 2: Numerical Studies of Steel and Concrete Girder Alternates," by F. Baron and R.E. Hamati - 1975 (PB 251 540)A10

EERC 75-10 "Static and Dynamic Analysis of Nonlinear Structures," by D.P. Mondkar and G.H. Powell - 1975 (PB 242 434)A08

EERC 75-11 "Hysteretic Behavior of Steel Columns," by E.P. Popov, V.V. Bertero and S. Chandramouli - 1975 (PB 252 365)A11

EERC 75-12 "Earthquake Engineering Research Center Library Printed Catalog," - 1975 (PB 243 711)A26

EERC 75-13 "Three Dimensional Analysis of Building Systems (Extended Version)," by E.L. Wilson, J.P. Hollings and H.H. Dovey - 1975 (PB 243 989)A07

EERC 75-14 "Determination of Soil Liquefaction Characteristics by Large-Scale Laboratory Tests," by P. De Alba, C.K. Chan and H.B. Seed - 1975 (NUREG 0027)A08

EERC 75-15 "A Literature Survey - Compressive, Tensile, Bond and Shear Strength of Masonry," by R.L. Mayes and R.W. Clough - 1975 (PB 246 292)A10

EERC 75-16 "Hysteretic Behavior of Ductile Moment Resisting Reinforced Concrete Frame Components," by V.V. Bertero and E.P. Popov - 1975 (PB 246 388)A05

EERC 75-17 "Relationships Between Maximum Acceleration, Maximum Velocity, Distance from Source, Local Site Conditions for Moderately Strong Earthquakes," by H.B. Seed, R. Murarka, J. Lysmer and I.M. Idriss - 1975 (PB 248 172)A03

EERC 75-18 "The Effects of Method of Sample Preparation on the Cyclic Stress-Strain Behavior of Sands," by J. Mulilis, C.K. Chan and H.B. Seed - 1975 (Summarized in EERC 75-28)

EERC 75-19 "The Seismic Behavior of Critical Regions of Reinforced Concrete Components as Influenced by Moment, Shear and Axial Force," by M.B. Atalay and J. Penzien - 1975 (PB 258 842)A11

EERC 75-20 "Dynamic Properties of an Eleven Story Masonry Building," by R.M. Stephen, J.P. Hollings, J.G. Bouwkamp and D. Jurukovski - 1975 (PB 246 945)A04

EERC 75-21 "State-of-the-Art in Seismic Strength of Masonry - An Evaluation and Review," by R.L. Mayes and R.W. Clough 1975 (PB 249 040)A07

EERC 75-22 "Frequency Dependent Stiffness Matrices for Viscoelastic Half-Plane Foundations," by A.K. Chopra, P. Chakrabarti and G. Dasgupta - 1975 (PB 248 121)A07

EERC 75-23 "Hysteretic Behavior of Reinforced Concrete Framed Walls," by T.Y. Wong, V.V. Bertero and E.P. Popov - 1975

EERC 75-24 "Testing Facility for Subassemblages of Frame-Wall Structural Systems," by V.V. Bertero, E.P. Popov and T. Endo - 1975

EERC 75-25 "Influence of Seismic History on the Liquefaction Characteristics of Sands," by H.B. Seed, K. Mori and C.K. Chan - 1975 (Summarized in EERC 75-28)

EERC 75-26 "The Generation and Dissipation of Pore Water Pressures during Soil Liquefaction," by H.B. Seed, P.P. Martin and J. Lysmer - 1975 (PB 252 648)A03

EERC 75-27 "Identification of Research Needs for Improving Aseismic Design of Building Structures," by V.V. Bertero 1975 (PB 248 136)A05

EERC 75-28 "Evaluation of Soil Liquefaction Potential during Earthquakes," by H.B. Seed, I. Arango and C.K. Chan - 1975 (NUREG 0026)A13

EERC 75-29 "Representation of Irregular Stress Time Histories by Equivalent Uniform Stress Series in Liquefaction Analyses," by H.B. Seed, I.M. Idriss, F. Makdisi and N. Banerjee - 1975 (PB 252 635)A03

EERC 75-30 "FLUSH - A Computer Program for Approximate 3-D Analysis of Soil-Structure Interaction Problems," by J. Lysmer, T. Udaka, C.-F. Tsai and H.B. Seed - 1975 (PB 259 332)A07

EERC 75-31 "ALUSH - A Computer Program for Seismic Response Analysis of Axisymmetric Soil-Structure Systems," by E. Berger, J. Lysmer and H.B. Seed - 1975

EERC 75-32 "TRIP and TRAVEL - Computer Programs for Soil-Structure Interaction Analysis with Horizontally Travelling Waves," by T. Udaka, J. Lysmer and H.B. Seed - 1975

EERC 75-33 "Predicting the Performance of Structures in Regions of High Seismicity," by J. Penzien - 1975 (PB 248 130)A03

EERC 75-34 "Efficient Finite Element Analysis of Seismic Structure - Soil - Direction," by J. Lysmer, H.B. Seed, T. Udaka, R.N. Hwang and C.-F. Tsai - 1975 (PB 253 570)A03

EERC 75-35 "The Dynamic Behavior of a First Story Girder of a Three-Story Steel Frame Subjected to Earthquake Loading," by R.W. Clough and L.-Y. Li - 1975 (PB 248 841)A05

EERC 75-36 "Earthquake Simulator Study of a Steel Frame Structure, Volume II - Analytical Results," by D.T. Tang - 1975 (PB 252 926)A10

EERC 75-37 "ANSR-I General Purpose Computer Program for Analysis of Non-Linear Structural Response," by D.P. Mondkar and G.H. Powell - 1975 (PB 252 386)A08

EERC 75-38 "Nonlinear Response Spectra for Probabilistic Seismic Design and Damage Assessment of Reinforced Concrete Structures," by M. Murakami and J. Penzien - 1975 (PB 259 530)A05

EERC 75-39 "Study of a Method of Feasible Directions for Optimal Elastic Design of Frame Structures Subjected to Earthquake Loading," by N.D. Walker and K.S. Pister - 1975 (PB 257 781)A06

EERC 75-40 "An Alternative Representation of the Elastic-Viscoelastic Analogy," by G. Dasgupta and J.L. Sackman - 1975 (PB 252 173)A03

EERC 75-41 "Effect of Multi-Directional Shaking on Liquefaction of Sands," by H.B. Seed, R. Pyke and G.R. Martin - 1975 (PB 258 781)A03

EERC 76-1 "Strength and Ductility Evaluation of Existing Low-Rise Reinforced Concrete Buildings - Screening Method," by T. Okada and B. Bresler - 1976 (PB 257 906)A11

EERC 76-2 "Experimental and Analytical Studies on the Hysteretic Behavior of Reinforced Concrete Rectangular and T-Beams," by S.-Y.M. Ma, E.P. Popov and V.V. Bertero - 1976 (PB 260 843)A12

EERC 76-3 "Dynamic Behavior of a Multistory Triangular-Shaped Building," by J. Petrovski, R.M. Stephen, E. Gartenbaum and J.G. Bouwkamp - 1976 (PB 273 279)A07

EERC 76-4 "Earthquake Induced Deformations of Earth Dams," by N. Serff, H.B. Seed, F.I. Makdisi & C.-Y. Chang - 1976 (PB 292 065)A08

EERC 76-5 "Analysis and Design of Tube-Type Tall Building Structures," by H. de Clercq and G.H. Powell - 1976 (PB 252 220) A10

EERC 76-6 "Time and Frequency Domain Analysis of Three-Dimensional Ground Motions, San Fernando Earthquake," by T. Kubo and J. Penzien (PB 260 556)A11

EERC 76-7 "Expected Performance of Uniform Building Code Design Masonry Structures," by R.L. Mayes, Y. Omote, S.W. Chen and R.W. Clough - 1976 (PB 270 098)A05

EERC 76-8 "Cyclic Shear Tests of Masonry Piers, Volume 1 - Test Results," by R.L. Mayes, Y. Omote, R.W. Clough - 1976 (PB 264 424)A06

EERC 76-9 "A Substructure Method for Earthquake Analysis of Structure - Soil Interaction," by J.A. Gutierrez and A.K. Chopra - 1976 (PB 257 783)A08

EERC 76-10 "Stabilization of Potentially Liquefiable Sand Deposits using Gravel Drain Systems," by H.B. Seed and J.R. Booker - 1976 (PB 258 820)A04

EERC 76-11 "Influence of Design and Analysis Assumptions on Computed Inelastic Response of Moderately Tall Frames," by G.H. Powell and D.G. Row - 1976 (PB 271 409)A06

EERC 76-12 "Sensitivity Analysis for Hysteretic Dynamic Systems: Theory and Applications," by D. Ray, K.S. Pister and E. Polak - 1976 (PB 262 859)A04

EERC 76-13 "Coupled Lateral Torsional Response of Buildings to Ground Shaking," by C.L. Kan and A.K. Chopra - 1976 (PB 257 907)A09

EERC 76-14 "Seismic Analyses of the Banco de America," by V.V. Bertero, S.A. Mahin and J.A. Hollings - 1976

EERC 76-15 "Reinforced Concrete Frame 2: Seismic Testing and Analytical Correlation," by R.W. Clough and J. Gidwani - 1976 (PB 261 323)A08

EERC 76-16 "Cyclic Shear Tests of Masonry Piers, Volume 2 - Analysis of Test Results," by R.L. Mayes, Y. Omote and R.W. Clough - 1976

EERC 76-17 "Structural Steel Bracing Systems: Behavior Under Cyclic Loading," by E.P. Popov, K. Takanashi and C.W. Roeder - 1976 (PB 260 715)A05

EERC 76-18 "Experimental Model Studies on Seismic Response of High Curved Overcrossings," by D. Williams and W.G. Godden - 1976 (PB 269 548)A08

EERC 76-19 "Effects of Non-Uniform Seismic Disturbances on the Dumbarton Bridge Replacement Structure," by F. Baron and R.E. Hamati - 1976 (PB 282 981)A16

EERC 76-20 "Investigation of the Inelastic Characteristics of a Single Story Steel Structure Using System Identification and Shaking Table Experiments," by V.C. Matzen and H.D. McNiven - 1976 (PB 258 453)A07

EERC 76-21 "Capacity of Columns with Splice Imperfections," by E.P. Popov, R.M. Stephen and R. Philbrick - 1976 (PB 260 378)A04

EERC 76-22 "Response of the Olive View Hospital Main Building during the San Fernando Earthquake," by S. A. Mahin, V.V. Bertero, A.K. Chopra and R. Collins - 1976 (PB 271 425)A14

EERC 76-23 "A Study on the Major Factors Influencing the Strength of Masonry Prisms," by N.M. Mostaghel, R.L. Mayes, R. W. Clough and S.W. Chen - 1976 (Not published)

EERC 76-24 "GADFLEA - A Computer Program for the Analysis of Pore Pressure Generation and Dissipation during Cyclic or Earthquake Loading," by J.R. Booker, M.S. Rahman and H.B. Seed - 1976 (PB 263 947)A04

EERC 76-25 "Seismic Safety Evaluation of a R/C School Building," by B. Bresler and J. Axley - 1976

EERC 76-26 "Correlative Investigations on Theoretical and Experimental Dynamic Behavior of a Model Bridge Structure," by K. Kawashima and J. Penzien - 1976 (PB 263 388)A11

EERC 76-27 "Earthquake Response of Coupled Shear Wall Buildings," by T. Srichatrapimuk - 1976 (PB 265 157)A07

EERC 76-28 "Tensile Capacity of Partial Penetration Welds," by E.P. Popov and R.M. Stephen - 1976 (PB 262 899)A03

EERC 76-29 "Analysis and Design of Numerical Integration Methods in Structural Dynamics," by H.M. Hilber - 1976 (PB 264 410)A06

EERC 76-30 "Contribution of a Floor System to the Dynamic Characteristics of Reinforced Concrete Buildings," by L.E. Malik and V.V. Bertero - 1976 (PB 272 247)A13

EERC 76-31 "The Effects of Seismic Disturbances on the Golden Gate Bridge," by F. Baron, M. Arikan and R.E. Hamati - 1976 (PB 272 279)A09

EERC 76-32 "Infilled Frames in Earthquake Resistant Construction," by R.E. Klingner and V.V. Bertero - 1976 (PB 265 892)A13

UCB/EERC-77/01 "PLUSH - A Computer Program for Probabilistic Finite Element Analysis of Seismic Soil-Structure Interaction," by M.P. Romo Organista, J. Lysmer and H.B. Seed - 1977

UCB/EERC-77/02 "Soil-Structure Interaction Effects at the Humboldt Bay Power Plant in the Ferndale Earthquake of June 7, 1975," by J.E. Valera, H.B. Seed, C.F. Tsai and J. Lysmer - 1977 (PB 265 795)A04

UCB/EERC-77/03 "Influence of Sample Disturbance on Sand Response to Cyclic Loading," by K. Mori, H.B. Seed and C.K. Chan - 1977 (PB 267 352)A04

UCB/EERC-77/04 "Seismological Studies of Strong Motion Records," by J. Shoja-Taheri - 1977 (PB 269 655)A10

UCB/EERC-77/05 "Testing Facility for Coupled-Shear Walls," by L. Li-Hyung, V.V. Bertero and E.P. Popov - 1977

UCB/EERC-77/06 "Developing Methodologies for Evaluating the Earthquake Safety of Existing Buildings," by No. 1 - B. Bresler; No. 2 - B. Bresler, T. Okada and D. Zisling; No. 3 - T. Okada and B. Bresler; No. 4 - V.V. Bertero and B. Bresler - 1977 (PB 267 354)A08

UCB/EERC-77/07 "A Literature Survey - Transverse Strength of Masonry Walls," by Y. Omote, R.L. Mayes, S.W. Chen and R.W. Clough - 1977 (PB 277 933)A07

UCB/EERC-77/08 "DRAIN-TABS: A Computer Program for Inelastic Earthquake Response of Three Dimensional Buildings," by R. Guendelman-Israel and G.H. Powell - 1977 (PB 270 693)A07

UCB/EERC-77/09 "SUBWALL: A Special Purpose Finite Element Computer Program for Practical Elastic Analysis and Design of Structural Walls with Substructure Option," by D.Q. Le, H. Peterson and E.P. Popov - 1977 (PB 270 567)A05

UCB/EERC-77/10 "Experimental Evaluation of Seismic Design Methods for Broad Cylindrical Tanks," by D.P. Clough (PB 272 280)A13

UCB/EERC-77/11 "Earthquake Engineering Research at Berkeley - 1976," - 1977 (PB 273 507)A09

UCB/EERC-77/12 "Automated Design of Earthquake Resistant Multistory Steel Building Frames," by N.D. Walker, Jr. - 1977 (PB 276 526)A09

UCB/EERC-77/13 "Concrete Confined by Rectangular Hoops Subjected to Axial Loads," by J. Vallenas, V.V. Bertero and E.P. Popov - 1977 (PB 275 165)A06

UCB/EERC-77/14 "Seismic Strain Induced in the Ground During Earthquakes," by Y. Sugimura - 1977 (PB 284 201)A04

UCB/EERC-77/15 "Bond Deterioration under Generalized Loading," by V.V. Bertero, E.P. Popov and S. Viwathanatepa - 1977

UCB/EERC-77/16 "Computer Aided Optimum Design of Ductile Reinforced Concrete Moment Resisting Frames," by S.W. Zagajeski and V.V. Bertero - 1977 (PB 280 137)A07

UCB/EERC-77/17 "Earthquake Simulation Testing of a Stepping Frame with Energy-Absorbing Devices," by J.M. Kelly and D.F. Tsztoo - 1977 (PB 273 506)A04

UCB/EERC-77/18 "Inelastic Behavior of Eccentrically Braced Steel Frames under Cyclic Loadings," by C.W. Roeder and E.P. Popov - 1977 (PB 275 526)A15

UCB/EERC-77/19 "A Simplified Procedure for Estimating Earthquake-Induced Deformations in Dams and Embankments," by F.I. Makdisi and H.B. Seed - 1977 (PB 276 820)A04

UCB/EERC-77/20 "The Performance of Earth Dams during Earthquakes," by H.B. Seed, F.I. Makdisi and P. de Alba - 1977 (PB 276 821)A04

UCB/EERC-77/21 "Dynamic Plastic Analysis Using Stress Resultant Finite Element Formulation," by P. Lukkunapvasit and J.M. Kelly - 1977 (PB 275 453)A04

UCB/EERC-77/22 "Preliminary Experimental Study of Seismic Uplift of a Steel Frame," by R.W. Clough and A.A. Huckelbridge 1977 (PB 278 769)A08

UCB/EERC-77/23 "Earthquake Simulator Tests of a Nine-Story Steel Frame with Columns Allowed to Uplift," by A.A. Huckelbridge - 1977 (PB 277 944)A09

UCB/EERC-77/24 "Nonlinear Soil-Structure Interaction of Skew Highway Bridges," by M.-C. Chen and J. Penzien - 1977 (PB 276 176)A07

UCB/EERC-77/25 "Seismic Analysis of an Offshore Structure Supported on Pile Foundations," by D.D.-N. Liou and J. Penzien 1977 (PB 283 180)A06

UCB/EERC-77/26 "Dynamic Stiffness Matrices for Homogeneous Viscoelastic Half-Planes," by G. Dasgupta and A.K. Chopra - 1977 (PB 279 654)A06

UCB/EERC-77/27 "A Practical Soft Story Earthquake Isolation System," by J.M. Kelly, J.M. Eidinger and C.J. Derham - 1977 (PB 276 814)A07

UCB/EERC-77/28 "Seismic Safety of Existing Buildings and Incentives for Hazard Mitigation in San Francisco: An Exploratory Study," by A.J. Meltsner - 1977 (PB 281 970)A05

UCB/EERC-77/29 "Dynamic Analysis of Electrohydraulic Shaking Tables," by D. Rea, S. Abedi-Hayati and Y. Takahashi 1977 (PB 282 569)A04

UCB/EERC-77/30 "An Approach for Improving Seismic - Resistant Behavior of Reinforced Concrete Interior Joints," by B. Galunic, V.V. Bertero and E.P. Popov - 1977 (PB 290 870)A06

UCB/EERC-78/01 "The Development of Energy-Absorbing Devices for Aseismic Base Isolation Systems," by J.M. Kelly and D.F. Tsztoo - 1978 (PB 284 978)A04

UCB/EERC-78/02 "Effect of Tensile Prestrain on the Cyclic Response of Structural Steel Connections, by J.G. Bouwkamp and A. Mukhopadhyay - 1978

UCB/EERC-78/03 "Experimental Results of an Earthquake Isolation System using Natural Rubber Bearings," by J.M. Eidinger and J.M. Kelly - 1978 (PB 281 686)A04

UCB/EERC-78/04 "Seismic Behavior of Tall Liquid Storage Tanks," by A. Niwa - 1978 (PB 284 017)A14

UCB/EERC-78/05 "Hysteretic Behavior of Reinforced Concrete Columns Subjected to High Axial and Cyclic Shear Forces," by S.W. Zagajeski, V.V. Bertero and J.G. Bouwkamp - 1978 (PB 283 858)A13

UCB/EERC-78/06 "Inelastic Beam-Column Elements for the ANSR-I Program," by A. Riahi, D.G. Row and G.H. Powell - 1978

UCB/EERC-78/07 "Studies of Structural Response to Earthquake Ground Motion," by O.A. Lopez and A.K. Chopra - 1978 (PB 282 790)A05

UCB/EERC-78/08 "A Laboratory Study of the Fluid-Structure Interaction of Submerged Tanks and Caissons in Earthquakes," by R.C. Byrd - 1978 (PB 284 957)A08

UCB/EERC-78/09 "Model for Evaluating Damageability of Structures," by I. Sakamoto and B. Bresler - 1978

UCB/EERC-78/10 "Seismic Performance of Nonstructural and Secondary Structural Elements," by I. Sakamoto - 1978

UCB/EERC-78/11 "Mathematical Modelling of Hysteresis Loops for Reinforced Concrete Columns," by S. Nakata, T. Sproul and J. Penzien - 1978

UCB/EERC-78/12 "Damageability in Existing Buildings," by T. Blejwas and B. Bresler - 1978

UCB/EERC-78/13 "Dynamic Behavior of a Pedestal Base Multistory Building," by R.M. Stephen, E.L. Wilson, J.G. Bouwkamp and M. Button - 1978 (PB 286 650)A08

UCB/EERC-78/14 "Seismic Response of Bridges - Case Studies," by R.A. Imbsen, V. Nutt and J. Penzien - 1978 (PB 286 503)A10

UCB/EERC-78/15 "A Substructure Technique for Nonlinear Static and Dynamic Analysis," by D.G. Row and G.H. Powell - 1978 (PB 288 077)A10

UCB/EERC-78/16 "Seismic Risk Studies for San Francisco and for the Greater San Francisco Bay Area," by C.S. Oliveira - 1978

UCB/EERC-78/17 "Strength of Timber Roof Connections Subjected to Cyclic Loads," by P. Gülkan, R.L. Mayes and R.W. Clough - 1978

UCB/EERC-78/18 "Response of K-Braced Steel Frame Models to Lateral Loads," by J.G. Bouwkamp, R.M. Stephen and E.P. Popov - 1978

UCB/EERC-78/19 "Rational Design Methods for Light Equipment in Structures Subjected to Ground Motion," by J.L. Sackman and J.M. Kelly - 1978 (PB 292 357)A04

UCB/EERC-78/20 "Testing of a Wind Restraint for Aseismic Base Isolation," by J.M. Kelly and D.E. Chitty - 1978 (PB 292 833)A03

UCB/EERC-78/21 "APOLLO - A Computer Program for the Analysis of Pore Pressure Generation and Dissipation in Horizontal Sand Layers During Cyclic or Earthquake Loading," by P.P. Martin and H.B. Seed - 1978 (PB 292 835)A04

UCB/EERC-78/22 "Optimal Design of an Earthquake Isolation System," by M.A. Bhatti, K.S. Pister and E. Polak - 1978 (PB 294 735)A06

UCB/EERC-78/23 "MASH - A Computer Program for the Non-Linear Analysis of Vertically Propagating Shear Waves in Horizontally Layered Deposits," by P.P. Martin and H.B. Seed - 1978 (PB 293 101)A05

UCB/EERC-78/24 "Investigation of the Elastic Characteristics of a Three Story Steel Frame Using System Identification," by I. Kaya and H.D. McNiven - 1978

UCB/EERC-78/25 "Investigation of the Nonlinear Characteristics of a Three-Story Steel Frame Using System Identification," by I. Kaya and H.D. McNiven - 1978

UCB/EERC-78/26 "Studies of Strong Ground Motion in Taiwan," by Y.M. Hsiung, B.A. Bolt and J. Penzien - 1978

UCB/EERC-78/27 "Cyclic Loading Tests of Masonry Single Piers: Volume 1 - Height to Width Ratio of 2," by P.A. Hidalgo, R.L. Mayes, H.D. McNiven and R.W. Clough - 1978

UCB/EERC-78/28 "Cyclic Loading Tests of Masonry Single Piers: Volume 2 - Height to Width Ratio of 1," by S.-W.J. Chen, P.A. Hidalgo, R.L. Mayes, R.W. Clough and H.D. McNiven - 1978

UCB/EERC-78/29 "Analytical Procedures in Soil Dynamics," by J. Lysmer - 1978

UCB/EERC-79/01 "Hysteretic Behavior of Lightweight Reinforced Concrete Beam-Column Subassemblages," by B. Forzani, E.P. Popov, and V.V. Bertero - 1979

UCB/EERC-79/02 "The Development of a Mathematical Model to Predict the Flexural Response of Reinforced Concrete Beams to Cyclic Loads, Using System Identification," by J.F. Stanton and H.D. McNiven - 1979

UCB/EERC-79/03 "Linear and Nonlinear Earthquake Response of Simple Torsionally Coupled Systems," by C.L. Kan and A.K. Chopra - 1979

UCB/EERC-79/04 "A Mathematical Model of Masonry for Predicting Its Linear Seismic Response Characteristics," by Y. Mengi and H.D. McNiven - 1979

UCB/EERC-79/05 "Mechanical Behavior of Light Weight Concrete Confined by Different Types of Lateral Reinforcement," by M.A. Manrique, V.V. Bertero and E.P. Popov - 1979

UCB/EERC-79/06 "Static Tilt Tests of a Tall Cylindrical Liquid Storage Tank," by R.W. Clough and A. Niwa - 1979

UCB/EERC-79/07 "The Design of Steel Energy Absorbing Restrainers and Their Incorporation Into Nuclear Power Plants for Enhanced Safety: Volume 1 - Summary Report," by P.N. Spencer, V.F. Zackay, and E.R. Parker - 1979

UCB/EERC-79/08 "The Design of Steel Energy Absorbing Restrainers and Their Incorporation Into Nuclear Power Plants for Enhanced Safety: Volume 2 - The Development of Analyses for Reactor System Piping," "Simple Systems" by M.C. Lee, J. Penzien, A.K. Chopra, and K. Suzuki "Complex Systems" by G.H. Powell, E.L. Wilson, R.W. Clough and D.G. Row - 1979

UCB/EERC-79/09 "The Design of Steel Energy Absorbing Restrainers and Their Incorporation Into Nuclear Power Plants for Enhanced Safety: Volume 3 - Evaluation of Commerical Steels," by W.S. Owen, R.M.N. Pelloux, R.O. Ritchie, M. Faral, T. Ohhashi, J. Toplosky, S.J. Hartman, V.F. Zackay, and E.R. Parker - 1979

UCB/EERC-79/10 "The Design of Steel Energy Absorbing Restrainers and Their Incorporation Into Nuclear Power Plants for Enhanced Safety: Volume 4 - A Review of Energy-Absorbing Devices," by J.M. Kelly and M.S. Skinner - 1979

UCB/EERC-79/11 "Conservatism In Summation Rules for Closely Spaced Modes," by J.M. Kelly and J.L. Sackman - 1979

UCB/EERC-79/12 "Cyclic Loading Tests of Masonry Single Piers Volume 3 - Height to Width Ratio of 0.5," by P.A. Hidalgo, R.L. Mayes, H.D. McNiven and R.W. Clough - 1979

UCB/EERC-79/13 "Cyclic Behavior of Dense Coarse-Grain Materials in Relation to the Seismic Stability of Dams," by N.G. Banerjee, H.B. Seed and C.K. Chan - 1979

UCB/EERC-79/14 "Seismic Behavior of R/C Interior Beam Column Subassemblages," by S. Viwathanatepa, E.P. Popov and V.V. Bertero - 1979

UCB/EERC-79/15 "Optimal Design of Localized Nonlinear Systems with Dual Performance Criteria Under Earthquake Excitations," by M.A. Bhatti - 1979

UCB/EERC-79/16 "OPTDYN - A General Purpose Optimization Program for Problems with or without Dynamic Constraints," by M.A. Bhatti, E. Polak and K.S. Pister - 1979