INTERNAL REPORTS IN

SIMULATION, OPTIMIZATION AND CONTROL

No. SOC-29

DISOPT - A GENERAL PROGRAM FOR CONTINUOUS AND DISCRETE NONLINEAR PROGRAMMING PROBLEMS

J.H.K. Chen

March 1974

(Revised June 1975)

FACULTY OF ENGINEERING
McMASTER UNIVERSITY
HAMILTON, ONTARIO, CANADA

AVAILABILITY OF REPORTS

In addition to extensive publications by G-SOC members, a series of reports covering simulation, optimization and control topics is published by the group. Preprints or extended versions of papers, reprints of papers appearing in conference proceedings, fully documented computer program descriptions including listings, theses, notes and manuals appear as reports.

A free list of SOC reports including numbers, titles, authors, dates of publication, and indication of the inclusion of computer listings is available on request. To offset preparation, printing and distribution costs the charges as noted must be made*.

Any number of these reports may be ordered**. Cheques should be made out in U.S. or Canadian Dollars and made payable to McMaster University. Requests must be addressed to:

Dr. J.W. Bandler
Coordinator, G-SOC
Faculty of Engineering
McMaster University
Hamilton, Canada L8S 4L7

Reports will not normally be sent out until payment is received.

Some reports may be temporarily restricted for internal use only. Some may be revised or ultimately superceded. Availability, descriptions or charges are subject to change without notice.

Typical of the 158 reports published up to January 1977 are:

SOC-29	DISOPT - A General Program for Continuous and Discrete Nonlinear Programming Problems***	Mar. 1974	80 pp.	$30
SOC-43	Convergence Acceleration in the Numerical Solution of Field Problems	Jun. 1974	324 pp.	$30
SOC-61	Canonical Forms for Linear Multivariable Systems	Oct. 1974	29 pp.	$5
SOC-82	Optimal Choice of the Sampling Interval for Discrete Process Control	Mar. 1975	41 pp.	$5
SOC-113	Notes on Numerical Methods of Optimization with Applications in Optimal Design***	Nov. 1975	396 pp.	$150
SOC-151	FLOPT4-A Program for Least pth Optimization with Extrapolation to Minimax Solutions***	Jan. 1977	97 pp.	$60

* Subscriptions and discounts are available.
** Special reduced rates will be quoted for multiple copies.
*** Include FORTRAN listings.

DISOPT - A GENERAL PROGRAM FOR CONTINUOUS AND DISCRETE NONLINEAR PROGRAMMING PROBLEMS

(Revised June 1975)

J.H.K. Chen

NOTE ON REVISED REPORT

This work originally appeared as an M.Eng. thesis by J.H.K. Chen. The June 1975 revised version has a new Appendix, a new program listing and some minor modifications or corrections in the remainder of the text to reflect the changes in the program. Minor updating of the references has also been carried out. The paper by the same title to be published in the International Journal of Systems Science is based on the original thesis.

The assistance of W.Y. Chu is gratefully acknowledged.

J.W. Bandler

June 1975

ABSTRACT

An integrated computer program in FORTRAN IV for continuous or discrete, constrained or unconstrained general optimization problems is presented. The program, called DISOPT, is applicable to a wide variety of design problems such as continuous and discrete tolerance assignments, digital filter design, circuit design, system modelling and approximation problems. Many recent techniques and algorithms for nonlinear programming have been incorporated. The user may optionally choose the combination of techniques and algorithms best suited to his problems.

ACKNOWLEDGEMENTS

The author is greatly indebted to Dr. J.W. Bandler for his guidance and encouragement throughout the course of this work. Special thanks are due to P.C. Liu who proofread the program and W.Y. Chu whose implementation of extrapolation was incorporated into DISOPT. The author also wishes to thank B.L. Bardakjian, Dr. C. Charalambous, J.R. Popović and Dr. T.V. Srinivasan for their useful suggestions.

Financial support provided by the National Research Council of Canada through grant A7239 and through the award of an NRC scholarship is gratefully acknowledged.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Optimization has become an almost indispensable step in engineering design. Many useful algorithms and techniques for optimization have been proposed. However, it would be very time-consuming and inconvenient for each individual engineer to implement these algorithms and techniques to solve his particular design problem. The objective of this report is to present an efficient, user-oriented computer program called DISOPT, in FORTRAN IV, which can solve continuous or discrete, constrained or unconstrained general optimization problems. Many recently proposed algorithms and techniques which have been reported to be efficient have been programmed into DISOPT. To the author's knowledge, it is the first time that many of these algorithms and techniques are incorporated in a general program. Several new ideas have also been introduced which allow the user to fully employ some of the latest developments.

Chapter II describes the two approaches to nonlinear programming incorporated in DISOPT. The first approach is the minimax approach proposed by Bandler and Charalambous [1]. Previous tests have shown this method to be at least comparable to, if not better than, the well-regarded sequential unconstrained minimization technique [2]. For the implementation of this minimax approach, in addition to adapting the various least pth optimization algorithms due to Bandler and Charalambous, a new algorithm utilizing an extrapolation technique is developed. With all the attempted problems, this last algorithm was found to converge to the minimax optimum faster than the others. The second approach to nonlinear programming is a modification of an existing non-parametric exterior-point method described by Lootsma [3]. Some examples have been included to demonstrate the performance of the methods.

The solution for discrete nonlinear programming problems is described in Chapter III. Recently, much attention has been directed to discrete optimization. The reason is obvious since, in practice, a discrete solution is more realistic than a continuous solution. For example, in practical network design problems, a compromise between maximum performance and minimum cost is often necessary because usually only components of certain discrete values are available on the market. Components of other values have to be custom-made and are therefore costly. The logic of the Dakin tree-search algorithm for integer programming [4] is followed but many modifications have been embodied in DISOPT to enhance the efficiency of the algorithm. Some of them are:

(1) reduction of the dimensionality of the problem,

(2) evaluation of an initial upper bound on the function value,

(3) checking the existence of a feasible solution and

(4) determination of the availability of a better solution after a discrete solution is obtained.

The algorithm has also been generalized to handle discrete problem of uniform as well as nonuniform quantization step sizes. Several illustrative examples are given.

The latest version of the variable metric algorithm due to Fletcher [5] is employed to perform the minimization. The formulation of the required derivatives may be optionally checked by DISOPT using numerical perturbation. A flow diagram and a complete FORTRAN listing of DISOPT together with the documentation for the user are given in the Appendix.

CHAPTER II

THE CONTINUOUS OPTIMIZATION ALGORITHMS

2.1 Introduction

Consider the nonlinear programming problem of minimizing

$$f \triangleq f(\boldsymbol{\phi})$$

subject to

$$g_i(\boldsymbol{\phi}) \geq 0 \ , \ i = 1, 2, \ldots, m$$

where f is the objective function, the vector $\boldsymbol{\phi}$ represents a set of k variables

$$\boldsymbol{\phi} \triangleq [\phi_1 \ \phi_2 \ \cdots \ \phi_k]^T$$

and $g_1(\boldsymbol{\phi})$, $g_2(\boldsymbol{\phi})$, ..., $g_m(\boldsymbol{\phi})$ are the constraint functions. Both f and the g_i's are, in general, nonlinear differentiable functions of the variables.

In order that efficient gradient minimization algorithms for unconstrained functions, such as the variable metric algorithm due to Fletcher [5], may be employed, the nonlinear programming problem has to be transformed into an equivalent unconstrained objective. The two transformation methods used in DISOPT will be described next.

2.2 Bandler-Charalambous Technique [1]

The nonlinear programming problem is transformed into the following unconstrained objective

$$V(\boldsymbol{\phi},\alpha) = \max_{1 \leq i \leq m} \left[f(\boldsymbol{\phi}),\ f(\boldsymbol{\phi}) - \alpha g_i(\boldsymbol{\phi})\right]$$

where

$$\alpha > 0.$$

Sufficiently large α must be chosen to satisfy the inequality

$$\frac{1}{\alpha} \sum_{i=1}^{m} u_i < 1$$

where the u_i's are the Kuhn-Tucker multipliers at the optimum.

The minimization of $V(\phi,\alpha)$ with respect to ϕ is a minimax problem and may be implemented by one of the several recent least pth optimization algorithms proposed by Bandler and Charalambous [6] - [9].

Let

$$e_i(\phi) \triangleq f(\phi) - \alpha g_i(\phi) \quad , \; i = 1, 2, \ldots, m$$

$$e_{m+1}(\phi) \triangleq f(\phi)$$

$$M_e(\phi) \triangleq \max_{1 \leq j \leq m+1} e_j(\phi)$$

Algorithm 1: Nonlinear minimax optimization as a least pth optimization with a large value of p.

Minimize with respect to ϕ the function

$$U(\phi) = (M_e(\phi)-\varepsilon) \left(\sum_{j \varepsilon J} \left(\frac{e_j(\phi)-\varepsilon}{M_e(\phi)-\varepsilon} \right)^q \right)^{1/q}$$

where

$$\varepsilon = \begin{cases} 0 \text{ for } M_e(\phi) \neq 0 \\ \text{small positive number for } M_e(\phi) = 0 \end{cases}$$

$$q = p \text{ sign } (M_e(\phi)-\varepsilon)$$

and

$$\text{if } M_e(\phi) \begin{cases} >0, \; 1<p<\infty, \; J=\{j \,|\, e_j(\phi)>0, \; j=1,2,\ldots,m+1\} \\ \leq 0, \; 1 \leq p<\infty, \; J=\{1,2,\ldots,m+1\} \end{cases}$$

By employing a sufficiently large value of p, the minimization yields, for all practical purposes, a minimax solution.

Algorithm 2: Nonlinear minimax optimization as a sequence of least pth optimization with increasing values of p.

$U(\phi)$ is defined as in algorithm 1 and minimized using increasing values of p. The optimum of each minimization is used as the starting point of the following minimization. The process is terminated if the relative decrease in $M_e(\check{\phi}^r)$, where $\check{\phi}^r$ is the optimum of the rth minimization, between two consecutive minimizations is less than a preset small positive quantity or after p has reached the maximum assigned value.

Algorithm 3: Application of an extrapolation technique to a sequence of least pth optimizations with geometrically increasing values of p.

The basic formulation is the same as in the two previous algorithms. $U(\phi)$ is minimized with geometrically increasing values of p, i.e., $p^r = p^1 c^{r-1}$* where p^r is the value of p used in the rth optimization and c is the multiplying factor. The optimum of each minimization is a function of p or 1/p. The minimax solution is obtained as $p \to \infty$ or $1/p \to 0$.

Fiacco and McCormick [2] have applied an extrapolation technique effectively to the SUMT constraint transformation algorithm. Since the situation here is analogous, it is felt that the convergence to the minimax solution may be improved by utilizing the same extrapolation technique. After each minimization, the extrapolation formula proposed by Fiacco and McCormick is applied to estimate the minimax optimum, $\check{\phi}$, and the optimum of the next optimization.

*c^{r-1} means c raised to the power r-1.

Let ϕ_j^i, i=1, 2,..., r, j=0,1,..., i-1 signify the jth order estimate of $\check{\phi}$ after i minima have been achieved, then

$$\phi_0^i = \check{\phi}^i, \quad i=1,2,\ldots,r$$

where $\check{\phi}^i$ is the optimum of the ith optimization and

$$\phi_j^i = \frac{c^j \phi_{j-1}^i - \phi_{j-1}^{i-1}}{c^j - 1}, \quad \begin{array}{l} i = 2,3,\ldots,r \\ j = 1,2,\ldots,i-1 \end{array}$$

The estimate of $\check{\phi}$ is given by

$$\check{\phi} = \phi_{r-1}^r$$

To estimate $\check{\phi}^{r+1}$, the recursive relation

$$\phi_{j-1}^{r+1} = \frac{(c^j - 1)\phi_j^{r+1} + \phi_{j-1}^r}{c^j}$$

is used and

$$\check{\phi}^{r+1} = \phi_0^{r+1}$$

The process stops if the absolute difference between the estimate of $\check{\phi}$ in two consecutive optimizations is less than a prescribed k-tuple $(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_k)$ where the elements are small positive numbers, or if the maximum allowable number of optimizations is exceeded.

Algorithm 4: Nonlinear minimax optimization as a sequence of least pth optimization with finite values of p.

(1) Define $\xi^1 = \min\, [0, M_e(\phi^0) + \gamma]$

where ϕ^0 is the starting point and γ is a small positive number.

(2) Set r = 1.

(3) Minimize with respect to ϕ the function

$$U_\xi(\phi, \xi^r) = (M_\xi(\phi, \xi^r) - \varepsilon)\left(\sum_{j\varepsilon J}\left(\frac{e_j(\phi) - \xi^r - \varepsilon}{M_\xi(\phi, \xi^r) - \varepsilon}\right)^q\right)^{1/q}$$

where

$$M_\xi(\phi,\xi^r) = M_e(\phi)-\xi^r$$

$$\varepsilon = \begin{cases} 0 \text{ for } M_\xi(\phi,\xi^r) \neq 0 \\ \text{small positive number for } M_\xi(\phi,\xi^r) = 0 \end{cases}$$

$$q = p \text{ sign } M_\xi(\phi,\xi^r)$$

and

$$\text{if } M_\xi(\phi,\xi^r) \begin{cases} >0, \text{ then } 1<p<\infty,\ J=\{j\,|\,e_j(\phi)\geq\xi^r,\ j=1,2,\ldots,m+1\} \\ \leq 0, \text{ then } 1\leq p<\infty,\ J=\{1,2,\ldots,m+1\}. \end{cases}$$

(4) Set $\xi^{r+1} = M_e(\check{\phi}^r) + \gamma$.

(5) If $|(\xi^{r+1}-\xi^r)/\xi^r| < \eta$, where η is a suitable small positive number, stop. Otherwise, set $r = r + 1$.

(6) Go to step (3).

In this algorithm, any finite value of p greater than unity can be used to produce minimax solutions.

2.3 A Modified Non-Parametric Exterior-Point Method [3]

The nonlinear programming problem is transformed into an equivalent least pth objective and implemented as follows.

Algorithm 5.

(1) Let t^1 be the initial optimistic estimate of $f(\check{\phi})$, i.e., $t^1 \leq f(\check{\phi})$.

(2) Set $r = 1$.

(3) Minimize with respect to ϕ, the function

$$U_t(\phi,t^r)=M_t(\phi,t^r)\left(\left(\frac{f(\phi)-t^r}{M_t(\phi,t^r)}\right)^p + \sum_{j\varepsilon J}\left(\frac{-g_j(\phi)}{M_t(\phi,t^r)}\right)^p\right)^{1/p}$$

where

$$M_t(\phi, t^r) = \max_{j \varepsilon J} [f(\phi) - t^r, \ -g_j(\phi)]$$

$$J = \{j | g_j < 0\}$$

and

$$1 < p < \infty.$$

(4) Set $t^{r+1} = t^r + U_t(\check{\phi}^r, t^r)$.

(5) If $|(t^{r+1} - t^r)/t^r| < \eta$, where η is a small positive number, stop. Otherwise, set $r = r + 1$.

(6) Go to step (3).

Theorem 1. If $t^r \leq f(\check{\phi})$, then $t^{r+1} \leq f(\check{\phi})$.

Proof: By definition of $\check{\phi}^r$,

$$U_t(\check{\phi}^r, t^r) \leq U_t(\check{\phi}, t^r)$$

$$= M_t(\check{\phi}, t^r)\left(\left(\frac{f(\check{\phi}) - t^r}{M_t(\check{\phi}, t^r)}\right)^p + \sum_{j \varepsilon J} \left(\frac{-g_j(\check{\phi})}{M_t(\check{\phi}, t^r)}\right)^p\right)^{1/p}$$

$$= M_t(\check{\phi}, t^r)\left(\left(\frac{f(\check{\phi}) - t^r}{M_t(\check{\phi}, t^r)}\right)^p\right)^{1/p}$$

(since $g_i(\check{\phi}) \geq 0$, $i = 1, 2, \ldots, m$, by definition)

$$= f(\check{\phi}) - t^r.$$

This implies that

$$U_t(\check{\phi}^r, t^r) + t^r \leq f(\check{\phi})$$

$$t^{r+1} \leq f(\check{\phi})$$

Theorem 2. If t^r is an exact estimate, i.e., $t^r = f(\check{\phi})$ then a solution of $U_t(\phi, t^r)$ is a solution of the nonlinear programming problem and vice versa.

Proof: $U_t(\check{\phi}^r, t^r) \leq U_t(\check{\phi}, t^r)$

$$= 0$$

since $f(\check{\phi}) = t^r$ and $g_i(\check{\phi}) \geq 0$, $i=1,2,\ldots,m$.

But

$$U_t(\phi,t^r) \geq 0.$$

Hence

$$U_t(\check{\phi}^r,t^r) = 0.$$

This implies that

$$f(\check{\phi}^r) = t^r = f(\check{\phi})$$

and

$$g_i(\check{\phi}^r) \geq 0 , \quad i=1,2,\ldots,m.$$

Thus $\check{\phi}^r$ is a solution of the nonlinear programming problem.

Conversely,

$$U_t(\check{\phi},t^r) = 0$$
$$\leq U_t(\phi,t^r).$$

Thus, $\check{\phi}$ is a solution of $U_t(\phi,t^r)$.

2.4 Numerical Examples

Two test functions and a continuous tolerance assignment problem were used to illustrate the performance of the aforementioned algorithms. All the programs were run on a CDC6400 computer.

Example 1: Beale constrained function [10]

Minimize

$$f(\phi) = 9-8\phi_1-6\phi_2-4\phi_3+2\phi_1^2+2\phi_2^2+\phi_3^2+2\phi_1\phi_2+2\phi_1\phi_3$$

subject to

$$\phi_i \geq 0, \quad i=1,2,3$$

$$3-\phi_1-\phi_2-2\phi_3 \geq 0.$$

The function has a minimum $f(\check{\phi}) = 1/9$ at $\check{\phi} = [4/3\ 7/9\ 4/9]^T$. The numerical results from a nonfeasible starting point are tabulated in Table 2.1.

Example 2: Rosen-Suzuki function [10]

Minimize

$$f(\phi) = \phi_1^2+\phi_2^2+2\phi_3^2+\phi_4^2-5\phi_1-5\phi_2-21\phi_3+7\phi_4$$

subject to

$$-\phi_1^2-\phi_2^2-\phi_3^2-\phi_4^2-\phi_1+\phi_2-\phi_3+\phi_4+8 \geq 0$$

$$-\phi_1^2-2\phi_2^2-\phi_3^2-2\phi_4^2+\phi_1+\phi_4+10 \geq 0$$

$$-2\phi_1^2-\phi_2^2-\phi_3^2-2\phi_1+\phi_2+\phi_4+5 \geq 0.$$

The function has a minimum $f(\check{\phi}) = -44$ at $\check{\phi} = [0\ 1\ 2\ -1]^T$. Table 2.2 shows the performance of the five algorithms from a feasible starting point.

Example 3: Tolerance assignment in the design of a lowpass filter [11]-[12].

Consider the lowpass filter shown in Figure 2.1. Minimize the cost function

$$f = \sum_{i=1}^{3} \frac{1}{\phi_i}$$

where ϕ_i is the percentage tolerance of component ϕ_{i+3}.

Let Γ denote the insertion loss. The passband and stopband specifications are given by

$$\Gamma(\phi,\omega) \leq 1.5 \text{ dB for } 0 \leq \omega \leq 1 \text{ rad/sec}$$

and

$$\Gamma(\phi,\omega) \geq 25 \text{ dB for } \omega \geq 2.5 \text{ rad/sec},$$

Algorithms	1	2	3	4	5
p value(s)	10^5	$10, 10^5$	4,16,64,256	10	1.5
Other parameter value(s)	$\alpha = 1$	$\alpha = 1$	$\alpha = 1$ Order of extrapolation = 3	$\alpha = 1$	$t^1 = 0$
ϕ_1	1.3333338	1.3333338	1.3333333	1.3333353	1.3333333
ϕ_2	0.7777775	0.7777772	0.7777778	0.7777776	0.7777778
ϕ_3	0.4444437	0.4444438	0.4444444	0.4444436	0.4444444
$f(\underset{\sim}{\phi})$	0.1111114	0.1111114	0.1111111	0.1111111	0.1111111
Number of function evaluations	79	57	45	57	63

Table 2.1 Comparison of continuous optimization algorithms on Beale function for starting point $\underset{\sim}{\phi}^0 = [1\ 2\ 1]^T$.

Algorithms	1	2	3	4	5
p value(s)	10^5	$10, 10^3, 10^5$	4, 16, 64, 256, 1024	10^2	1.5
Other parameter value(s)	$\alpha = 10$	$\alpha = 10$	$\alpha = 10$ Order of extrapolation = 4	$\alpha = 10$	$t^1 = -50$
ϕ_1	-0.0000021	0.0000021	-0.0000014	0.0000080	0.0000071
ϕ_2	0.9999976	0.9999976	1.0000045	0.9999996	1.0000033
ϕ_3	1.9999908	1.9999908	1.9999985	2.0000043	2.0000079
ϕ_4	-0.9999883	-0.9999883	-1.0000031	-0.9999653	-0.9999848
$f(\underset{\sim}{\phi})$	-43.9998041	-43.9998041	-44.0000025	-43.9999210	-44.0000720
Number of function evaluations	110	90	74	90	300

Table 2.2 Comparison of continuous optimization algorithms on Rosen-Suzuki function for starting point $\underset{\sim}{\phi}^0 = [0\ 0\ 0\ 0]^T$.

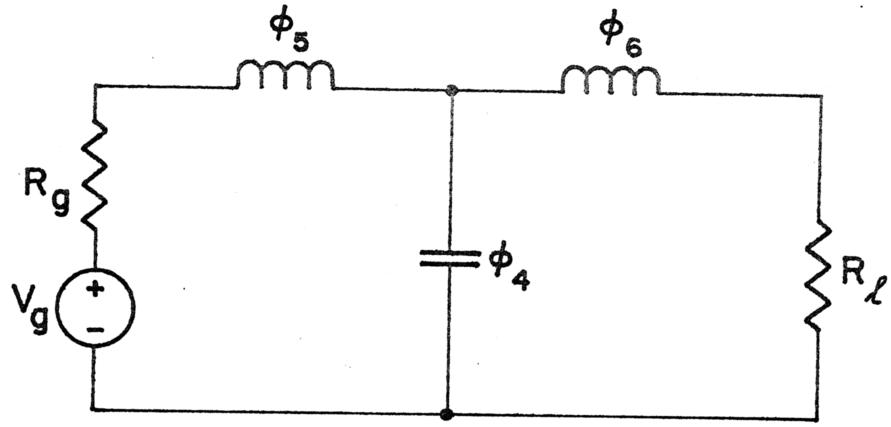

Figure 2.1 LC lowpass filter used in a tolerance assignment problem.

respectively. A set Ω of five sampling frequency points, namely,

$$\Omega = \{0.50, 0.55, 0.60, 1.00, 2.50\} \text{ rad/sec}$$

were chosen. Minimization was started from a nonfeasible point and the results are shown in Table 2.3.

2.5 Discussion

The efficiency of the algorithms depends heavily on the values of p and other parameters used as well as the starting point $\underset{\sim}{\phi}^0$, hence, it is very difficult to draw a definite conclusion regarding their performance.

Algorithm 5 appears to be quite efficient when a close initial optimistic estimate, t^1, of $f(\check{\underset{\sim}{\phi}})$ is available.

The numerical examples indicate algorithm 3 to be consistently superior to algorithms 1, 2 and 4. Convergence to the minimax solution is much improved by the application of extrapolation.

The use of a single large value of p in algorithm 1 may result in poor scaling, hence, slow convergence if the starting point, $\underset{\sim}{\phi}^0$, is not in the vicinity of $\check{\underset{\sim}{\phi}}$.

2.6 Existence of a Feasible Solution

If the constraints cannot be satisfied at the optimum of the least pth objective with any value of p greater than unity, then no feasible solution is attainable for all permissible values of p [7]. The existence of a feasible solution may be optionally checked by DISOPT before solving the nonlinear programming problem. DISOPT minimizes with a small value of p the function

Algorithm	1	2	3	4	5
p value(s)	10^5	$10, 10^3, 10^5$	4,16,64,256	4	1.5
Other parameter value(s)	$\alpha = 100$	$\alpha = 100$	$\alpha = 100$ Order of extrapolation = 3	$\alpha = 100$	$t^1 = 0$
ϕ_1	7.58638	7.60603	7.60599	7.60604	7.60600
ϕ_2	9.87321	9.89770	9.89772	9.89771	9.89778
ϕ_3	9.86777	9.89771	9.89772	9.89771	9.89778
ϕ_4	0.90573	0.90564	0.90564	0.90564	0.90563
ϕ_5	2.04267	1.99923	1.99923	1.99923	1.99923
ϕ_6	1.95586	1.99923	1.99923	1.99923	1.99923
$f(\boldsymbol{\phi})$	0.33444	0.33354	0.33354	0.33354	0.33354
Number of function evaluations	700*	425	337	478	157

*700 is the maximum allowable number of function evaluations.

Table 2.3 Comparison of continuous optimization algorithms on LC lowpass filter tolerance assignment problem for starting point $\boldsymbol{\phi}^0 = [5\ 5\ 5\ 1\ 1\ 1]^T$.

$$U_g(\phi) = M_g(\phi) \left(\sum_{j \varepsilon J} \left(\frac{-g_j(\phi)}{M_g(\phi)} \right)^p \right)^{1/p}$$

where

$$M_g(\phi) = \max_{j \varepsilon J} [-g_j(\phi)]$$

$$J = \{j \mid g_j(\phi) \leq 0, \ j=1,2,\ldots,m\}$$

The minimization terminates if $M_g(\phi) \leq 0$. A nonpositive value of $M_g(\phi)$ at the minimum or even before the minimum is reached indicates that a feasible solution is perceivable. Otherwise, there is no feasible solution to the problem with the current set of constraints.

CHAPTER III

THE DISCRETE OPTIMIZATION ALGORITHM

3.1 Introduction

When some or all of the variables in an otherwise nonlinear programming problem are further restricted to take on only certain discrete values, a discrete programming problem results. The special case of integer programming problems will be considered first.

3.2 Integer Programming

One trivial approach to integer programming would be to evaluate the function $f(\underset{\sim}{\phi})$ at all integer combinations $\underset{\sim}{\phi}$ satisfying the given constraints. In most practical situations, because of the vast number of such possible combinations, this exhaustive enumeration technique would be computationally disastrous. Random search techniques may greatly reduce the amount of computation required but there is no guarantee that the optimum integer solution would be obtained.

Although chopping or rounding off of a continuous solution to the nearest set of feasible integers may sometimes yield an excellent approximation of the optimum integer solution when the solution values are large numbers, it does not, in general, provide an accurate solution as illustrated in Figure 3.1.

The need for a systematic procedure which will identify the

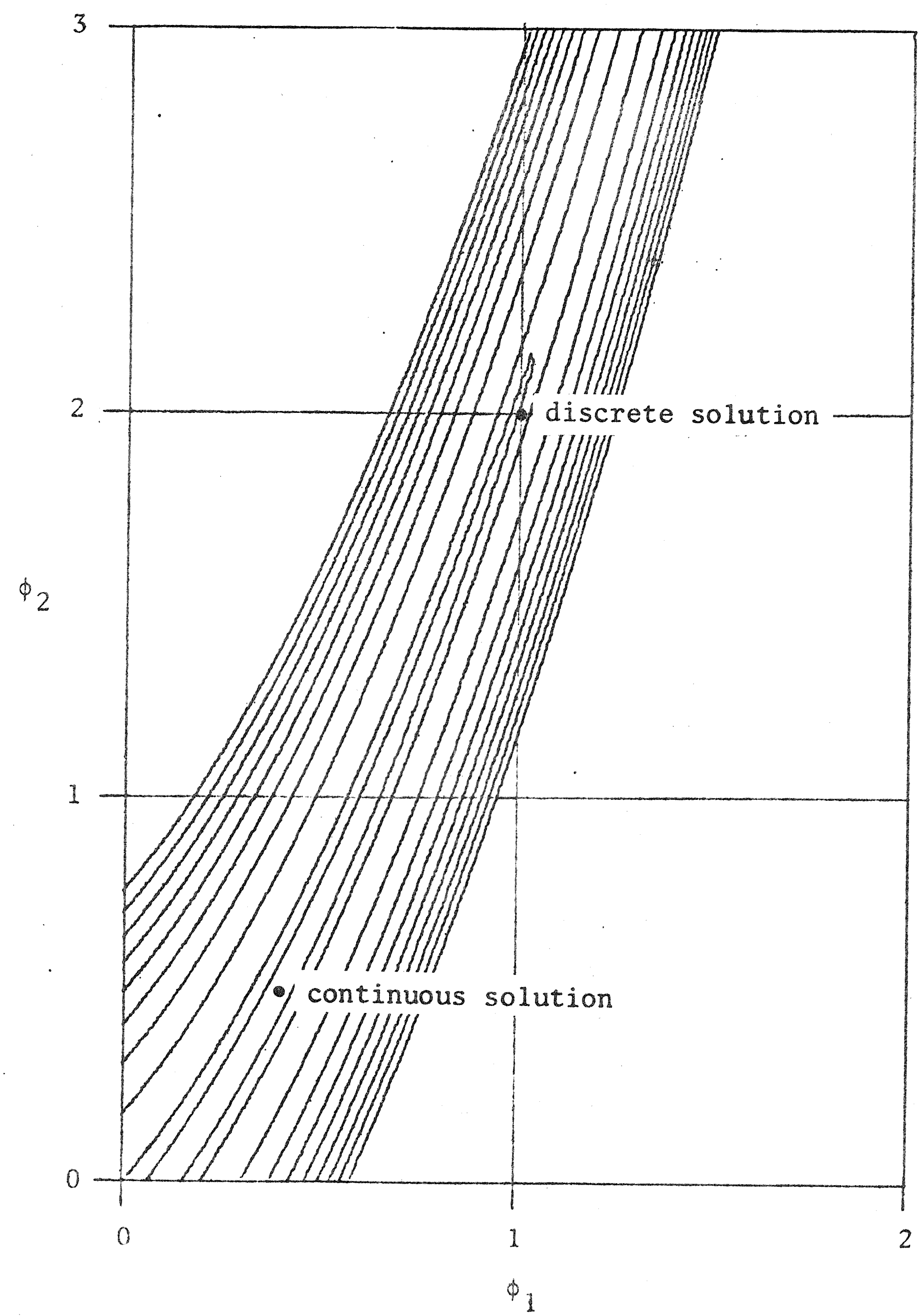

Figure 3.1 Contour plot for the modified banana shape function.

optimum integer solution efficiently is thus apparent. Various algorithms are available [13]. If the function $f(\underset{\sim}{\phi})$ can be expressed as a polynomial, then the lexicographic enumeration algorithm or the pseudo-boolean programming algorithm may be employed. However, for a general integer programming problem, the branch and bound or tree-search technique appears to be most attractive at present.

The branch and bound technique was first proposed by Land and Doig [14] and later modified by Dakin [4]. The solution of a discrete programming problem by DISOPT follows the logic of this latter approach.

3.3 Dakin's Tree-Search Algorithm

The algorithm first finds a solution to the continuous problem. If this solution happens to be integral, the integer problem is solved. If it is not, then at least one of the integer variables, e.g., ϕ_i, is non-integral and assumes a value $\phi_i{}^*$, say, in this solution. The range

$$[\phi_i{}^*] < \phi_i < [\phi_i{}^*] + 1$$

where $[\phi_i{}^*]$ is the largest integer value included in $\phi_i{}^*$, is inadmissible and consequently we may divide all solutions to the given problem into two non-overlapping groups, namely,

(1) solutions in which

$$\phi_i \leq [\phi_i{}^*]$$

(2) solutions in which

$$\phi_i \geq [\phi_i{}^*] + 1$$

Each of the constraints is added to the continuous problem sequentially and the corresponding augmented problems are solved. The procedure is repeated for each of the two solutions so obtained. Each resulting nonlinear programming problem thus constitutes a node and from each node two branches may emanate. A node will be fathomed if the following happens:

(1) the solution is integral

(2) no feasible solution for the current set of constraints is achievable

(3) the current optimum solution is worse than the best integer solution obtained so far.

The search stops when all the nodes are fathomed.

It seems, then, that the most efficient way of searching would be to branch, at each stage, from the node with the lowest $f(\underset{\sim}{\phi})$ value. This would minimize the searching of unlikely subtrees. To do this, all information about a node has to be retained for comparison and this may require cumbersome housekeeping and excessive storage for computer implementation. One way of compromising is to search the tree in an orderly manner; each branch is followed until it is fathomed.

The tree is not, in general, unique for a given problem. The tree structure depends on the order of partitioning on the discrete variables used. The amount of computation may be vastly different for different trees.

3.4 Discrete Programming

For the case of discrete programming problems subject to uniform quantization step sizes, the Dakin algorithm is modified as follows. Let ϕ_i be the discrete variable which assumes a non-discrete solution, ϕ_i^*, and q_i be the corresponding quantization step, then the two variable constraints added sequentially after each node become

$$\phi_i \geq [\phi_i^*/q_i]q_i + q_i$$

and

$$\phi_i \leq [\phi_i^*/q_i]q_i$$

The integer problem is thus a special case of the discrete problem with $q_i = 1$, $i = 1,2,\ldots,n$, where n is the number of discrete variables.

If, however, a finite set of discrete values given by

$$S_i = \{s_1, s_2, \ldots, s_j, s_{j+1}, \ldots, s_d\}, \quad i = 1,2,\ldots,n$$

is imposed upon each of the discrete variables, the variable constraints are then added according to the following rules:

(1) if $s_j < \phi_i^* < s_{j+1}$, then add the two constraints

$$\phi_i \leq s_j$$

and

$$\phi_i \geq s_{j+1}$$

sequentially

(2) if $\phi_i^* < s_1$, only add the constraint

$$\phi_i \geq s_1$$

(3) if $\phi_i^* > s_d$, only add the constraint

$$\phi_i \leq s_d$$

The resulting nonlinear programming problem at each node is solved by one of the algorithms described in Chapter II in conjunction with the Fletcher unconstrained minimization program. The feasibility checking mentioned in Section 2.6 is particularly useful here since the additional variable constraints may conflict with the original constraints on the continuous problem. If an upper bound, $\bar{f}$, on $f(\underset{\sim}{\phi})$ is available, then the additional constraint

$$f(\underset{\sim}{\phi}) \leq \bar{f}$$

is included in the feasibility checking. This upper bound, if not specified, will be taken as the current best discrete solution. To obtain an initial upper bound on $f(\underset{\sim}{\phi})$ for a discrete problem, DISOPT may be asked to check the nearest set of discrete solutions about the continuous optimum and store the best feasible solution.

The new variable constraint added at each node always excludes the preceding optimum point from the current solution space and the constraint is therefore active if the function is locally unimodal. Thus the value of the variable under the new constraint may be optionally fixed on the constraint boundary. Hence, only a k-1 variable problem need be solved and much computational effort would be saved.

3.5 Numerical Examples

Four discrete minimization problems have been included here to demonstrate the use of the program.

Example 1: Modified banana shape function.

Minimize

$$f(\underset{\sim}{\phi}) = 100((\phi_2+0.5)-(\phi_1+0.6)^2)^2 + (0.4-\phi_1)^2$$

subject to

$$\phi_1, \phi_2 \text{ natural numbers}$$

The results are tabulated in Table 3.1. This example serves to illustrate that the optimum discrete solution is not guaranteed by simply chopping or rounding off the continuous solution. From the contour plot shown in Figure 3.1 it is obvious that the optimum discrete solution is not given by any of the vertices about the continuous solution. The best vertex is given by $\underset{\sim}{\phi} = [0\ 0]^T$ with a function value $f(\underset{\sim}{\phi}) = 2.12$ which is much higher than that for the optimum discrete solution.

Example 2: Beale constrained function.

Minimize the Beale function (see Section 2.4) subject to the additional constraint that the variables must be integers. The results are shown in Table 3.2. All the three optimum discrete solutions of unity function value are detected by the algorithm. However, if the user indicates that only one optimum discrete solution is required, DISOPT will check the existence of a better solution before solving the nonlinear programming problem at a node. As illustrated by this example, this will reduce the necessary computational effort.

Example 3: Tolerance assignment in the design of a voltage divider [15].

Consider the simple voltage divider as shown in Figure 3.2. The transfer function is given by $T = \phi_4/(\phi_3+\phi_4)$ and the input

Solution	Continuous	Discrete
ϕ_1	0.4000	1
ϕ_2	0.5000	2
$f(\phi)$	0.0000	0.72
Function evaluations	878	
Nodes	9	
Time (sec.)	8	

Table 3.1 Results for example 1 starting at $\phi^0=[-1.8 \quad 0.5]^T$ and using algorithm 3. $p^1 = 4$, $c = 4$.

Solution	Continuous	Discrete		
ϕ_1	1.3333	2	1	2
ϕ_2	0.7778	0	1	1
ϕ_3	0.4444	0	0	0
$f(\phi)$	0.1111	1	1	1
Number of optimum discrete solutions required			3	1
Function evaluations			226	160
Nodes			7	7
Time (sec.)			5	4

Table 3.2 Results for example 2 starting at $\phi^0 = [1\ 2\ 1]^T$ and using algorithm 1. $p = 10^3$.

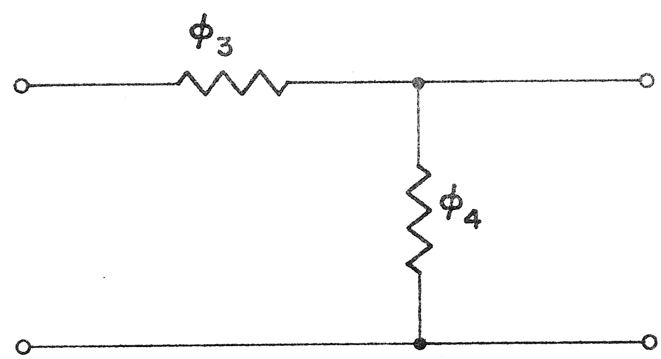

Figure 3.2 Voltage divider used in a tolerance assignment problem.

resistance is $R = \phi_3 + \phi_4$. The design specifications are $0.46 \le T \le 0.53$ and $1.85 \le R \le 2.15$. The obtainable discrete tolerances for both ϕ_3 and ϕ_4 are given by the set

$$S = \{1, 3, 5, 10, 15\} \text{ per cent.}$$

The cost function

$$f = \sum_{i=1}^{2} \frac{1}{\phi_i}$$

where ϕ_i is the percentage tolerance in component ϕ_{i+2}, was first minimized by fixing one variable at each node in the search for discrete solution. The minimization was then repeated as a 4-dimensional problem throughout to highlight the extra amount of effort that was required. The numerical results are shown in Table 3.3. The main program and the user subroutines for this problem are given in the Appendix.

Example 4: Tolerance assignment in the design of a lowpass filter.

The cost function for the tolerance assignment problem in Section 2.4 was minimized with the additional constraint that only the following set, S, of discrete tolerances was available for each of the components:

$$S = \{1, 2, 5, 10, 15\} \text{ per cent.}$$

Two different tree structures are shown in Figures 3.3 and 3.4 and the numerical results are tabulated in Table 3.4. This example illustrates that the tree structure and hence the computational effort is dependent upon the order of partitioning on the discrete variables.

Solution	Continuous	Discrete	
ϕ_1	7.0007	5	
ϕ_2	7.0007	5	
ϕ_3	1.0137		
ϕ_4	0.9935		
f	0.2857	0.4	
Dimensionality of the problem used in the search for discrete solution		3	4
Function evaluations		577	1083
Nodes		9	9
Time (sec.)		10	17

Table 3.3 Results for example 3 starting at $\phi^0 = [1\ 1\ 1\ 1]^T$ and using algorithm 4. p = 6.

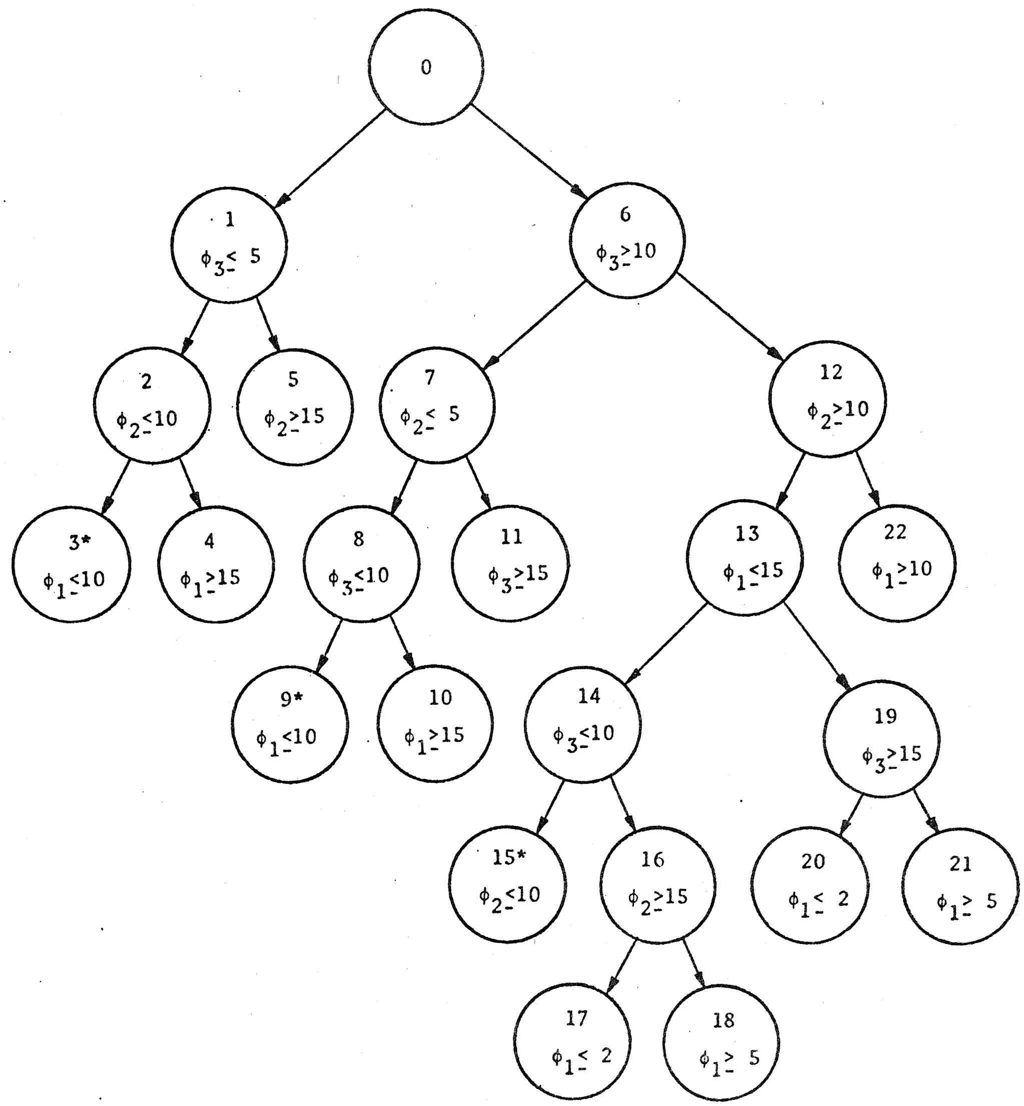

Figure 3.3 A tree-structure for the tolerance assignment in the design of a lowpass filter. Partitioning on ϕ_3 first. * denotes optimum discrete solution.

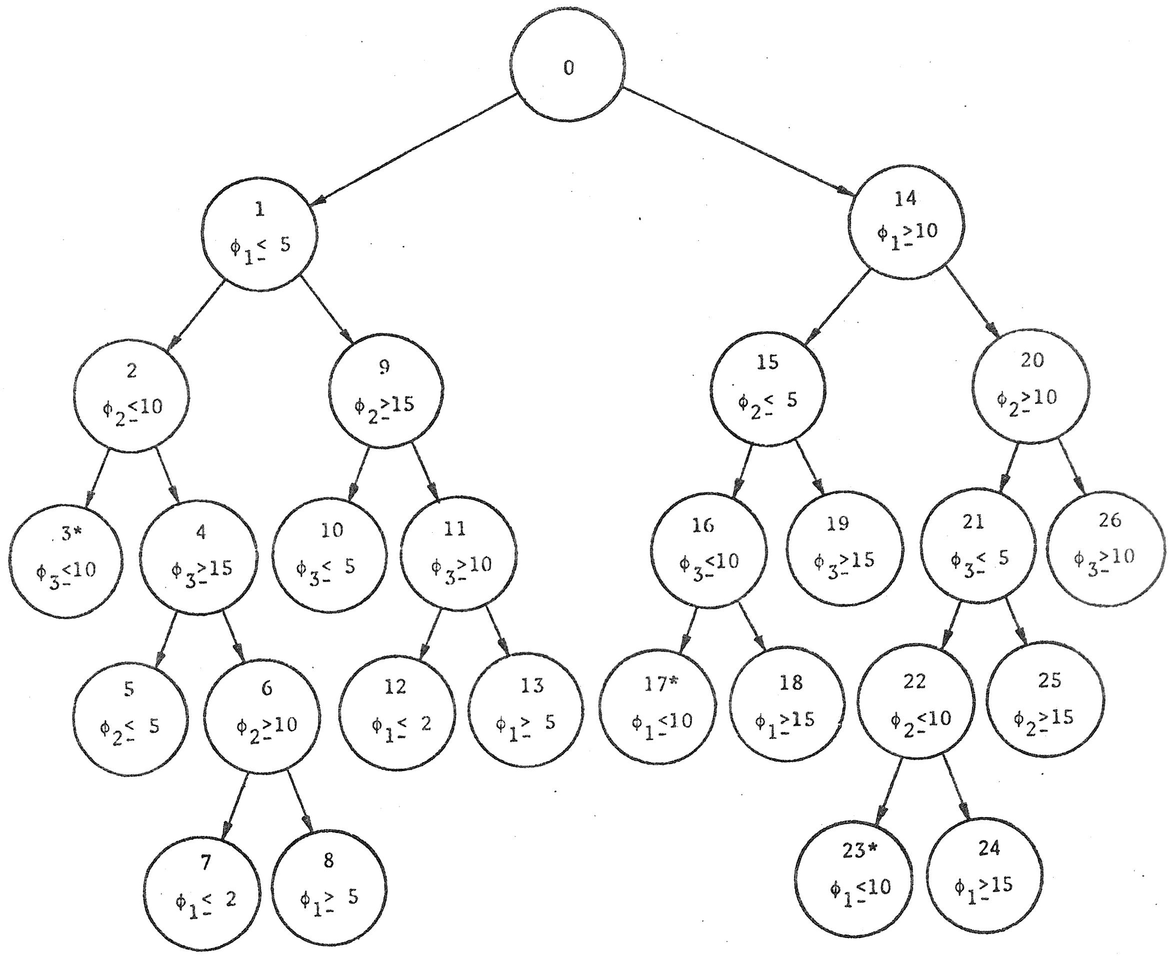

Figure 3.4 A tree-structure for the tolerance assignment in the design of a lowpass filter. Partitioning on ϕ_1 first. * denotes optimum discrete solution.

Solution	Continuous	Discrete		
ϕ_1	7.6061	5	10	10
ϕ_2	9.8978	10	5	10
ϕ_3	9.8978	10	10	5
ϕ_4		0.9056		
ϕ_5		1.9992		
ϕ_6		1.9992		
f	0.3335		0.4	

The discrete variable first used for constructing the variable constraints	ϕ_1	ϕ_3
Function evaluations	3704	3314
Nodes	27	23
Time (sec.)	91	82

Table 3.4 Results for example 4 starting at $\underset{\sim}{\phi}^0 = [5\ 5\ 5\ 1\ 1\ 1]^T$ and using algorithm 5. p = 1.5.

CHAPTER IV

CONCLUSIONS

An integrated optimization program called DISOPT is presented in this thesis. Many up to date algorithms and techniques have been incorporated into one program and made available to the user. Illustrative examples have been included to demonstrate the efficiency of DISOPT and the various options present.

An unfortunate characteristic of optimization is that no one technique is best for all kinds of problems. Hence, it is advantageous to have a multitechnique general program. From the author's experience, algorithm 5 should be recommended only if a good optimistic estimate of the optimum function value is available. Otherwise, the minimax approach to nonlinear programming should be used. If the starting point is not likely to lie in the close vicinity of the optimum, a sequence of least pth optimizations should be used to avoid poor scaling of the problem. However, if the starting point happens to be very close to the optimum, the use of small values of p in the initial optimizations will actually give worse estimates of the optimum.

The amount of programming effort required of the user has been reduced to a minimum. A user is responsible only for

(1) supplying the values and/or proper dimensioning of the parameters in the argument list and

(2) writing any service subroutines to define the objective function, the constraints and their respective partial derivatives.

DISOPT will, on exit, output the required solution or a message if a solution does not exist.

Since many design problems can be easily formulated as nonlinear programming problems, DISOPT enjoys a very wide range of applications. DISOPT [16] has been successfully applied to tolerance optimization in microwave circuits [17] and the optimal design of recursive digital filters with optimum finite word length to meet prescribed magnitude characteristics in the frequency domain [18].

The program can be easily incorporated into other user-oriented computer-aided design packages such as automated digital filter design, tolerance optimization or system modelling packages.

Any modification to DISOPT can be introduced only by a person who is familiar with the whole program structure because of the integration of various subroutines. Thus, one possible future improvement would be to reorganize the program into a coordinated package of independent subroutines.

Engineering design problems often involve least pth approximation in an effort to meet certain performance specifications. Since the subroutine for the formulation of a least pth objective is available in DISOPT, the program can be slightly modified to handle such problems directly without having the user set up the least pth objective himself or reformulate his problem as a nonlinear programming problem.

It has been brought to the author's attention that a similar approach to algorithm 5 has been proposed by Charalambous [19].

APPENDIX

THE DISOPT PROGRAM*

A.1 Purpose

DISOPT is a package of subroutines for solving continuous or discrete, constrained or unconstrained general optimization problems. That is, it minimizes a function

$$f \triangleq f(\underset{\sim}{x})$$

of n variables $\underset{\sim}{x}$ which may be subject to the constraints

$$g_i(\underset{\sim}{x}) \geq 0 \ , \ i=1,2,\ldots,m$$

and/or

$$x_j \ , \ j=1,2,\ldots,k \ , \ k \leq n, \text{ must have certain discrete values.}$$

A constrained problem is transformed into an equivalent unconstrained objective by any of the five algorithms described in Chapter II. The solution of a discrete problem follows the logic of the tree-search algorithm described in Chapter III.

The flow diagram of DISOPT is shown in Figure A.1

*The notation used in the Appendix is designed to appear consistent with the FORTRAN names suggested to the user.

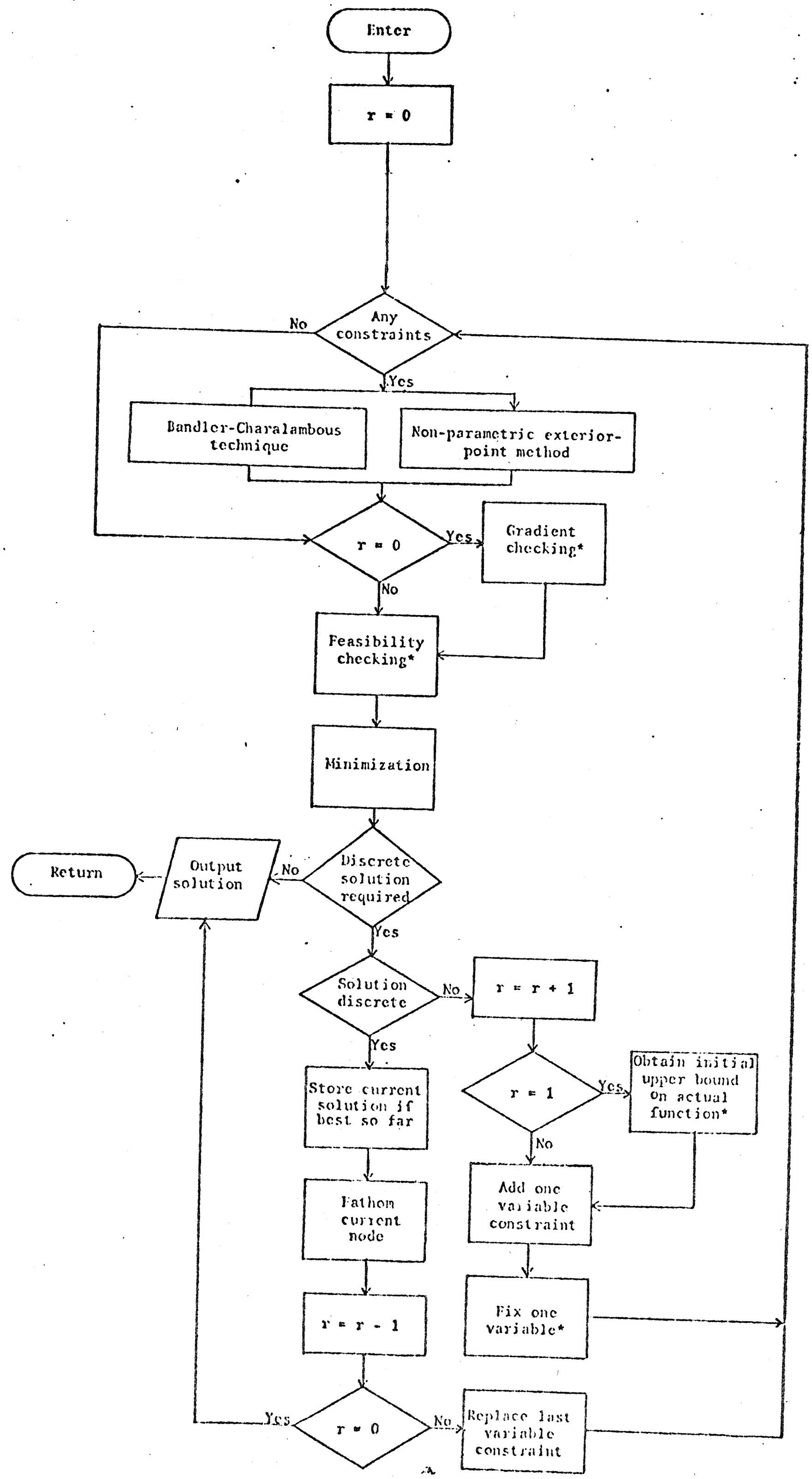

Figure A.1 Flow diagram of DISOPT. * indicates optional.

A.2 The program may be called as follows:

CALL DISOPT (NR, X, EPS, G, NP, PS, K, NSTEP, DISCR, QSTEP, XB, IX, X1, X2, XU, XL, ID, IB, W, H, XE, ICHECK, IVAR, P2, P1, INDX, GPHI, IAA, IBB, GU, PHI, A, T1, T1P, AL, ESTD).

It was convenient to place the following user - specified variables in: COMMON/DSPT01/IOPT1, IOPT2, IOPT3, IOPT4, IOPT5, IOPT6, IOPT7, NCONS, IDATA, IPRINT, MAX, EST, EST1, AO, XMAL, AI, ZERO, ETA, INSOLN, BSOLN, MAXNOD, ERR, ICON.

NR	an integer set to the number of variables ($NR \geq 2$).
X	a real array of NR elements in which the current estimate of the solution is stored. An initial approximation must be set in X on entry, and for the case of the continuous problem, the optimum solution will be returned in X on exit.
EPS	a real array of NR elements set to the test quantities used in the Fletcher program.
G	a real array of NR elements in which the gradient vector corresponding to X above will be returned for the case of the continuous problem.
NP	an integer set to the number of p values used.
PS	a real array of NP elements set to the value(s) of p. The array for extrapolation is internally constructed from PS(1) and PS(2).
K	an integer set to the number of discrete variables. Otherwise, set K to 1.
NSTEP	a real array of K elements set to the number of discrete values available for each of the K discrete variables if IOPT5 = 1.
DISCR	a real two suffix array of K rows and NSTEP columns to be set to the discrete values imposed upon each discrete variable if IOPT5 = 1.

QSTEP	a real array of K elements to be set to the quantization step sizes for the K discrete variables if IOPT5 $\neq$ 1.
XB	a real array of NR elements in which the optimum discrete solution will be returned on exit.
IX, X1, X2, INDX, GU	working arrays of NR elements.
XU, XL	working arrays of K elements.
ID	a working array of 2^K elements.
IB	a two suffix working array of K rows and 2^K columns.
W	a working array of 4NR elements.
H	a working array of NR(NR+1)/2 elements.
XE	a three suffix working array of NR x NP x NP elements.
ICHECK, IVAR, P2, P1, ESTD, AL	working arrays of M elements; here M is the anticipated maximum number of additional variable constraints.
GPHI	a two suffix working array of NR rows and (NCONS+M) columns.
A, T1, T1P, IAA, IBB	working arrays of (NCONS+M+1) elements.
PHI	a working array of (NCONS+M) elements.
IOPT1	an integer set to 1 if the dimensionality of the problem is to be reduced by 1 in the search for discrete solution. Otherwise, set to any other value.
IOPT2	an integer set to 1 if a gradient check at the starting point by perturbation is desired. Otherwise, set to any other value.

IOPT3	an integer set to 1 if the vertices about the continuous solution are to be checked for an initial discrete solution. Otherwise, set to any other value.
IOPT4	an integer set to 1 if the existence of a feasible solution is to be checked from the very beginning or set to 2 if the feasibility check is to be carried out only for the discrete problem. Otherwise, set to any other value.
IOPT5	an integer set to 1 if a finite set of discrete values is imposed upon each discrete variable. Uniform quantization step size for each discrete variable is assumed if IOPT5 is set to any other value.
IOPT6	an integer set to i if algorithm i is to be used. For an unconstrained problem, set IOPT6 to 0.
IOPT7	an integer set to 1 if only one discrete optimum solution is required. Otherwise, set to any other value.
NCONS	an integer set to the number of constraints on the continuous problem.
IDATA	an integer set to 1 if input data is to be printed. Otherwise, set to any other value.
IPRINT	an integer controlling output printing to be set as follows : IPRINT>0, printing at every IPRINT iterations IPRINT=0, printing at each node IPRINT=-1, printing of the optimum continuous and discrete solutions only IPRINT$\leq$-2, printing suppressed.

MAX	an integer set to the maximum permissible number of function evaluations per node.
EST	a real number set to the estimated minimum value of the artificial unconstrained objective.
EST1	a real number set to the initial estimated minimum value of the actual objective function when using algorithm 5.
AO	a real number set to the initial value of α when using algorithms 1 to 4.
XMAL	a real number set to the maximum allowable value of α when using algorithms 1 to 4.
AI	a real number set to the multiplying factor for α when using algorithms 1 to 4.
ZERO	a nonpositive real number set to the error tolerance in the constraints.
ETA	a real number set to the stopping test quantity when using algorithms 2, 4 or 5.
INSOLN	an integer set to 1 if an upper bound on the actual function value is available. Otherwise, set to any other value.
BSOLN	a real number set to the upper bound on the actual function value if INSOLN is set to 1.
MAXNOD	an integer set to the maximum allowable number of nodes. Set MAXNOD to 0 if only the continuous solution is required.
ERR	a real number set to the absolute value of the tolerable error in discrete values if IOPT5 = 1 or the absolute value

of the relative tolerable error with respect to the quantization step size if IOPT5 $\neq$ 1. The optimum discrete solution does not have exact discrete values. DISOPT treats any value, which does not differ from the discrete value by more than the prespecified amount, as the discrete value.

ICON an integer set to 1 if the partitioning is imposed on ϕ_1 first. Otherwise, set to any other value and the partitioning will be imposed on ϕ_k first.

A typical main program to supply the values and proper dimensioning for the parameters of subroutine DISOPT is displayed in Figure A.2 and typical printouts of data and results are shown in Figure A.3. The example used is the tolerance assignment in the design of a voltage divider (see Section 3.5).

A.3 User Subroutine

The user must provide a subroutine headed

```
SUBROUTINE DSPTF (X, G, F, N, GF, INDX, GG, NR, IG)
DIMENSION X(1), G(1), GF(1), INDX(1), GG(NR, 1)
```

This subroutine should use the values of the design parameters supplied in array X, the current number of variables supplied in N, and the index set for the current variables supplied in array INDX to compute the objective function, the constraint functions and their corresponding partial derivatives and place them in F and arrays G, GF and GG, respectively.

A zero value of the input parameter IG indicates that the partial derivatives are not required.

```
      PROGRAM MAIN (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
C     DISCRETE TOLERANCE OPTIMIZATION IN THE VOLTAGE DIVIDER EXAMPLE
C
      DIMENSION X(4),EPS(4),G(4),PS(1),NSTEP(2),DISCR(2,5),QSTEP(2),
     1  XB(4),IX(4),X1(4),X2(4),INDX(4),GU(4),XU(2),XL(2),ID(4),IB(2,4),
     2  W(16),H(10),XE(4,1,1),GPHI(4,16),PHI(16),ICHECK(10),IVAR(10),
     3  P2(10),P1(10),ESTD(10),AL(10),A(17),T1(17),T1P(17),IAA(17),
     4  IBB(17)
      COMMON /DSPT01/ IOPT1,IOPT2,IOPT3,IOPT4,IOPT5,IOPT6,IOPT7,
     1  NCONS,IDATA,IPRINT,MAX,EST,EST1,AO,XMAL,AI,ZERO,ETA,INSOLN,
     2  BSOLN,MAXNOD,ERR,ICON
C
C     THIS MAIN PROGRAM SUPPLIES THE VALUES AND/OR PROPER DIMENSIONING
C     OF THE PARAMETERS IN THE ARGUMENT LIST
C
C     READS INPUT DATA
C
      READ(5,6) IOPT1,IOPT2,IOPT3,IOPT4,IOPT5,IOPT6,IOPT7,NR,K,NCONS,NP
      IF (IOPT6.EQ.0) GO TO 5
      IF(IOPT5.NE.1) GO TO 2
      READ (5,6) (NSTEP(I),I=1,K)
      DO 1 I=1,K
      NS=NSTEP(I)
      READ (5,7) (DISCR(I,J),J=1,NS)
1     CONTINUE
      GO TO 3
2     READ (5,7) (QSTEP(I),I=1,K)
3     READ (5,7) (PS(I),I=1,NP)
5     READ (5,6) MAX,IPRINT,IDATA,MAXNOD,ICON,INSOLN
      READ (5,7) (X(I),I=1,NR)
      READ (5,7) (EPS(I),I=1,NR)
      READ (5,7) EST,ZERO,ERR,ETA,EST1,AO,XMAL,AI,BSOLN
C
6     FORMAT (16I5)
7     FORMAT (5E16.8)
C
      CALL DISOPT (NR,X,EPS,G,NP,PS,K,NSTEP,DISCR,QSTEP,XB,IX,X1,X2,XU,
     1  XL,ID,IB,W,H,XE,ICHECK,IVAR,P2,P1,INDX,GPHI,IAA,IBB,GU,PHI,A,T1,
     2  T1P,AL,ESTD)
C
      STOP
      END
C
C     INPUT DATA
C
    1    1    1    2    1    4    0    4    2    6    1
    5    5
1.              3.              5.              1.00000000E+01  1.50000000E+01
1.              3.              5.              1.00000000E+01  1.50000000E+01
6.
300   20    1   12    0
1.              1.              1.              1.
1.00000000E-06  1.00000000E-06  1.00000000E-06  1.00000000E-06
0.             -1.00000000E-06  5.00000000E-03  1.00000000E-03
1.00000000E+02  1.00000000E+05  1.00000000E+01
```

Figure A.2 Typical main program for the DISOPT program.

```
INPUT DATA
----------

DISCRETE VALUES FOR THE VARIABLES
   .10000000E+01   .30000000E+01   .50000000E+01   .10000000E+02   .15000000E+02
   .10000000E+01   .30000000E+01   .50000000E+01   .10000000E+02   .15000000E+02
NUMBER OF INDEPENDENT VARIABLES....................................NR=    4
NUMBER OF DISCRETE VARIABLES........................................K=    2
NUMBER OF CONSTRAINTS ON THE CONTINUOUS PROBLEM.................NCONS=    6
MAXIMUM NUMBER OF ALLOWABLE FUNCTION EVALUATIONS..................MAX=  300
MAXIMUM NUMBER OF NODES TO BE SEARCHED.........................MAXNOD=   12
INTERMEDIATE OUTPUT TO BE PRINTED EVERY IPRINT ITERATIONS......IPRINT=   20
STARTING VALUE FOR VECTOR X(I)..................................X( 1)=   .10000000E+01
                                                                X( 2)=   .10000000E+01
                                                                X( 3)=   .10000000E+01
                                                                X( 4)=   .10000000E+01
ERROR TOLERANCE IN CONSTRAINTS...................................ZERO=  -.10000000E-05
ERROR TOLERANCE IN DISCRETE VALUES................................ERR=   .50000000E-02
TEST QUANTITIES TO BE USED IN FLETCHER METHOD.................EPS( 1)=   .10000000E-05
                                                              EPS( 2)=   .10000000E-05
                                                              EPS( 3)=   .10000000E-05
                                                              EPS( 4)=   .10000000E-05
ESTIMATE OF LOWER BOUND ON ARTIFICIAL OBJECTIVE FUNCTION..........EST=  0.
INITIAL VALUE OF THE PARAMETER ALPHA...............................AC=   .10000000E+03
MAXIMUM ALLOWABLE VALUE OF THE PARAMETER ALPHA...................XMAL=   .10000000E+06
MULTIPLYING FACTOR IN ALPHA VALUE..................................AI=   .10000000E+02
TEST QUANTITY TO BE USED IN NLP ALGORITHM 2/4/5...................ETA=   .10000000E-02
NUMBER OF P VALUES.................................................NP=    1
VALUE(S) OF P USED IN NLP ALGORITHM............................PS( 1)=   .60000000E+01
FOLLOWING OPTIONS USED
----------------------

NLP ALGORITHM 4 EMPLOYED
(N-1) VARIABLE OPTIMIZATION PERFORMED
VERTICES CHECKED
FEASIBILITY CHECKED FOR DISCRETE PROBLEM ONLY
PARTITIONING STARTS ON LAST DISCRETE VARIABLE
```

Figure A.3 Typical printouts of input data and results for the DISOPT program.

```
        FOLLOWING IS THE OPTIMUM SOLUTION
        ---------------------------------

                         NODE NUMBER =       0

ARTIFICIAL UNCONSTRAINED FUNCTION U =   -.20506248E-04

        ACTUAL OBJECTIVE FUNCTION F =    .28568653E+00

   X( 1)=    .70006801E+01 GU( 1)=      .54258628E-08
   X( 2)=    .70006799E+01 GU( 2)=      .67832173E-09
   X( 3)=    .10136897E+01 GU( 3)=     -.25270715E-05
   X( 4)=    .99351873E+00 GU( 4)=      .25783774E-05

   INEQUALITY CONSTRAINTS
                             G( 1)=     .70006801E+01
                             G( 2)=     .70006799E+01
                             G( 3)=     .11745128E-06
                             G( 4)=     .11731997E-06
                             G( 5)=     .22733623E-02
                             G( 6)=     .16690162E-01

     NUMBER OF VIOLATED CONSTRAINTS =     0
    NUMBER OF FUNCTION EVALUATIONS =     98
  FINAL VALUE OF THE PARAMETER ALPHA =   .10000000E+03
          EXECUTION TIME IN SECONDS =     .99800



        BEST DISCRETE SOLUTION FOUND SO FAR
                           F =     .40000000E+00

                        X( 1)  =   .50000000E+01
                        X( 2)  =   .50000000E+01
                        X( 3)  =   .10136897E+01
                        X( 4)  =   .99351873E+00

INEQUALITY CONSTRAINTS  G( 1)  =   .50000000E+01
                        G( 2)  =   .50000000E+01
                        G( 3)  =   .10014581E-01
                        G( 4)  =   .99904556E-02
                        G( 5)  =   .42431180E-01
                        G( 6)  =   .56847980E-01

NUMBER OF FUNCTION EVALUATIONS =  104



          OPTIMUM DISCRETE SOLUTION FOUND
                    MINIMUM  F =   .40000000E+00

                        X( 1)  =   .50000000E+01
                        X( 2)  =   .50000000E+01
                        X( 3)  =   .10136897E+01
                        X( 4)  =   .99351873E+00

INEQUALITY CONSTRAINTS  G( 1)  =   .50000000E+01
                        G( 2)  =   .50000000E+01
                        G( 3)  =   .10014581E-01
                        G( 4)  =   .99904556E-02
                        G( 5)  =   .42431180E-01
                        G( 6)  =   .56847980E-01

NUMBER OF FUNCTION EVALUATIONS =  577
```

Figure A.3 Typical printouts of input data and results for the DISOPT program [continued].

A typical user subroutine is shown in Figure A.4. Again, the voltage divider problem is chosen as the example.

A.4 Other Subroutines

The following is a brief description of the subroutines called by DISOPT.

DSPTA coordinates the input, the output and the minimization.

DSPTB minimizes a function using the Fletcher unconstrained minimization program.

DSPTC formulates the artificial unconstrained objective function and the necessary gradients.

DSPTD supplies additional variable constraints for discrete optimization.

DSPTE returns the gradients of the additional variable constraints.

DSPTH transforms a nonlinear programming problem into an equivalent unconstrained objective function.

DSPTI prints the input data.

DSPTJ outputs the result of the feasibility check and/or the optimum solution at each node.

DSPTK outputs the best current discrete solution after checking the vertices about the continuous solution and the optimum discrete solution.

DSPTL checks the gradient formulation by perturbation.

DSPTM performs extrapolation when using algorithm 3.

The overall structure of the program is shown in Figure A.5.

```
      SUBROUTINE DSPTF(X,G,F,N,GF,INDX,GG,NR,IG)
C
      DIMENSION X(1),G(1),GF(1),INDX(1),GG(NR,1),F(2),DE(4)
C
C     THIS SUBROUTINE DEFINES THE OBJECTIVE FUNCTION, THE CONSTRAINTS
C     AND THEIR GRADIENTS OF THE CONTINUOUS PROBLEM
C
      TM=1./X(1)
      TN=1./X(2)
      F=TM+TN
      DE(1)=X(1)*0.01
      DE(2)=X(2)*0.01
      E(1)=DE(1)*X(3)
      E(2)=DE(2)*X(4)
      TA=X(3)+E(1)
      TB=X(3)-E(1)
      TC=X(4)+E(2)
      TD=X(4)-E(2)
      TE=TB+TC
      TF=TA+TD
      G(1)=X(1)
      G(2)=X(2)
      G(3)=0.53-TC/TE
      G(4)=TD/TF-0.46
      G(5)=2.15-TC-TA
      G(6)=TD+TB-1.85
      IF(IG.EQ.0) RETURN
      DE(3)=X(3)*0.01
      DE(4)=X(4)*0.01
      TG=TE*TE
      TH=TF*TF
      TI=1.+DE(1)
      TJ=1.-DE(1)
      TK=1.+DE(2)
      TL=1.-DE(2)
      TP=TC/TG
      TQ=-TD/TH
      TR=TA/TH
      TS=-TB/TG
      DO 5 I=1,N
      IND=INDX(I)
      GO TO (1,2,3,4,5), IND
1     GF(1)=-TM/X(1)
      GG(1,1)=1.
      GG(1,2)=0.
      GG(1,3)=-TP*DE(3)
      GG(1,4)=TQ*DE(3)
      GG(1,5)=-DE(3)
      GG(1,6)=GG(1,5)
      GO TO 5
2     GF(2)=-TN/X(2)
      GG(2,1)=0.
      GG(2,2)=1.
      GG(2,3)=TS*DE(4)
      GG(2,4)=-TR*DE(4)
      GG(2,5)=-DE(4)
      GG(2,6)=GG(2,5)
      GO TO 5
```

Figure A.4 Typical user subroutine for the DISOPT program.

```
3     GF(3)=0.
      GG(3,1)=0.
      GG(3,2)=0.
      GG(3,3)=TP*TJ
      GG(3,4)=TQ*TI
      GG(3,5)=-TI
      GG(3,6)=TJ
      GO TO 5
4     GF(4)=0.
      GG(4,1)=0.
      GG(4,2)=0.
      GG(4,3)=TS*TK
      GG(4,4)=TR*TL
      GG(4,5)=-TK
      GG(4,6)=TL
5     CONTINUF
      RETURN
      END
```

Figure A.4 Typical user subroutine for the DISOPT program [continued].

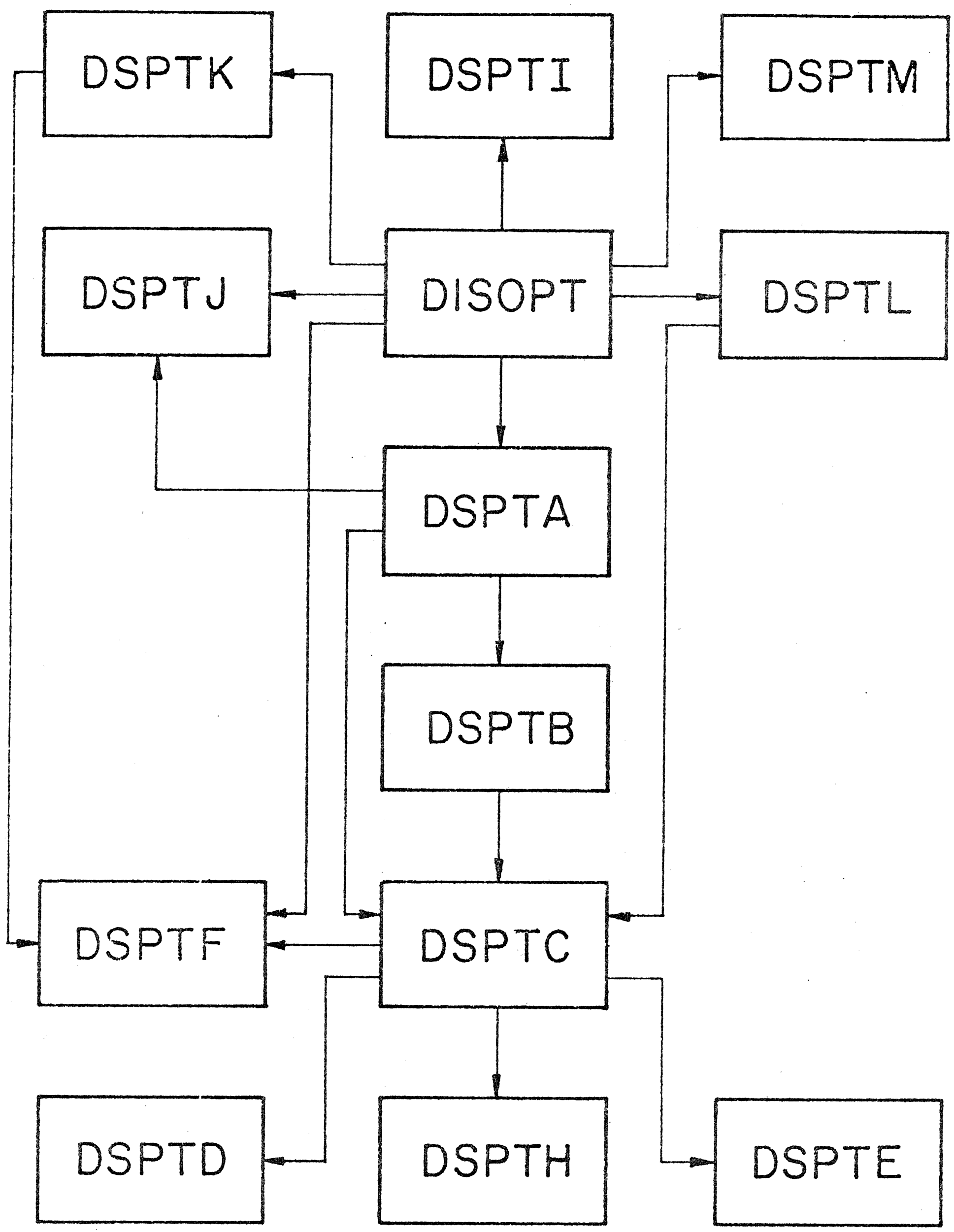

Figure A.5 Overall structure of DISOPT.

A.5 FORTRAN Listing for DISOPT Program

```
      SUBROUTINE DISOPT (NR,X,EPS,G,NP,PS,K,NSTEP,DISCR,QSTEP,XB,IX,X1,X
     12,XU,XL,ID,IB,W,H,XE,ICHECK,IVAR,P2,P1,INDX,GPHI,IAA,IBB,GU,PHI,A,
     2T1,T1P,AL,ESTD)
C
      DIMENSION NSTEP(1), DISCR(K,1), QSTEP(1), XX(1), PY(1), Y(1), IX(1
     1), XB(1), PHI(1), X(1), G(1), EPS(1), H(1), W(1), XL(1), ID(1), IB
     2(K,1), IAA(1), IBB(1), XU(1), PS(1), ESTD(1), X1(1), X2(1), XE(NR,
     3NP,1), ICHECK(1), IVAR(1), P2(1), P1(1), INDX(1), GPHI(NR,1), GU(1
     4), A(1), T1(1), T1P(1), AL(1)
      COMMON /DSPT01/ IOPT1,IOPT2,IOPT3,IOPT4,IOPT5,IOPT6,IOPT7,NCONS,ID
     1ATA,IPRINT,MAX,EST,EST1,A0,XMAL,AI,ZERO,ETA,INSOLN,RSOLN,MAXNOD,ER
     2R,ICON
      COMMON /DSPT02/ NOD,KK,NORG,NOR
      COMMON /DSPT03/ AM,PSI,PA,ALPHA,IFLAGA,ICHEK,KKK,INDA,INDB,UR,NC,K
     10,IFN
      COMMON /DSPT04/ SUMD,INDC
C
C     THIS SUBROUTINE SOLVES CONTINUOUS OR DISCRETE PROGRAMMING PROBLEMS
C     THE SOLUTION OF A DISCRETE PROBLEM FOLLOWS THE LOGIC OF DAKINS
C     TREE-SEARCH ALGORITHM
C
C
C     J.H.K. CHEN, DISOPT- A GENERAL PROGRAM FOR CONTINUOUS AND DISCRETE
C     NONLINEAR PROGRAMMING PROBLEMS, MCMASTER UNIVERSITY, HAMILTON,
C     CANADA, INTERNAL REPORT IN SIMULATION, OPTIMIZATION AND CONTROL,
C     NO. SOC-29, MARCH 1974
C
      INDC=0
      NOR=NR
      N=NOR
      INSOL=0
      IFNT=0
      AL(1)=A0
      INDA=0
      IF (IDATA.NE.1) GO TO 1
C
C     PRINTS INPUT DATA
C
      CALL DSPTI (K,N,EPS,X,PS,NP,QSTEP,NSTEP,DISCR)
1     KK=0
      NOD=0
      NORG=NCONS
      PSI=0.
      ESTD(1)=EST1
2     IF (IOPT1.NE.1.AND.KK.GT.0) GO TO 5
      DO 3 I=1,NOR
      INDX(I)=I
3     CONTINUE
      IF (IOPT2.NE.1) GO TO 5
      IF (IOPT6.EQ.0) GO TO 4
      PA=PS(1)
4     ICHEK=0
      ALPHA=AL(1)
C
C     GRADIENT CHECK AT STARTING POINT BY NUMERICAL PERTURBATION
C
      CALL DSPTL (N,X,G,X1,X2,IPRINT,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,
     1IBB,GU,PHI,A,T1,T1P)
```

```
      IOPT2=0
 5    KKK=0
      IF (IPRINT.GT.0) WRITE (6,102)
      CALL SECOND (T3)
      IF (NCONS.EQ.0) GO TO 21
      IF (IOPT1.EQ.1.AND.NCONS.EQ.1) GO TO 21
      IF (KK.GT.0.AND.IOPT4.EQ.2) IOPT4=1
      IF (IOPT4.NE.1) GO TO 6
C
C     FEASIBILITY CHECK, THE VALUE OF P USED IS 2
C
      PA=2.
      INDA=1
      CALL DSPTA (N,X,G,H,EPS,1,W,F,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,I
     1BB,GU,PHI,A,T1,T1P,AL)
      KKK=1
      INDA=0
      IF (IOPT6.EQ.-1) GO TO 93
      IF (IFLAGA.EQ.1) GO TO 24
      IF (IPRINT.GT.0) WRITE (6,102)
C
C     ONE OF THE LEAST PTH OPTIMIZATION ALGORITHMS IS EMPLOYED
C
 6    IF (IOPT6.EQ.0) GO TO 22
      IF (IOPT6.EQ.2) GO TO 11
      IF (IOPT6.EQ.4) GO TO 7
      IF (IOPT6.EQ.5) GO TO 9
      IF (IOPT6.EQ.3) GO TO 14
      GO TO 22
C
C     NONLINEAR MINIMAX OPTIMIZATION AS A SEQUENCE OF LEAST PTH
C     OPTIMIZATION WITH FINITE VALUES OF P
C
 7    IT=1
      PA=PS(1)
 8    CALL DSPTA (N,X,G,H,EPS,IT,W,F,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,
     1IBB,GU,PHI,A,T1,T1P,AL)
      IF (KO.EQ.0) GO TO 23
      IT=IT+1
      KKK=1
      PSIO=PSI
      PSI=AM+PSIO+1.E-10
      IF (IPRINT.GT.0) WRITE (6,94) PSIO
      IF (IT.EQ.2) GO TO 8
      IF (ABS((PSIO-PSI)/PSIO).GT.ETA) GO TO 8
      GO TO 23
C
C     MODIFIED NON-PARAMETRIC EXTERIOR-POINT METHOD
C
 9    IT=1
      PA=PS(1)
      PSI=ESTD(NOD+1)
 10   CALL DSPTA (N,X,G,H,EPS,IT,W,F,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,
     1IBB,GU,PHI,A,T1,T1P,AL)
      IF (KO.EQ.0) GO TO 23
      IT=IT+1
      KKK=1
      KR=0
```

```
      PSIO=PSI
      PSI=PSIO+SUMD
      IF (IPRINT.GT.0) WRITE (6,95) PSIO
      IF (IT.EQ.2) GO TO 10
      IF (ABS(SUMD/PSIO).GT.ETA) GO TO 10
      GO TO 23
C
C     NONLINEAR MINIMAX OPTIMIZATION AS A SEQUENCE OF LEAST PTH
C     OPTIMIZATION WITH INCREASING VALUES OF P
C
11    DO 13 I=1,NP
      PA=PS(I)
      CALL DSPTA (N,X,G,H,EPS,I,W,F,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,I
     1BB,GU,PHI,A,T1,T1P,AL)
      IF (KO.EQ.0) GO TO 23
      IF (I.LT.2) GO TO 12
      IF (ABS((AMD-AM)/AMD).LE.ETA) GO TO 23
12    AMD=AM
13    CONTINUE
      GO TO 23
C
C     APPLICATION OF AN EXTRAPOLATION TECHNIQUE TO A SEQUENCE OF LEAST
C     PTH OPTIMIZATIONS WITH GEOMETRICALLY INCREASING VALUES OF P
C
14    JORDER=NP-1
      PI=PS(2)/PS(1)
      DO 20 I=1,NP
      PA=PS(I)
      CALL DSPTA (N,X,G,H,EPS,I,W,F,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,I
     1BB,GU,PHI,A,T1,T1P,AL)
      IF (KO.EQ.0) GO TO 23
      CALL DSPTM (NOR,X,XE,I,NP,PI,X1,JORDER)
      IF (I.EQ.1) GO TO 18
      IF (I.EQ.NP) GO TO 16
      DO 15 J=1,NOR
      X2(J)=ABS(X2(J)-X1(J))
      IF (X2(J).GT.EPS(J)*10.) GO TO 18
15    CONTINUE
16    DO 17 J=1,NOR
      X(J)=X1(J)
17    CONTINUE
      GO TO 23
18    DO 19 J=1,NOR
      X2(J)=X1(J)
19    CONTINUE
20    CONTINUE
      GO TO 23
21    PA=PS(1)
C
C     NONLINEAR MINIMAX OPTIMIZATION AS A LEAST PTH OPTIMIZATION WITH A
C     LARGE VALUE OF P
C
22    CALL DSPTA (N,X,G,H,EPS,1,W,F,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,I
     1BB,GU,PHI,A,T1,T1P,AL)
23    CALL DSPTF (X,PHI,U,N,GU,INDX,GPHI,NR,0)
      CALL DSPTD (X,PHI,IAA,IBB,ICHECK,IVAR,P2,P1)
      IFNT=IFNT+1
      IF (IPRINT.LE.-2) GO TO 25
```

```
      IF (IPRINT.EQ.-1.AND.KK.GT.0) GO TO 25
      CALL DSPTJ (N,X,F,G,PHI,U,IVAR)
 24   IF (IPRINT.LT.0) GO TO 25
      CALL SECOND (T)
      T=T-T3
      WRITE (6,101) T
 25   KK=KK+1
      INDC=0
      IFNT=IFNT+IFN
C
C     CHECK IF SOLUTION IS FEASIBLE
C
      IF (IOPT6.EQ.5) FSTD(NOD+2)=U
      IF (IFLAGA.NE.0) GO TO 26
      IF (MAXNOD.EQ.0) GO TO 92
      GO TO 29
 26   IF (KK.EQ.1) GO TO 87
C
C     CHECK IF ALTERNATIVE CONSTRAINT AT A PARTICULAR NODE HAS BEEN
C     ADDED
C
 27   IF (ICHECK(NOD).EQ.0.OR.ICHECK(NOD).EQ.2) GO TO 28
      ICHECK(NOD)=0
      NCONS=NORG+NOD
      GO TO 2
C
C     CHECK IF ALL NODES HAVE BEEN SEARCHED
C
 28   NOD=NOD-1
      IF (NOD.EQ.0) GO TO 86
      GO TO 27
C
C     CHECK IF DISCRETE VALUE SOLUTION IS ATTAINED
C
 29   IF (IOPT5.EQ.1) GO TO 30
      GO TO 49
 30   DO 40 M=1,K
      IF (ICON.NE.1) GO TO 31
      I=M
      GO TO 32
 31   I=K+1-M
 32   NS=NSTEP(I)
      IF (X(I).LT.DISCR(I,1)) GO TO 33
      IF (X(I).GT.DISCR(I,NS)) GO TO 35
      GO TO 37
 33   D1=DISCR(I,1)-X(I)
      IF (D1.LE.ERR) GO TO 40
      IF (INSOLN.EQ.0) GO TO 34
      IF (IOPT7.EQ.1) DIFFER=(BSOLN*(1.-SIGN(1.E-6,BSOLN))+ZERO)-U
      IF (IOPT7.NE.1) DIFFER=BSOLN-U
      IF (DIFFER.LT.ZERO) GO TO 27
 34   NOD=NOD+1
      ICHECK(NOD)=0
      IVAR(NOD)=I
      P2(NOD)=DISCR(I,1)
      GO TO 43
 35   D2=X(I)-DISCR(I,NS)
      IF (D2.LE.ERR) GO TO 40
```

```
      IF (INSOLN.EQ.0) GO TO 36
      IF (IOPT7.EQ.1) DIFFER=(BSOLN*(1.-SIGN(1.E-6,BSOLN))+ZERO)-U
      IF (IOPT7.NE.1) DIFFER=BSOLN-U
      IF (DIFFER.LT.ZERO) GO TO 27
36    NOD=NOD+1
      ICHECK(NOD)=2
      IVAR(NOD)=I
      P1(NOD)=DISCR(I,NS)
      GO TO 43
37    NV=NS-1
      DO 38 J=1,NV
      IF (X(I).GE.DISCR(I,J).AND.X(I).LE.DISCR(I,J+1)) GO TO 39
 8    CONTINUE
39    D1=X(I)-DISCR(I,J)
      D2=DISCR(I,J+1)-X(I)
      IF (D1.GT.ERR.AND.D2.GT.ERR) GO TO 41
40    CONTINUE
      GO TO 76
41    L=I
      LL=J
      IF (INSOLN.EQ.0) GO TO 42
      IF (IOPT7.EQ.1) DIFFER=(BSOLN*(1.-SIGN(1.E-6,BSOLN))+ZERO)-U
      IF (IOPT7.NE.1) DIFFER=BSOLN-U
      IF (DIFFER.LT.ZERO) GO TO 27
42    NOD=NOD+1
      ICHECK(NOD)=1
      IVAR(NOD)=L
      P1(NOD)=DISCR(L,LL)
      P2(NOD)=DISCR(L,LL+1)
43    IF (KK.NE.1.OR.IOPT3.NE.1) GO TO 75
      DO 48 I=1,K
      IF (X(I).LT.DISCR(I,1)) GO TO 45
      IF (X(I).GT.DISCR(I,NS)) GO TO 46
      NS=NSTEP(I)
      DO 44 J=1,NS
      IF (X(I).GE.DISCR(I,J).AND.X(I).LE.DISCR(I,J+1)) GO TO 47
44    CONTINUE
45    XL(I)=DISCR(I,1)
      XU(I)=XL(I)
      GO TO 48
46    XL(I)=DISCR(I,NS)
      XU(I)=XL(I)
      GO TO 48
47    XL(I)=DISCR(I,J)
      XU(I)=DISCR(I,J+1)
48    CONTINUE
      GO TO 55
49    DO 52 J=1,K
      IF (ICON.NE.1) GO TO 50
      I=J
      GO TO 51
50    I=K+1-J
C
C     X IS INCREASED BY THE TOLERATED ERROR TO GET PROPER DISCRETE
C     VALUES OF X
C
51    ERRO=ERR*QSTEP(I)
      D1=SIGN(ERRO,X(I))
```

```
      X(I)=X(I)+D1
      IX(I)=IFIX(X(I)/QSTEP(I))
      X(I)=X(I)-D1
      X1(I)=X(I)-FLOAT(IX(I))*QSTEP(I)
      IF (ABS(X1(I)).GT.ERRO) GO TO 53
52    CONTINUE
      GO TO 76
53    L=I
      IF (KK.NE.1.OR.IOPT3.NE.1) GO TO 72
      DO 54 I=1,K
      ERRO=ERR*QSTEP(I)
      D1=SIGN(ERRO,X(I))
      X(I)=X(I)+D1
      IX(I)=IFIX(X(I)/QSTEP(I))
      XL(I)=FLOAT(IX(I))*QSTEP(I)
      X(I)=X(I)-D1
54    CONTINUE
C
C     CHECK THE VERTICES ABOUT THE SOLUTION TO THE ORIGINAL
C     CONTINUOUS PROBLEM TO OBTAIN AN INITIAL UPPER BOUND ON THE
C     OBJECTIVE FUNCTION
C
55    NV=2**K
      DO 56 I=1,NV
      ID(I)=I
56    CONTINUE
      DO 58 I=1,NV
      ISUM=1
      DO 57 J=1,K
      M=K+1-J
      MP=2**(M-1)
      IB(M,I)=(ID(I)-ISUM)/MP
      ISUM=ISUM+IB(M,I)*MP
57    CONTINUE
58    CONTINUE
      IF (K.EQ.NOR) GO TO 60
      KP1=K+1
      DO 59 I=KP1,NOR
      X1(I)=X(I)
59    CONTINUE
60    DO 70 I=1,NV
      IF (IOPT5.EQ.1) GO TO 62
      DO 61 J=1,K
      X1(J)=XL(J)+SIGN(FLOAT(IB(J,I))*QSTEP(J),X(J))
61    CONTINUE
      GO TO 64
62    DO 63 J=1,K
      IF (IB(J,I).EQ.0) X1(J)=XL(J)
      IF (IB(J,I).EQ.1) X1(J)=XU(J)
63    CONTINUE
64    CALL DSPTF (X1,PHI,UD,N,GU,INDX,GPHI,NR,0)
      IFNT=IFNT+1
      IF (NORG.EQ.0) GO TO 66
      DO 65 J=1,NORG
      IF (PHI(J).LT.ZERO) GO TO 70
65    CONTINUE
66    IF (INSOLN.NE.0) GO TO 69
```

```
67    BSOLN=UD
      INSOLN=1
      INSOL=1
      DO 68 J=1,NOR
      XB(J)=X1(J)
68    CONTINUE
      GO TO 70
69    IF (UD.GE.BSOLN) GO TO 70
      GO TO 67
70    CONTINUE
      IF (INSOL.EQ.0) GO TO 71
      KO=1
      IF (IPRINT.GT.-2) CALL DSPTK (BSOLN,XB,PHI,KO,IFNT,IAA,IBB,N,GU,IN
     1DX,GPHI,NR)
71    IF (IOPT5.EQ.1) GO TO 75
      GO TO 73
C
C     TERMINATE A NODE IF CORRESPONDING OBJECTIVE FUNCTION VALUE IS
C     WORSE THAN THE BEST DISCRETE VALUE SOLUTION OBTAINED SO FAR
C
72    IF (INSOLN.EQ.0) GO TO 73
      IF (IOPT7.EQ.1) DIFFER=(BSOLN*(1.-SIGN(1.E-6,BSOLN))+ZERO)-U
      IF (IOPT7.NE.1) DIFFER=BSOLN-U
      IF (DIFFER.LT.ZERO) GO TO 27
C
C     IF SOLUTION IS NOT DISCRETE ADD CONSTRAINTS
C
73    NOD=NOD+1
      ICHECK(NOD)=1
      IVAR(NOD)=L
      IF (X(L).LT.0.) GO TO 74
      P1(NOD)=FLOAT(IX(L))*QSTEP(L)
      P2(NOD)=P1(NOD)+QSTEP(L)
      GO TO 75
74    P2(NOD)=FLOAT(IX(L))*QSTEP(L)
      P1(NOD)=P2(NOD)-QSTEP(L)
75    NCONS=NORG+NOD
C
C     CHECK IF MAXIMUM NUMBER OF NODES ALLOWED HAS BEEN EXCEEDED
C
      IF (KK.GT.MAXNOD) GO TO 88
      GO TO 2
C
C     IF DISCRETE VALUE SOLUTION IS BEST SO FAR RECORD IT
C
76    NNK=INSOLN
      IF (IPRINT.GE.0) WRITE (6,103)
      INSOLN=1
      IF (KK.EQ.1) GO TO 85
      IF (NNK.NE.0) GO TO 84
C
C     NON-ZERO VALUE OF NNK INDICATES THAT AT LEAST A DISCRETE SOLUTION
C     HAS BEEN FOUND
C
77    IF (IOPT5.EQ.1) GO TO 80
      DO 78 I=1,K
      XB(I)=FLOAT(IX(I))*QSTEP(I)
78    CONTINUE
```

```
      IF (K.EQ.NOR) GO TO 83
      KP1=K+1
      DO 79 I=KP1,NOR
      XB(I)=X(I)
79    CONTINUE
      GO TO 83
80    DO 81 I=1,K
      XB(I)=X(I)+SIGN(ERR,X(I))
      IX(I)=IFIX(XB(I))
      XB(I)=FLOAT(IX(I))
81    CONTINUE
      IF (K.EQ.NOR) GO TO 83
      KP1=K+1
      DO 82 I=KP1,NOR
      XB(I)=X(I)
82    CONTINUE
83    CALL DSPTF (XB,PHI,BSOLN,N,GU,INDX,GPHI,NR,0)
      IFNT=IFNT+1
      INSOL=1
      GO TO 27
84    IF (U.LT.BSOLN) GO TO 77
      GO TO 27
85    KO=0
      IF (IPRINT.GT.-2) CALL DSPTK (U,X,PHI,KO,IFNT,IAA,IBB,N,GU,INDX,GP
     1HI,NR)
      RETURN
86    KO=0
      IF (INSOL.EQ.0) GO TO 90
      GO TO 91
87    WRITE (6,96)
      RETURN
88    IF (INSOL.EQ.0) GO TO 89
      WRITE (6,97)
      KO=1
      GO TO 91
89    WRITE (6,98) MAXNOD
      RETURN
90    WRITE (6,99)
      RETURN
 1    IF (IPRINT.GT.-2) CALL DSPTK (BSOLN,XB,PHI,KO,IFNT,IAA,IBB,N,GU,IN
     1DX,GPHI,NR)
      RETURN
92    WRITE (6,100)
      RETURN
 3    WRITE (6,104)
      RETURN
C
C
C
94    FORMAT (1H0,*VALUE OF PSI =*,E14.6)
95    FORMAT (1H0,*ESTIMATE OF MINIMUM ACTUAL FUNCTION VALUE =*,E14.6)
96    FORMAT (1H0,*NO CONTINUOUS SOLUTION*)
97    FORMAT (1H0,*MAXIMUM ALLOWABLE NUMBER OF NODES EXCEEDED, BEST DISC
     1RETE SOLUTION IS PRINTED OUT*)
98    FORMAT (1H0,*NO DISCRETE SOLUTION FOUND AFTER*,I5,* NODES*)
99    FORMAT (1H0,*NO DISCRETE SOLUTION*)
100   FORMAT (1H0,*ONLY CONTINUOUS SOLUTION HAS BEEN REQUESTED*)
101   FORMAT (1H0,10X,*EXECUTION TIME IN SECONDS =*,F10.5)
```

```
102   FORMAT (1H1)
103   FORMAT (1H0,*THIS IS A DISCRETE SOLUTION*)
104   FORMAT (1H0,*NO OPTIMIZATION HAS BEEN REQUESTED*)
      END
```

```
      SUBROUTINE DSPTA (N,X,G,H,EPS,KR,W,F,ICHECK,IVAR,P2,P1,INDX,GPHI,N
     1R,IAA,IBB,GU,PHI,A,T1,T1P,AL)
C
      DIMENSION W(1), X(1), G(1), H(1), EPS(1), ICHECK(1), IVAR(1), P2(1
     1), P1(1), INDX(1), GPHI(NR,1), IAA(1), IBB(1), GU(1), PHI(1), A(1)
     2, T1(1), T1P(1), AL(1)
      COMMON /DSPT01/ IOPT1,IOPT2,IOPT3,IOPT4,IOPT5,IOPT6,IOPT7,NCONS,ID
     1ATA,IPRINT,MAX,EST,EST1,AO,XMAL,AI,ZERO,ETA,INSOLN,BSOLN,MAXNOD,ER
     2R,ICON
      COMMON /DSPT02/ NOD,KK,NORG,NOR
      COMMON /DSPT03/ AM,PSI,PA,ALPHA,IFLAGA,ICHEK,KKK,INDA,INDB,UR,NC,K
     1O,IFN
      EXTERNAL DSPTC
C
C     THIS SUBROUTINE COORDINATES THE INPUT, THE OUTPUT AND THE
C     MINIMIZATION
C
      DO 1 I=1,NR
      EPS(I)=EPS(I)/(10.**(KR-1))
    1 CONTINUE
      IF (IOPT1.EQ.1.AND.KK.NE.0) GO TO 2
      GO TO 5
    2 N=NOR-1
      IFV=IVAR(NOD)
      IF (IFV.EQ.NOR) GO TO 4
      TE=EPS(IFV)
      DO 3 I=IFV,NOR
      IP1=I+1
      X(I)=X(IP1)
      EPS(I)=EPS(IP1)
      INDX(I)=IP1
    3 CONTINUE
    4 IF (ICHECK(NOD).EQ.0) X(NOR)=P2(NOD)
      IF (ICHECK(NOD).EQ.1.OR.ICHECK(NOD).EQ.2) X(NOR)=P1(NOD)
    5 KO=1
      ICHEK=0
      IF (KK.GT.1) GO TO 6
      ALPHA=AL(1)
      GO TO 7
    6 IF (ICHECK(NOD).NE.0) ALPHA=AL(NOD)
    7 AL(NOD+1)=ALPHA
      IF (IPRINT.GT.0) WRITE (6,17) KR
C
C     SUBROUTINE DSPTB PERFORMS MINIMIZATION BY A VARIABLE METRIC
C     ALGORITHM DUE TO FLETCHER
C
      IF (IOPT6.EQ.4.AND.KR.EQ.1) GO TO 8
      GO TO 9
    8 CALL DSPTC (N,X,F,G,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,IBB,GU,PHI,
     1A,T1,T1P)
      PSI=AMIN1(0.,AM+1.E-10)
    9 ESD=EST
      CALL DSPTB (DSPTC,N,X,ESD,G,H,W,0.,EPS,1,MAX,IPRINT,IFXIT,ICHECK,I
     1VAR,P2,P1,INDX,GPHI,NR,IAA,IBB,GU,PHI,A,T1,T1P)
      ICHEK=1
      KKK=1
C
C     CHECK FEASIBILITY OF CURRENT OPTIMUM SOLUTION
```

```
C
      CALL DSPTC (N,X,F,W,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,IBB,GU,PHI,
     1A,T1,T1P)
      IF (IOPT6.EQ.5) GO TO 10
      IF (IFLAGA.EQ.0.OR.IEXIT.EQ.3.OR.INDA.EQ.1) GO TO 10
      KO=0
      ALPHD=ALPHA*AI
      IF (ALPHD.GT.XMAL) GO TO 10
      ALPHA=ALPHD
      IF (NOD.NE.0) AL(NOD)=ALPHA
      AL(NOD+1)=ALPHA
      KO=1
      GO TO 9
10    IF (IEXIT.EQ.3) KO=0
      IF (INDA.NE.1) GO TO 11
      KO=2
      INDA=0
      IF (IOPT6.EQ.4) CALL DSPTC (N,X,F,W,ICHECK,IVAR,P2,P1,INDX,GPHI,NR
     1,IAA,IBB,GU,PHI,A,T1,T1P)
11    IF (IOPT1.EQ.1.AND.KK.NE.0.AND.IFV.NE.NOR) GO TO 12
      GO TO 15
12    TS=X(NOR)
      JJ=NOR+1
      NI=NOR-IFV
      DO 13 I=1,NI
      J=JJ-I
      X(J)=X(J-1)
      EPS(J)=EPS(J-1)
13    CONTINUE
      X(IFV)=TS
      EPS(IFV)=TE
      IF (IFV.EQ.NOR) GO TO 15
      DO 14 I=IFV,N
      J=NOR+IFV-I
      G(J)=G(J-1)
14    CONTINUE
15    IF (KO.EQ.2.AND.IPRINT.GT.-1) CALL DSPTJ (N,X,F,G,PHI,U,IVAR)
      DO 16 I=1,NR
      EPS(I)=EPS(I)*10.**(KR-1)
16    CONTINUE
      RETURN
C
C
C
17    FORMAT (1H0,*OPTIMIZATION*,I3,/,* ---------------*,/,1X,*ITER*,3X,
     1*FUNCT*,8X,*OBJECTIVE*,6X,*VARIABLE*,7X,*GRADIENT*,/)
      END
```

```
      SUBROUTINE DSPTB (FUNCT,N,X,F,G,H,W,DFN,EPS,MODE,MAXFN,IPRINT,IEXI
     1T,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,IBB,GU,PHI,A,T1,T1P)
C
      DIMENSION X(1), G(1), H(1), W(1), EPS(1), ICHECK(1), IVAR(1), P2(1
     1), P1(1), INDX(1), GPHI(NR,1), IAA(1), IBB(1), GU(1), PHI(1), A(1)
     2, T1(1), T1P(1)
      COMMON /DSPT03/ AM,PSI,PA,XXXXX,IFLAGA,ICHEK,KKK,INDA,INDB,UR,NC,K
     10,IFN
C
C     UNCONSTRAINED MINIMIZATION METHOD
C
C     R. FLETCHER, FORTRAN SUBROUTINES FOR MINIMIZATION BY QUASI-NEWTON
C     METHODS, ATOMIC ENERGY RESEARCH ESTABLISHMENT, HARWELL, BERKSHIRE,
C     ENGLAND, REPORT AERE- R7125, 1972
C
      IF (KKK.NE.0) GO TO 1
      ITN=0
      IFN=1
    1 CONTINUE
      NP=N+1
      N1=N-1
      NN=N*NP/2
      IS=N
      IU=N
      IV=N+N
      IB=IV+N
      IEXIT=0
      IF (MODE.EQ.3) GO TO 7
      IF (MODE.EQ.2) GO TO 4
      IJ=NN+1
      DO 3 I=1,N
      DO 2 J=1,I
      IJ=IJ-1
      H(IJ)=0.
    2 CONTINUE
      H(IJ)=1.
    3 CONTINUE
      GO TO 7
    4 CONTINUE
      IJ=1
      DO 6 I=2,N
      Z=H(IJ)
      IF (Z.LE.0.) RETURN
      IJ=IJ+1
      I1=IJ
      DO 6 J=I,N
      ZZ=H(IJ)
      H(IJ)=H(IJ)/Z
      JK=IJ
      IK=I1
      DO 5 K=I,J
      JK=JK+NP-K
      H(JK)=H(JK)-H(IK)*ZZ
      IK=IK+1
    5 CONTINUE
    6 IJ=IJ+1
      IF (H(IJ).LE.0.) RETURN
    7 CONTINUE
```

```
      IJ=NP
      DMIN=H(1)
      DO 8 I=2,N
      IF (H(IJ).GE.DMIN) GO TO 8
      DMIN=H(IJ)
8     IJ=IJ+NP-I
      IF (DMIN.LE.0.) RETURN
      Z=F
      CALL FUNCT (N,X,F,G,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,IBB,GU,PHI,
     1A,T1,T1P)
      IF (INDB.EQ.1) GO TO 37
      DF=DFN
      IF (DFN.EQ.0.) DF=F-Z
      IF (DFN.LT.0.) DF=ABS(DF*F)
      IF (DF.LE.0.) DF=1.
9     CONTINUE
      IF (IPRINT.LE.0) GO TO 10
      IF (MOD(ITN,IPRINT).NE.0) GO TO 10
      PRINT 38, ITN,IFN,F,((X(I),G(I)),I=1,N)
10    CONTINUE
      ITN=ITN+1
      W(1)=-G(1)
      DO 12 I=2,N
      IJ=I
      I1=I-1
      Z=-G(I)
      DO 11 J=1,I1
      Z=Z-H(IJ)*W(J)
      IJ=IJ+N-J
11    CONTINUE
      W(I)=Z
12    CONTINUE
      W(IS+N)=W(N)/H(NN)
      IJ=NN
      DO 14 I=1,N1
      IJ=IJ-1
      Z=0.
      DO 13 J=1,I
      Z=Z+H(IJ)*W(IS+NP-J)
      IJ=IJ-1
13    CONTINUE
      W(IS+N-I)=W(N-I)/H(IJ)-Z
14    CONTINUE
      GS=0.
      DO 15 I=1,N
      GS=GS+W(IS+I)*G(I)
15    CONTINUE
      IEXIT=2
      IF (GS.GE.0.) GO TO 37
      GS0=GS
      ALPHA=-2.*DF/GS
      IF (ALPHA.GT.1.) ALPHA=1.
      DF=F
      TOT=0.
16    CONTINUE
      IEXIT=3
      IF (IFN.EQ.MAXFN) GO TO 37
      ICON=0
```

```
      IEXIT=1
      DO 17 I=1,N
      Z=ALPHA*W(IS+I)
      IF (ABS(Z).GE.EPS(I)) ICON=1
      X(I)=X(I)+Z
   17 CONTINUE
      CALL FUNCT (N,X,FY,W,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,IBB,GU,PHI
     1,A,T1,T1P)
      IF (INDB.EQ.1) GO TO 37
      IFN=IFN+1
      GYS=0.
      DO 18 I=1,N
      GYS=GYS+W(I)*W(IS+I)
   18 CONTINUE
      IF (FY.GE.F) GO TO 19
      IF (ABS(GYS/GS0).LE..9) GO TO 21
      IF (GYS.GT.0.) GO TO 19
      TOT=TOT+ALPHA
      Z=10.
      IF (GS.LT.GYS) Z=GYS/(GS-GYS)
      IF (Z.GT.10.) Z=10.
      ALPHA=ALPHA*Z
      F=FY
      GS=GYS
      GO TO 16
   19 CONTINUE
      DO 20 I=1,N
      X(I)=X(I)-ALPHA*W(IS+I)
   20 CONTINUE
      IF (ICON.EQ.0) GO TO 37
      Z=3.*(F-FY)/ALPHA+GYS+GS
      ZZ=SQRT(Z**2-GS*GYS)
      Z=1.-(GYS+ZZ-Z)/(2.*ZZ+GYS-GS)
      ALPHA=ALPHA*Z
      GO TO 16
   21 CONTINUE
      ALPHA=TOT+ALPHA
      F=FY
      IF (ICON.EQ.0) GO TO 35
      DF=DF-F
      DGS=GYS-GS0
      LINK=1
      IF (DGS+ALPHA*GS0.GT.0.) GO TO 23
      DO 22 I=1,N
      W(IU+I)=W(I)-G(I)
   22 CONTINUE
      SIG=1./(ALPHA*DGS)
      GO TO 30
   23 CONTINUE
      ZZ=ALPHA/(DGS-ALPHA*GS0)
      Z=DGS*ZZ-1.
      DO 24 I=1,N
      W(IU+I)=Z*G(I)+W(I)
   24 CONTINUE
      SIG=1./(ZZ*DGS**2)
      GO TO 30
   25 CONTINUE
      LINK=2
```

```
      DO 26 I=1,N
      W(IU+I)=G(I)
26    CONTINUE
      IF (DGS+ALPHA*GS0.GT.0.) GO TO 27
      SIG=1./GS0
      GO TO 30
27    CONTINUE
      SIG=-ZZ
      GO TO 30
28    CONTINUE
      DO 29 I=1,N
      G(I)=W(I)
29    CONTINUE
      GO TO 9
30    CONTINUE
      W(IV+1)=W(IU+1)
      DO 32 I=2,N
      IJ=I
      I1=I-1
      Z=W(IU+I)
      DO 31 J=1,I1
      Z=Z-H(IJ)*W(IV+J)
      IJ=IJ+N-J
31    CONTINUE
      W(IV+I)=Z
32    CONTINUE
      IJ=1
      DO 33 I=1,N
      Z=H(IJ)+SIG*W(IV+I)**2
      IF (Z.LE.0.) Z=DMIN
      IF (Z.LT.DMIN) DMIN=Z
      H(IJ)=Z
      W(IB+I)=W(IV+I)*SIG/Z
      SIG=SIG-W(IB+I)**2*Z
      IJ=IJ+NP-I
33    CONTINUE
      IJ=1
      DO 34 I=1,N1
      IJ=IJ+1
      I1=I+1
      DO 34 J=I1,N
      W(IU+J)=W(IU+J)-H(IJ)*W(IV+I)
      H(IJ)=H(IJ)+W(IB+I)*W(IU+J)
34    IJ=IJ+1
      GO TO (25,28), LINK
35    CONTINUE
      DO 36 I=1,N
      G(I)=W(I)
36    CONTINUE
37    CONTINUE
      IF (IPRINT.LE.0) RETURN
      PRINT 38, ITN,IFN,F,((X(I),G(I)),I=1,N)
      IF (INDB.EQ.1) RETURN
      PRINT 39, IEXIT
      IF (IEXIT.EQ.1) PRINT 40
      IF (IEXIT.EQ.2) PRINT 41
      IF (IEXIT.EQ.3) PRINT 42
C
```

```
      RETURN
C
C
C
38    FORMAT (1H ,I4,3X,I4,6X,E14.6,1X,80(E14.6,1X,E14.6,/,33X))
39    FORMAT (1H0,*IEXIT =*,I5)
40    FORMAT (1H0,*NORMAL EXIT*)
41    FORMAT (1H0,*EPS IS PROBABLY SET TOO SMALL*)
42    FORMAT (1H0,*PERMISSIBLE NUMBER OF FUNCTION EVALUATIONS EXCEEDED*)
      END
```

```
      SUBROUTINE DSPTC (N,Y,F,G,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,IBB,G
     1U,PHI,A,T1,T1P)
C
      DIMENSION Y(1), G(1), IAA(1), IBB(1), GU(1), PHI(1), GPHI(NR,1), I
     1CHECK(1), IVAR(1), P2(1), P1(1), A(1), T1(1), T1P(1), INDX(1)
      COMMON /DSPT01/ IOPT1,IOPT2,IOPT3,IOPT4,IOPT5,IOPT6,IOPT7,NCONS,ID
     1ATA,IPRINT,MAX,EST,EST1,AO,XMAL,AI,ZERO,ETA,INSOLN,BSOLN,MAXNOD,ER
     2R,ICON
      COMMON /DSPT02/ NOD,KK,NORG,NOR
      COMMON /DSPT03/ AM,PSI,PA,ALPHA,IFLAGA,ICHEK,KKK,INDA,INDB,UR,NC,K
     10,IFN
      COMMON /DSPT04/ SUMD,INDC
C
C     THIS SUBROUTINE COORDINATES SUBROUTINES DSPTD,DSPTE,DSPTF,DSPTG
C     AND DSPTH TO FORMULATE THE ARTIFICIAL UNCONSTRAINED OBJECTIVE
C     FUNCTION AND THE NECESSARY GRADIENTS
C
      IDD=0
      IFV=IVAR(NOD)
      IF (IOPT1.EQ.1.AND.KK.GT.0.AND.IFV.NE.NOR) GO TO 1
      GO TO 3
    1 IDD=1
      TS=Y(NOR)
      NI=NOR-IFV
      JJ=NOR+1
      DO 2 I=1,NI
      J=JJ-I
      Y(J)=Y(J-1)
    2 CONTINUE
      Y(IFV)=TS
    3 CALL DSPTF (Y,PHI,U,N,GU,INDX,GPHI,NR,1)
      CALL DSPTD (Y,PHI,IAA,IBB,ICHECK,IVAR,P2,P1)
      IF (KK.EQ.0.OR.INDC.EQ.1) GO TO 4
      IF (IOPT1.EQ.1.AND.NOD.LE.1) GO TO 4
      CALL DSPTE (IAA,IBB,GPHI,NR,IVAR)
    4 IF (IOPT1.NE.1.OR.KK.EQ.0.OR.IFV.EQ.NOR) GO TO 8
      IF (INDC.EQ.1) NCM1=NORG
      IF (INDC.NE.1) NCM1=NCONS-1
      DO 6 J=1,NCM1
      DO 5 I=IFV,N
      GPHI(I,J)=GPHI(I+1,J)
    5 CONTINUE
    6 CONTINUE
      DO 7 I=IFV,N
      GU(I)=GU(I+1)
    7 CONTINUE
    8 CALL DSPTH (U,GU,PHI,N,F,G,A,T1,T1P,NR,GPHI)
      IF (IDD.EQ.0) GO TO 10
      TTS=Y(IFV)
      DO 9 I=IFV,N
      Y(I)=Y(I+1)
    9 CONTINUE
      Y(NOR)=TTS
   10 IF (KK.NE.0) INDC=1
      RETURN
      END
```

```
      SUBROUTINE DSPTD (X,PHI,IAA,IBB,ICHECK,IVAR,P2,P1)
C
      DIMENSION X(1), PHI(1), IAA(1), IBB(1), ICHECK(1), IVAR(1), P2(1),
     1 P1(1)
      COMMON /DSPT01/ IOPT1,IOPT2,IOPT3,IOPT4,IOPT5,IOPT6,IOPT7,NCONS,ID
     1ATA,IPRINT,MAX,EST,EST1,AO,XMAL,AI,ZERO,ETA,INSOLN,BSOLN,MAXNOD,ER
     2R,ICON
      COMMON /DSPT02/ NOD,KK,NORG,NOR
C
C     THIS SUBROUTINE RETURNS ADDITIONAL PARAMETER CONSTRAINTS FOR
C     DISCRETE VALUE OPTIMIZATION
C
      IF (NOD.EQ.0) GO TO 4
      IF (NOD.EQ.1.AND.IOPT1.EQ.1) GO TO 4
      MN=NOD
      IF (IOPT1.EQ.1) MN=NOD-1
      DO 3 I=1,MN
      L=IVAR(I)
      II=I+NORG
      IF (L.EQ.IVAR(NOD).AND.IOPT1.EQ.1) GO TO 2
      IF (ICHECK(I).EQ.0) GO TO 1
      PHI(II)=P1(I)-X(L)
      IAA(II)=-1
      IBB(II)=L
      GO TO 3
    1 PHI(II)=(X(L)-P2(I))
      IAA(II)=1
      IBB(II)=L
      GO TO 3
    2 PHI(II)=1.E+10
      IAA(II)=0
    3 CONTINUE
    4 RETURN
      END
```

```
      SUBROUTINE DSPTE (IAA,IBB,GPHI,NR,IVAR)
C
      DIMENSION IAA(1), IBB(1), GPHI(NR,1), IVAR(1)
      COMMON /DSPT01/ IOPT1,IOPT2,IOPT3,IOPT4,IOPT5,IOPT6,IOPT7,NCONS,ID
     1ATA,IPRINT,MAX,EST,EST1,AO,XMAL,AI,ZERO,ETA,INSOLN,BSOLN,MAXNOD,ER
     2R,ICON
      COMMON /DSPT02/ NOD,KK,NORG,NOR
C
C     THIS SUBROUTINE RETURNS THE GRADIENTS OF THE ADDITIONAL PARAMETER
C     CONSTRAINTS FOR DISCRETE VALUE OPTIMIZATION
C
      NORGP=NORG+1
      MN=NCONS
      IF (IOPT1.EQ.1) MN=NCONS-1
      DO 3 J=1,NOR
      IF (IOPT1.EQ.1.AND.J.EQ.IVAR(NOD)) GO TO 3
      DO 2 I=NORGP,MN
      IF (IBB(I).NE.J) GO TO 1
      GPHI(J,I)=IAA(I)
      GO TO 2
1     GPHI(J,I)=0.
2     CONTINUE
3     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DSPTH (U,GU,PHI,N,F,G,A,T1,T1P,NR,GPHI)
C
      DIMENSION GU(1), PHI(1), G(1), A(1), T1(1), T1P(1), GPHI(NR,1)
      COMMON /DSPT01/ IOPT1,IOPT2,IOPT3,IOPT4,IOPT5,IOPT6,IOPT7,NCONS,ID
     1ATA,IPRINT,MAX,EST,EST1,AO,XMAL,AI,ZERO,ETA,INSOLN,BSOLN,MAXNOD,ER
     2R,ICON
      COMMON /DSPT02/ NOD,KK,NORG,NOR
      COMMON /DSPT03/ AMD,PSI,PA,ALPHA,IFLAGA,ICHEK,KKK,INDA,INDB,UR,NC,
     1KO,IFN
      COMMON /DSPT04/ SUMD,INDC
C
C     THIS SUBROUTINE TRANSFORMS THE CONSTRAINED PROBLEM INTO AN
C     UNCONSTRAINED OBJECTIVE USING THE BANDLER-CHARALAMBOUS TECHNIQUE
C     OR A MODIFIED NON-PARAMETRIC EXTERIOR-POINT METHOD
C
      EPSPHI=-ZERO
      P=PA
      AE=0.
      UR=U
      NC=NCONS
      IFLAGA=0
      INDB=0
      IF (NCONS.EQ.0.OR.ALPHA.EQ.0.) GO TO 23
      NCM=NCONS
      IF (IOPT1.EQ.1.AND.KK.GT.0) NCM=NCONS-1
      IF (NCM.EQ.0) GO TO 23
      NT=NCM+1
      IF (INDA.EQ.1.OR.IOPT6.EQ.5) GO TO 2
      V=(U-PSI)/ALPHA
      DO 1 I=1,NCM
      A(I)=V-PHI(I)
    1 CONTINUE
      A(NT)=V
      GO TO 6
    2 DO 3 I=1,NCM
      A(I)=-PHI(I)
    3 CONTINUE
      IF (INSOLN.EQ.1.AND.INDA.EQ.1) GO TO 4
      A(NT)=-1.E+20
      GO TO 5
    4 IF (IOPT7.EQ.1) A(NT)=U-(BSOLN*(1.-SIGN(ETA,BSOLN))+ZERO)
      IF (IOPT7.NE.1) A(NT)=U-BSOLN
    5 IF (IOPT6.EQ.5.AND.INDA.NE.1) A(NT)=U-PSI
    6 AM=A(1)
      DO 7 I=2,NT
      AM=AMAX1(AM,A(I))
    7 CONTINUE
      IF (INDA.NE.1.AND.IOPT6.NE.5) AMD=AM*ALPHA
      IF (INDA.EQ.1.AND.AM.LE.EPSPHI) INDB=1
      IF (AM.LE.0.) P=-PA
      SUM1=0.
      DO 11 I=1,NT
      IF (AM) 10,8,9
    8 AE=1.E-10
      GO TO 10
    9 IF (A(I).LE.0.) GO TO 11
   10 T1(I)=(A(I)-AE)/(AM-AE)
      T1P(I)=T1(I)**P
```

```
      SUM1=SUM1+T1P(I)
 11   CONTINUE
      SUM3=SUM1**(1./P)
      F=(AM-AE)*SUM3
      IF (IOPT6.NE.5.AND.INDA.NE.1) F=F*ALPHA
      SUMD=F
      DO 22 I=1,N
      SUM2=0.
      IF (INDA.EQ.1.OR.IOPT6.EQ.5) GO TO 16
      DO 15 J=1,NT
      IF (AM) 13,13,12
 12   IF (A(J).LE.0.) GO TO 15
 13   IF (J.EQ.NT) GO TO 14
      SUM2=SUM2+T1P(J)/T1(J)*(GU(I)/ALPHA-GPHI(I,J))
      GO TO 15
 14   SUM2=SUM2+T1P(J)/T1(J)*GU(I)/ALPHA
 15   CONTINUE
      SUM2=SUM2*ALPHA
      GO TO 21
 16   DO 20 J=1,NT
      IF (AM) 18,18,17
 17   IF (A(J).LE.0.) GO TO 20
 18   IF (J.EQ.NT) GO TO 19
      SUM2=SUM2-T1P(J)/T1(J)*GPHI(I,J)
      GO TO 20
 19   IF (IOPT6.NE.5.AND.INSOLN.NE.INDA) GO TO 20
      SUM2=SUM2+T1P(J)/T1(J)*GU(I)
 20   CONTINUE
 21   G(I)=SUM3/SUM1*SUM2
 22   CONTINUE
      GO TO 25
 23   F=U
      DO 24 I=1,N
      G(I)=GU(I)
 24   CONTINUE
 25   IF (ICHEK.EQ.0.OR.NCONS.EQ.0.OR.NCM.EQ.0) RETURN
      DO 26 I=1,NCM
      IF (PHI(I).LT.ZERO) IFLAGA=1
 26   CONTINUE
      IF (INDA.NE.1) RETURN
      IF (A(NT).LE.EPSPHI) RETURN
      IFLAGA=1
      RETURN
      END
```

```
      SUBROUTINE DSPTI (K,N,EPS,X,PS,NP,QSTEP,NSTEP,DISCR)
C
      COMMON /DSPT01/ IOPT1,IOPT2,IOPT3,IOPT4,IOPT5,IOPT6,IOPT7,NCONS,ID
     1ATA,IPRINT,MAX,EST,EST1,AO,XMAL,AI,ZERO,ETA,INSOLN,BSOLN,MAXNOD,ER
     2R,ICON
      DIMENSION X(1), EPS(1), PS(1), QSTEP(1), NSTEP(1), DISCR(K,1)
C
C     THIS SUBROUTINE PRINTS THE INPUT DATA
C
      WRITE (6,6)
      IF (IOPT5.NE.1) GO TO 2
      WRITE (6,7)
      DO 1 I=1,K
      NS=NSTEP(I)
      WRITE (6,8) (DISCR(I,J),J=1,NS)
    1 CONTINUE
    2 WRITE (6,9) N
      IF (MAXNOD.NE.0) WRITE (6,19) K
      WRITE (6,39) NCONS
      WRITE (6,10) MAX
      WRITE (6,18) MAXNOD
      WRITE (6,11) IPRINT
      WRITE (6,12) X(1)
      WRITE (6,13) (I,X(I),I=2,N)
      IF (IOPT6.NE.0) WRITE (6,21) ZERO
      IF (MAXNOD.EQ.0) GO TO 3
      WRITE (6,22) ERR
      IF (IOPT5.EQ.1) GO TO 3
      WRITE (6,44) QSTEP(1)
      IF (K.LT.2) GO TO 3
      WRITE (6,45) (I,QSTEP(I),I=2,K)
    3 WRITE (6,14) EPS(1)
      WRITE (6,15) (I,EPS(I),I=2,N)
      WRITE (6,16) EST
      IF (IOPT6.EQ.0) RETURN
      IF (IOPT6.EQ.5) WRITE (6,43) EST1
      IF (INSOLN.EQ.1) WRITE (6,40) BSOLN
      IF (IOPT6.EQ.5) GO TO 4
      WRITE (6,17) AO
      WRITE (6,20) XMAL
      WRITE (6,23) AI
    4 IF (IOPT6.EQ.4.OR.IOPT6.EQ.2.OR.IOPT6.EQ.5) WRITE (6,30) ETA
      WRITE (6,34) NP
      WRITE (6,35) PS(1)
      IF (NP.LT.2) GO TO 5
      WRITE (6,36) (I,PS(I),I=2,NP)
    5 WRITE (6,24)
      IF (IOPT6.EQ.4) WRITE (6,31)
      IF (IOPT6.EQ.1) WRITE (6,33)
      IF (IOPT6.EQ.2) WRITE (6,32)
      IF (IOPT6.EQ.5) WRITE (6,38)
      IF (IOPT6.EQ.3) WRITE (6,41)
      IF (MAXNOD.EQ.0) RETURN
      IF (IOPT1.EQ.1) WRITE (6,25)
      IF (IOPT3.EQ.1) WRITE (6,26)
      IF (IOPT4.EQ.1) WRITE (6,27)
      IF (IOPT4.EQ.2) WRITE (6,37)
      IF (IOPT7.EQ.1) WRITE (6,42)
```

```
      IF (ICON.EQ.1) WRITE (6,28)
      IF (ICON.NE.1) WRITE (6,29)
      RETURN
C
C
C
6     FORMAT (1H1,*INPUT DATA*,/,1X,10(*-*),//)
7     FORMAT (* DISCRETE VALUES FOR THE VARIABLES*)
8     FORMAT (1H0,5E16.8)
9     FORMAT (1H0,*NUMBER OF INDEPENDENT VARIABLES*,35(*.*),*NR=*,I5)
10    FORMAT (1H0,*MAXIMUM NUMBER OF ALLOWABLE FUNCTION EVALUATIONS*,17(
     1*.*),*MAX=*,I5)
11    FORMAT (1H0,*INTERMEDIATE OUTPUT TO BE PRINTED EVERY IPRINT ITERAT
     1IONS*,5(*.*),*IPRINT=*,I5)
12    FORMAT (1H0,*STARTING VALUE FOR VECTOR X(I)*,33(*.*),*X( 1)=*,E16.
     18)
13    FORMAT (64X,*X(*,I2,*)=*,E16.8)
14    FORMAT (1H0,*TEST QUANTITIES TO BE USED IN FLETCHER METHOD*,16(*.*
     1),*EPS( 1)=*,E16.8)
15    FORMAT (62X,*EPS(*,I2,*)=*,E16.8)
16    FORMAT (1H0,*ESTIMATE OF LOWER BOUND ON ARTIFICIAL OBJECTIVE FUNCT
     1ION*,9(*.*),*EST=*,E16.8)
17    FORMAT (1H0,*INITIAL VALUE OF THE PARAMETER ALPHA*,30(*.*),*AO=*,E
     116.8)
18    FORMAT (1H0,*MAXIMUM NUMBER OF NODES TO BE SEARCHED*,24(*.*),*MAXN
     1OD=*,I5)
19    FORMAT (1H0,*NUMBER OF DISCRETE VARIABLES*,39(*.*),*K=*,I5)
20    FORMAT (1H0,*MAXIMUM ALLOWABLE VALUE OF THE PARAMETER ALPHA*,18(*.
     1*),*XMAL=*,E16.8)
21    FORMAT (1H0,*ERROR TOLERANCE IN CONSTRAINTS*,34(*.*),*ZERO=*,E16.8
     1)
22    FORMAT (1H0,*ERROR TOLERANCE IN DISCRETE VALUES*,31(*.*),*ERR=*,E1
     16.8)
23    FORMAT (1H0,*MULTIPLYING FACTOR IN ALPHA VALUE*,33(*.*),*AI=*,E16.
     18)
24    FORMAT (1H0,*FOLLOWING OPTIONS USED*,/,1X,22(*-*),/)
25    FORMAT (1H0,*(N-1) VARIABLE OPTIMIZATION PERFORMED*)
26    FORMAT (1H0,*VERTICES CHECKED*)
27    FORMAT (1H0,*FEASIBILITY CHECKED*)
28    FORMAT (1H0,*PARTITIONING STARTS ON FIRST DISCRETE VARIABLE*)
29    FORMAT (1H0,*PARTITIONING STARTS ON LAST DISCRETE VARIABLE*)
30    FORMAT (1H0,*TEST QUANTITY TO BE USED IN NLP ALGORITHM 2/4/5*,18(*
     1.*),*ETA=*,E16.8)
31    FORMAT (1H0,*NLP ALGORITHM 4 EMPLOYED*)
32    FORMAT (1H0,*NLP ALGORITHM 2 EMPLOYED*)
33    FORMAT (1H0,*NLP ALGORITHM 1 EMPLOYED*)
34    FORMAT (1H0,*NUMBER OF P VALUES*,48(*.*),*NP=*,I5)
35    FORMAT (1H0,*VALUE(S) OF P USED IN NLP ALGORITHM*,27(*.*),*PS( 1)=
     1*,E16.8)
36    FORMAT (63X,*PS(*,I2,*)=*,E16.8)
37    FORMAT (1H0,*FEASIBILITY CHECKED FOR DISCRETE PROBLEM ONLY*)
38    FORMAT (1H0,*NLP ALGORITHM 5 EMPLOYED*)
39    FORMAT (1H0,*NUMBER OF CONSTRAINTS ON THE CONTINUOUS PROBLEM*,16(*
     1.*),*NCONS=*,I5)
40    FORMAT (1H0,*UPPER BOUND ON ARTIFICIAL OBJECTIVE FUNCTION*,21(*.*)
     1,*IDS=*,E16.8)
41    FORMAT (1H0,*NLP ALGORITHM 3 EMPLOYED*)
42    FORMAT (1H0,*ONLY ONE OPTIMUM DISCRETE SOLUTION REQUIRED*)
```

```
43    FORMAT (1H0,*ESTIMATE OF LOWER BOUND ON ACTUAL OBJECTIVE FUNCTION*
     1,12(*.*),*EST1=*,F16.8)
44    FORMAT (1H0,*QUANTIZATION STEP SIZES FOR THE DISCRETE VARIABLES*,9
     1(*.*),*QSTEP( 1)=*,E16.8)
45    FORMAT (60X,*QSTEP(*,I2,*)=*,E16.8)
      END
```

```
      SUBROUTINE DSPTJ (N,X,F,G,PHI,U,IVAR)
C
      COMMON /DSPT01/ IOPT1,IOPT2,IOPT3,IOPT4,IOPT5,IOPT6,IOPT7,NCONS,ID
     1ATA,IPRINT,MAX,EST,EST1,AO,XMAL,AI,ZERO,ETA,INSOLN,BSOLN,MAXNOD,ER
     2R,ICON
      COMMON /DSPT02/ NOD,KK,NORG,NOR
      COMMON /DSPT03/ AM,PSI,PA,ALPHA,IFLAGA,ICHEK,KKK,INDA,INDB,UR,NC,K
     1O,NUMF
      DIMENSION X(1), G(1), PHI(1), IVAR(1)
C
C     THIS SUBROUTINE OUTPUTS THE SOLUTION AT EACH NODE
C
      IF (IOPT1.EQ.1.AND.KK.GT.0) GO TO 1
      NCM=NC
      NM=N
      GO TO 2
1     NCM=NC-1
      NM=N+1
2     NVIOL=0
      IF (KO.EQ.0) WRITE (6,16)
      IF (KO.EQ.1) WRITE (6,15)
      IF (KO.EQ.2) WRITE (6,18)
      WRITE (6,22) KK
      IF (KO.EQ.2) GO TO 7
      IF (IOPT6.NE.3) WRITE (6,17) F
      WRITE (6,10) U
      IF (IOPT6.NE.3) GO TO 4
      DO 3 I=1,NM
      WRITE (6,23) I,X(I)
3     CONTINUE
      GO TO 7
4     DO 6 I=1,NM
      IF (IOPT1.EQ.1.AND.I.EQ.IVAR(NOD)) GO TO 5
      WRITE (6,19) I,X(I),I,G(I)
      GO TO 6
5     WRITE (6,20) I,X(I)
6     CONTINUE
7     IF (NCM.EQ.0) GO TO 9
      DO 8 I=1,NCM
      IF (PHI(I).LT.ZERO) NVIOL=NVIOL+1
8     CONTINUE
      WRITE (6,11)
      WRITE (6,12) (I,PHI(I),I=1,NCM)
      WRITE (6,21) NVIOL
9     WRITE (6,13) NUMF
      IF (KO.EQ.2) RETURN
      IF (IOPT6.NE.5) WRITE (6,14) ALPHA
      RETURN
C
C
C
10    FORMAT (1H0,8X,*ACTUAL OBJECTIVE FUNCTION F =*,E16.8,/)
11    FORMAT (1H0,7X,*INEQUALITY CONSTRAINTS*,/)
12    FORMAT (32X,*G(*,I2,*)=*,E16.8)
13    FORMAT (1H0,5X,*NUMBER OF FUNCTION EVALUATIONS =*,I5)
14    FORMAT (1H0,1X,*FINAL VALUE OF THE PARAMETER ALPHA =*,E16.8)
15    FORMAT (1H1,11X,*FOLLOWING IS THE OPTIMUM SOLUTION*,/,12X,*-------
     1-------------------------*)
```

```
16     FORMAT (1H1,15X,*RESULTS AT LAST ITERATION*/,16X,*---------------
      1---------*)
17     FORMAT (1H0,*ARTIFICIAL UNCONSTRAINED FUNCTION U =*,E16.8)
18     FORMAT (1H1,10X,*RESULTS OF THE FEASIBILITY CHECK*,/,11X,32(*-*))
19     FORMAT (8X,*X(*,I2,*)=*,E16.8,1X,*GU(*,I2,*)=*,E16.8)
20     FORMAT (8X,*X(*,I2,*)=*,E16.8)
21     FORMAT (1H0,5X,*NUMBER OF VIOLATED CONSTRAINTS =*,I5)
22     FORMAT (1H0,24X,*NODE NUMBER =*,I5)
23     FORMAT (32X,*X(*,I2,*)=*,E16.8)
       END
```

```
      SUBROUTINE DSPTK (U,X,PHI,KO,IFNT,IAA,IBB,N,GU,INDX,GPHI,NR)
C
      DIMENSION X(1), PHI(1), IAA(1), IBB(1), GU(1), INDX(1), GPHI(NR,1)
      COMMON /DSPT01/ IOPT1,IOPT2,IOPT3,IOPT4,IOPT5,IOPT6,IOPT7,NCONS,ID
     1ATA,IPRINT,MAX,EST,EST1,AO,XMAL,AI,ZERO,ETA,INSOLN,BSOLN,MAXNOD,ER
     2R,ICON
      COMMON /DSPT02/ NOD,KK,NORG,NOR
C
C     THIS SUBROUTINE OUTPUTS THE FINAL SOLUTION IN A STANDARD FORM
C
      WRITE (6,11)
      IF (KO.EQ.0) GO TO 1
      WRITE (6,4)
      WRITE (6,5) U
      GO TO 2
1     WRITE (6,6)
      WRITE (6,7) U
2     WRITE (6,8) (I,X(I),I=1,NOR)
      IF (NORG.EQ.0) GO TO 3
      CALL DSPTF (X,PHI,U,N,GU,INDX,GPHI,NR,0)
      IFNT=IFNT+1
      WRITE (6,9)
      WRITE (6,10) (I,PHI(I),I=1,NORG)
3     WRITE (6,12) IFNT
      RETURN
C
C
C
4     FORMAT (1H-,10X,35HBEST DISCRETE SOLUTION FOUND SO FAR,/)
5     FORMAT (30X,3HF =,E16.8//)
6     FORMAT (1H0,13X,31HOPTIMUM DISCRETE SOLUTION FOUND,/)
7     FORMAT (21X,12HMINIMUM  F =,E16.8//)
8     FORMAT (26X,2HX(,I2,3H) =,E16.8)
9     FORMAT (1H-,22HINEQUALITY CONSTRAINTS)
10    FORMAT (26X,2HG(,I2,3H) =,E16.8)
11    FORMAT (1H1)
12    FORMAT (1H0,32HNUMBER OF FUNCTION EVALUATIONS =,I5)
      END
```

```
      SUBROUTINE DSPTL (N,X,G,PY,Y,IPRINT,ICHECK,IVAR,P2,P1,INDX,GPHI,NR
     1,IAA,IBB,GU,PHI,A,T1,T1P)
C
      DIMENSION X(1), G(1), PY(1), Y(1), ICHECK(1), IVAR(1), P2(1), P1(1
     1), INDX(1), GPHI(NR,1), IAA(1), IBB(1), GU(1), PHI(1), A(1), T1(1)
     2, T1P(1)
C
C     THIS SUBROUTINE CHECKS THE ANALYTICAL PARTIAL DERIVATIVE FORMULA-
C     TION AT THE STARTING POINT BY NUMERICAL PERTURBATION
C
      CALL DSPTC (N,X,F,G,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,IBB,GU,PHI,
     1A,T1,T1P)
      DO 1 I=1,N
      Z=X(I)
      DELX=1.E-4*X(I)
      IF (ABS(X(I)).LT.1.E-10) DELX=1.E-10
      X(I)=Z+DELX
      CALL DSPTC (N,X,F2,PY,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,IBB,GU,PH
     1I,A,T1,T1P)
      X(I)=Z-DELX
      CALL DSPTC (N,X,F1,PY,ICHECK,IVAR,P2,P1,INDX,GPHI,NR,IAA,IBB,GU,PH
     1I,A,T1,T1P)
      Y(I)=0.5*(F2-F1)/DELX
      X(I)=Z
    1 CONTINUE
      DO 2 I=1,N
      IF (ABS(Y(I)).LT.1.E-20) Y(I)=1.E-20
      IF (ABS(G(I)).LT.1.E-20) G(I)=1.E-20
      PY(I)=ABS((Y(I)-G(I))/Y(I))*100.
    2 CONTINUE
      IF (IPRINT.LT.-1) GO TO 3
      WRITE (6,6)
      WRITE (6,7)
      WRITE (6,8) (I,X(I),I=1,N)
      WRITE (6,9)
      WRITE (6,10) (G(I),Y(I),PY(I),I=1,N)
    3 DO 4 I=1,N
      IF (PY(I).GT.10.) GO TO 5
    4 CONTINUE
      IF (IPRINT.GE.-1) WRITE (6,11)
      RETURN
    5 WRITE (6,12)
      CALL EXIT
C
C
C
    6 FORMAT (1H1)
    7 FORMAT (1H0,5X,*GRADIENTS CHECKING*,/,6X,18(*-*),//,6X,*GRADIENTS
     1HAVE BEEN CHECKED AT THE FOLLOWING POINT*/)
    8 FORMAT (10X,*X(*,I2,*)=*,E16.8)
    9 FORMAT (///,1H0,5X,*ANALYTICAL GRADIENTS*,5X,*NUMERICAL GRADIENTS*
     1,7X,*PERCENTAGE ERROR*,/)
   10 FORMAT (6X,E16.8,9X,E16.8,9X,E16.8)
   11 FORMAT (1H0,//,6X,*GRADIENTS ARE O. K.*)
   12 FORMAT (1H0,//,6X,*YOUR PROGRAM HAS BEEN TERMINATED BECAUSE GRADIE
     1NTS ARE INCORRECT*,/6X,*PLEASE CHECK IT AGAIN*)
      END
```

```
      SUBROUTINE DSPTM (N,X,XE,IH,IK,RF,X1,JORDER)
C
      DIMENSION X(1), XE(N,IK,1), X1(1)
C
C     THIS SUBROUTINE EXTRAPOLATES ON THE VARIABLES TO ACCELERATE THE
C     CONVERGENCE IN ALGORITHM 3
C
C     A.V. FIACCO AND G.P. MCCORMICK, NONLINEAR PROGRAMMING- SEQUENTIAL
C     UNCONSTRAINED MINIMIZATION TECHNIQUES. NEW YORK- WILEY, 1968
C
      I=IH
      II=I+1
      DO 1 J=1,N
      XE(J,I,1)=X(J)
1     CONTINUE
      IF (I.LT.2) GO TO 11
      IF (I.GT.JORDER) GO TO 2
      IJ=I
      GO TO 3
2     IJ=JORDER+1
3     DO 5 L=2,IJ
      LL=L-1
      S=RF**LL
C
C     ESTIMATE OF THE ULTIMATE SOLUTION
C
      DO 4 J=1,N
      XE(J,I,L)=(S*XE(J,I,LL)-XE(J,I-1,LL))/(S-1.0)
4     CONTINUE
5     CONTINUE
      DO 6 J=1,N
      X1(J)=XE(J,I,IJ)
6     CONTINUE
      IF (I.EQ.IK) RETURN
C
C     ESTIMATE OF THE NEXT STARTING POINT
C
      DO 7 J=1,N
      XE(J,II,IJ)=XE(J,I,IJ)
7     CONTINUE
      DO 9 K=2,IJ
      L=IJ+1-K
      SS=RF**L
      DO 8 J=1,N
      XE(J,II,L)=((SS-1.)*XE(J,II,L+1)+XE(J,I,L))/SS
8     CONTINUE
9     CONTINUE
      DO 10 J=1,N
      X(J)=XE(J,II,1)
10    CONTINUE
      RETURN
11    DO 12 J=1,N
      X1(J)=XE(J,I,1)
12    CONTINUE
      RETURN
      END
```

REFERENCES

[1] J.W. Bandler and C. Charalambous, "Nonlinear programming using minimax techniques", J. Optimization Theory and Applications, vol. 13, pp. 607-619, June 1974.

[2] A.V. Fiacco and G.P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques. New York: Wiley, 1968.

[3] F.A. Lootsma, "A survey of methods for solving constrained minimization problems via unconstrained minimization", in Numerical Methods for Nonlinear Optimization, F.A. Lootsma, Ed. New York: Academic Press, 1972.

[4] R.J. Dakin, "A tree-search algorithm for mixed integer programming problems", Computer J., vol. 8, pp. 250-255, 1966.

[5] R. Fletcher, "FORTRAN subroutines for minimization by quasi-Newton methods", Atomic Energy Research Establishment, Harwell, Berkshire, England, Report AERE-R7125, 1972.

[6] J.W. Bandler and C. Charalambous, "Practical least pth optimization of networks", IEEE Trans. Microwave Theory Tech., vol. MTT-20, pp. 834-840, Dec. 1972.

[7] C. Charalambous and J.W. Bandler, "New algorithms for network optimization", IEEE Trans. Microwave Theory Tech., vol. MTT-21, pp. 815-818, Dec. 1973.

[8] C. Charalambous and J.W. Bandler, "Nonlinear minimax optimization as a sequence of least pth optimization with finite values of p", McMaster University, Hamilton, Canada, Internal Report in Simulation, Optimization and Control, No. SOC-3, June 1973.

[9] J.W. Bandler, "New directions in nonlinear programming for circuit design", Proc. 16th Midwest Symp. on Circuit Theory (Waterloo, Canada, April 1973), vol. 1, pp. VI.3.1-VI.3.10.

[10] J. Kowalik and M.R. Osborne, Methods for Unconstrained Optimization Problems. New York: Elsevier, 1968.

[11] J.W. Bandler, "Optimization of design tolerances using nonlinear programming", J. Optimization Theory and Applications, vol. 14, pp. 99-114, July 1974.

[12] J.W. Bandler and P.C. Liu, "Automated network design with optimal tolerances", IEEE Trans. Circuit Theory, vol. CT-21, pp. 219-222, March 1974.

[13] R.S. Garfinkel and G.L. Nemhauser, Integer Programming. New York: Wiley, 1972.

[14] H.A. Taha, Operations Research - An Introduction. New York: MacMillan, 1971.

[15] B.J. Karafin, "The optimum assignment of component tolerances for electrical networks", B.S.T.J., vol. 50, pp. 1225-1242, April 1972.

[16] J.W. Bandler and J.H.K. Chen, "DISOPT - a general program for continuous and discrete nonlinear programming problems", 8th Princeton Conf. on Information Sciences and Systems (Princeton, N.J., March 1974).

[17] J.W. Bandler, P.C. Liu and J.H.K. Chen, "Computer-aided tolerance optimization applied to microwave circuits", IEEE Int. Microwave Symp. Digest (Atlanta, Georgia, June 1974), pp. 275-277.

[18] J.W. Bandler, B.L. Bardakjian and J.H.K. Chen, "Design of recursive digital filters with optimum word length coefficients", 8th Princeton Conf. on Information Sciences and Systems (Princeton, N.J., March 1974).

[19] C. Charalambous, "A negative-positive barrier method for nonlinear programming", University of Waterloo, Waterloo, Canada, Department of Combinatorics and Optimization, Research Report 74-2, January 1974.

SOC-29

DISOPT - A GENERAL PROGRAM FOR CONTINUOUS AND DISCRETE NONLINEAR PROGRAMMING PROBLEMS

J.H.K. Chen

March 1974, No. of Pages: 80

Revised: June 1975

Key Words: Engineering design, optimization programs, discrete optimization, tolerance assignment

Abstract: An integrated computer program in FORTRAN IV for continuous or discrete, constrained or unconstrained general optimization problems is presented. The program, called DISOPT, is applicable to a wide variety of design problems such as continuous and discrete tolerance assignments, digital filter design, circuit design, system modelling and approximation problems. Many recent techniques and algorithms for nonlinear programming have been incorporated. The user may optionally choose the combination of techniques and algorithms best suited to his problems.

Description: M. Eng. Thesis.
Contains Fortran listing, user´s manual.
Source deck available for $100.00.

Related Work: SOC-1, SOC-2, SOC-3, SOC-4, SOC-13, SOC-18, SOC-24, SOC-36, SOC-49, SOC-62, SOC-65, SOC-71, SOC-80, SOC-87, SOC-113. Supercedes SOC-4.

Price: $30.00.