OPTIMIZATION OF INDUSTRIAL SYSTEMS WITH THE SEPARABLE PROGRAMMING AND THE GENERALIZED REDUCED GRADIENT METHODS

by

JEREL L. WILLIAMS

B.A., KANSAS STATE TEACHERS COLLEGE, EMPORIA, 1970

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1972

Approved by:

Ching-Lai Hwang
Major Professor

THIS BOOK CONTAINS NUMEROUS PAGES WITH THE ORIGINAL PRINTING BEING SKEWED DIFFERENTLY FROM THE TOP OF THE PAGE TO THE BOTTOM.

THIS IS AS RECEIVED FROM THE CUSTOMER.

THIS BOOK IS OF POOR LEGIBILITY DUE TO LIGHT PRINTING THROUGH OUT IT'S ENTIRETY.

THIS IS AS RECEIVED FROM THE CUSTOMER.

ACKNOWLEDGEMENT

The author would like to express deep appreciation to his major professor, Dr. C.L. Hwang, for his guidance, comments, and ideas during the preparation of this thesis. The major portion of this work is indebted to Dr. Hwang, Dr. L.T. Fan, and indirectly to Dr. J. Abadie of Electricité de France for allowing the author access to the GREG program. The author is also grateful to his committee members, Dr. S. Konz and Dr. Fan, for their comments on the work.

A special thank you is reserved for his wife, Kathy, for her patience in typing the manuscript.

The study was partly supported by the Kansas Water Resources Research Institute and the Office of Water Resources Research of the U.S. Department of Interior.

TABLE OF CONTENTS

CHAPTER ONE

INTRODUCTION

The constrained mathematical optimization problem has had an important role in the analysis of complex systems. Use of linear programming is common in the management decision process. Its significant success leads to the development of more sophisticated nonlinear models that attempt to describe in more detail the intricacies of the system.

In the past two decades the barrier to nonlinear programming research was the absence of computer coded algorithms. Today, modern technology has blessed (some may prefer to say burdened) us with the necessary computer hardware and software for coding and executing in reasonable time the procedures needed to attack the nonlinear programming optimization problem. With reasonably simple computer languages such as FORTRAN, ALGOL, and PL-1 the researcher, or systems analsyst has the ability to express his specialized algorithm in computer code. Ideally then, most of the time of the problem solving process can be devoted to the problem analysis and not to its computational aspects.

In the case of linear programming the application of the computer for computation is a simple process when compared to the modeling process. The systems analyst is allowed to devote practically all of his effort into the problem definition and analysis. More time is available for redefining the assumptions and measuring the resultant reaction of the system. For this reason linear models are often developed in spite of nonlinearities.

In contrast, the development of nonlinear programming has been hampered by theoretical difficulties, The problems of nonconvex sets and local optima are encountered and prevalent. Consequently, the coding of existing algorithms has been limited to special cases for which the behavior of the functions are predictable. Indeed, the algorithm superior to all algorithms is probably nonexistent.

The point is that in a study involving constrained nonlinear optimization

the modeling process is often secondary to the computational aspects of the problem. Efficient algorithms are either not available in computer code or are unknown. A solution may then be achieved by some obscure, problem oriented, possibly inefficient technique that is known only to a few experts in the field. This makes it difficult for others to verify or extend work. It is this difficulty that obscures academic progress, hinders project evaluation, and may cause relevant and important studies to be ignored. For this reason it is important to realize existing and efficient nonlinear programming codes of algorithms that are well based theoretically. When a particular known code can be applied to a problem it should be used in lieu of attempting to develop a new technique.

It is the purpose of this thesis to study in depth two existing and apparently efficient computer coded nonlinear programming algorithms. The rational is that this is another small step in the collection and presentation of information that is vital to the individual involved in constrained nonlinear optimization.

The separable programming method is a special purpose procedure in nonlinear optimization. It is an extension of the simplex method of linear programming. This is important because of the overwhelming success of coded simplex procedures. The separable programming method is, for this reason, very reliable.

One linear programming code that contains the separable programming method as an option is the Mathematical Programming System/360 (MPS/360)[37]. The usage of the code along with the theoretical development of separable programming is discussed in Chapter Two. The method requires much arithmetic in the formulation of the problem in the format acceptable by MPS/360. This necessitates the development of supplementary programs to generate the separable

programming data. For this reason a FORTRAN program is presented that aids in the usage of MPS/360. Several examples are worked including a very interesting extension. The application of separable programming to the geometric programming dual problem with N degrees of difficulty is discussed. This may be important to the nonlinear programming layman who has occasional experiences with nonlinear problems.

One important aspect of a simplex related nonlinear method is the potential for sensitivity analysis. Of course, MPS/360 contains any procedure desired by the user for such analysis and they can be applied to the separable programming option. It is this aspect coupled with the reliability of the separable programming method that warrants its presentation in Chapter Two.

The generalized reduced gradient (GRG) method is an elaborate extension of the hill-climbing gradient techniques. It is, therefore, limited to applications containing continuous, differentiable functions. The method is designed to optimize a variety of nonlinear programming problems and is, as suspected, very complicated.

The method has been coded in FORTRAN in a program named GREG[30]. The code is available but is still considered experimental in spite of its excellent performance. Because both the technique and code are new, the remaining chapters (three thru six) of this thesis are devoted to discussions on the GRG theory and on the application of the GREG code. The theory is presented in Chapter Three along with an illustrative numerical example. This is followed in Chapter Four by introducing the usage of the GREG code. Applications are discussed in Chapter Five in the form of numerical comparisons with previously worked problems from the literature. The finally is the application of the code to solve a multistage water quality management problem. In all cases the

superiority of the GRG method is apparent even with the minor shortcomings of the GREG code which are accordingly pointed out. The GRG method promises to remain an excellent nonlinear programming tool.

CHAPTER TWO

SEPARABLE PROGRAMMING

2.1 INTRODUCTION

Separable programming is a special class of nonlinear programming that is adaptable to linear programming. The problems are constructed of separable functions which have the form

$$\phi(\bar{x}) = \sum_{i=1}^{m} h_i(x_i) \tag{1}$$

The separable programming problem can be defined as finding a set of x_i, i=1,2, ..., m which maximizes (or minimizes)

$$c(\bar{x}) = \sum_{i=1}^{m} f_i(x_i) \tag{2}$$

subject to the constraints

$$\sum_{i=1}^{m} g_{ki}(x_i) \leq b_k, \qquad k=1,\ldots,p \tag{3}$$

and

$$x_i \geq 0$$

By approximating a nonlinear function of one variable by a piecewise linear function, the problem becomes a restricted linear programming problem, and can be solved by a slightly revised simplex method.

This chapter reviews the separable programming theory used in the Mathematical Programming System /360 (MPS/360) [37]. It shall be apparent that the arithmetic involved in setting up a separable programming problem for the revised simplex method becomes unbearably cumbersome. For this reason a FORTRAN program is presented that does all of the necessary calculations and produces input data for MPS/360. The program is logically simple and is exemplified.

2.2 AN APPROXIMATION OF SEPARABLE FUNCTIONS

A continuous nonlinear function of a single variable, x_i, can be approximated be a piecewise linear function over a specified interval domain. This is done by partitioning this interval domain into n_i disjoint, but contiguous, intervals. The $(n_i + 1)$ points of the partitions are represented by the set

$$S = \{ x_i^0, x_i^1, x_i^2, \ldots , x_i^{n_i}\}$$

There are two methods of representing the piecewise linear approximation of a continuous nonlinear function of one variable. The method employed in this chapter is known as the "delta method." Both methods are developed in G. Hadley's Nonlinear and Dynamic Programming [31]. The "delta method" uses the differences of adjacent points of the set, S, and the differences of the functional values at the adjacent points in developing the approximating equation of a function, $f_i(x_i)$. The differences are represented by

$$\Delta x_i^j = x_i^j - x_i^{j-1},$$

$$\Delta f_i^j = f_i(x_i^j) - f_i(x_i^{j-1}),$$

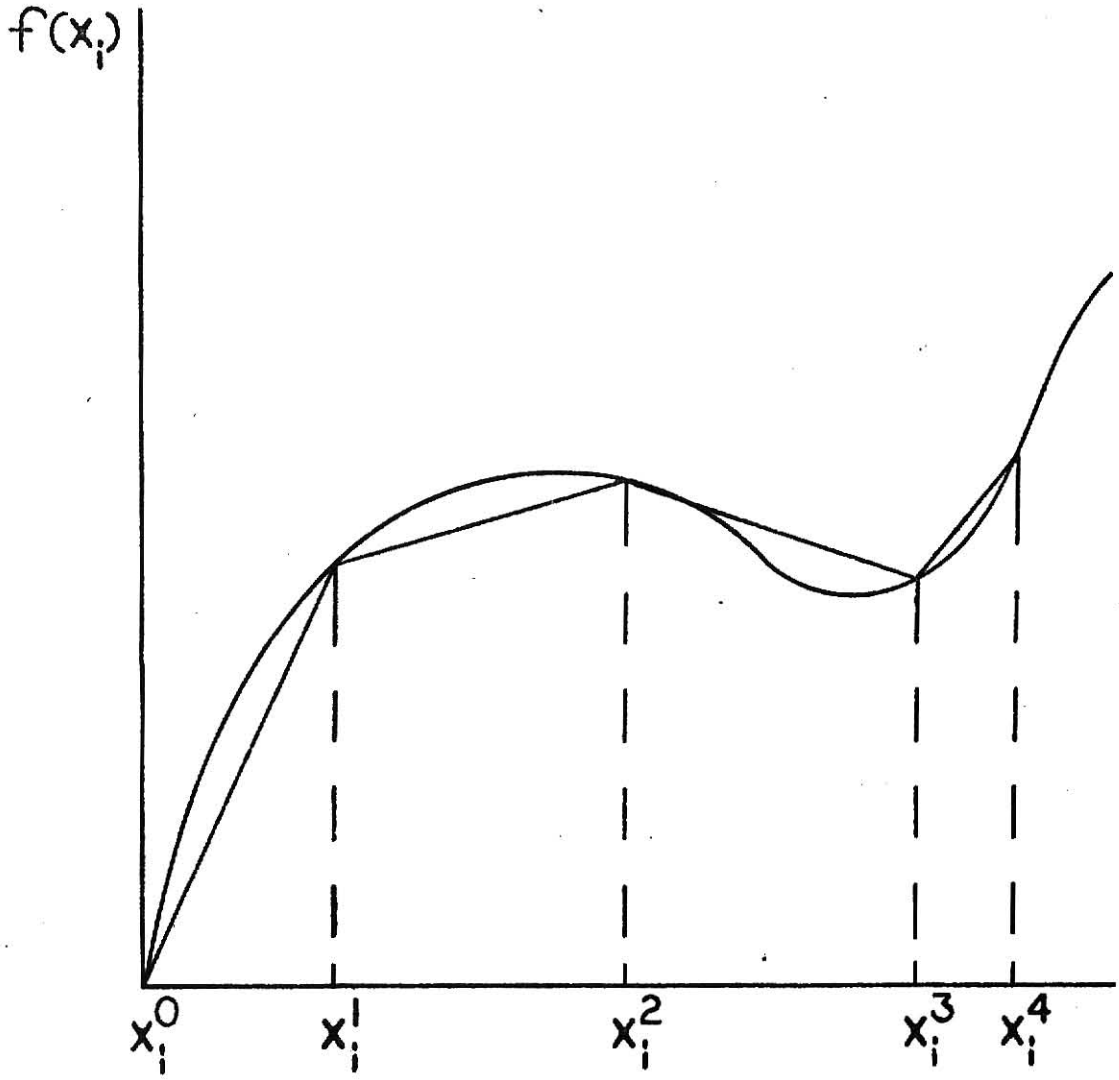
$$i = 1,2,\ldots,m \qquad j = 1,2,\ldots,n_i \qquad (4)$$

where the subscript refers to a function and/or variable such as x_i, $f_i(x_i)$, and $g_{ki}(x_i)$, and the superscript refers to a partitioning of a variable. That is, $f_i(x_i^j)$ is the value of $f_i(x_i)$ at $x_i = x_i^j$. The differences for adjacent points and the corresponding functional values for a function with $n_i = 4$ are shown in Fig. 2.1.

To represent the variable x_i and the approximation of $f_i(x_i)$, a set of variables, D_i^j , j=1, 2, ..., n_i , is created that follows what is known as the "restricted-basis-entry rule." The rule is satisfied for any one of the following conditions.

THIS BOOK CONTAINS NUMEROUS PAGES WITH DIAGRAMS THAT ARE CROOKED COMPARED TO THE REST OF THE INFORMATION ON THE PAGE.

THIS IS AS RECEIVED FROM CUSTOMER.



$$\Delta x_i^1 = x_i^1 - x_i^0 \qquad \Delta f_i^1 = f_i(x_i^1) - f_i(x_i^0)$$

$$\Delta x_i^2 = x_i^2 - x_i^1 \qquad \Delta f_i^2 = f_i(x_i^2) - f_i(x_i^1)$$

$$\Delta x_i^3 = x_i^3 - x_i^2 \qquad \Delta f_i^3 = f_i(x_i^3) - f_i(x_i^2)$$

$$\Delta x_i^4 = x_i^4 - x_i^3 \qquad \Delta f_i^4 = f_i(x_i^4) - f_i(x_i^3)$$

Fig. 2.1. Linear approximation of $f_i(x_i)$.

(i) $0 \leq D_i^1 \leq 1$ iff $D_i^j = 0,\ j = 2, 3, \ldots, n_i$

(ii) $0 \leq D_i^j \leq 1$ iff $D_i^\ell = 1,\ \ell = 1, 2, \ldots, j-1$

and

$D_i^k = 0,\ k = j+1, \ldots, n_i$

(iii) $0 \leq D_i^{n_i} \leq 1$ iff $D_i^j = 1,\ j = 1, 2, \ldots, n_i - 1$

where n_i is the number of partitioning intervals for a variable x_i. D_i^j represents a special variable created for the partition j of variable x_i. Intuitively, for any $0 \leq D_i^j \leq 1$ all previous D_i^ℓ variables ($\ell = 1, \ldots, j-1$) must have a value of one and all following values ($\ell = j+1, \ldots, n_i$) must be zero.

Example 1

Let $n_i = 4$. Cases a, b, c, and d satisfy the "restricted-basis-entry rule."

	D_i^1	D_i^2	D_i^3	D_i^4
a	1/2	0	0	0
b	1	1/4	0	0
c	1	1	7/8	0
d	1	1	1	2

Note that in case d variable D_i^4 does not have to be less than or equal to one because it is the last variable in the set.

Using the "restricted-basis-entry rule" the variable x_i in Fig. 2.1 is represented by

$$x_i = \Delta x_i^1 D_i^1 + \Delta x_i^2 D_i^2 + \Delta x_i^3 D_i^3 + \Delta x_i^4 D_i^4 \qquad (5)$$

and the function $f_i(x_i)$ is approximated by

$$f_i \cong \Delta f_i^1 D_i^1 + \Delta f_i^2 D_i^2 + \Delta f_i^3 D_i^3 + \Delta f_i^4 D_i^4 \qquad (6)$$

It is apparent that x_i is represented exactly and that $f_i(x_i)$ is approximated by what is essentially linear interpolation.

Example 2

Approximate $f_i(x_i) = (x_i)^3$ using three partitions over (0,2). Let $S = \{0, 1/2, 1, 2\}$.

$$x_i^0 = 0 \qquad f_i(x_i^0) = 0$$
$$x_i^1 = 1/2 \qquad f_i(x_i^1) = 1/8$$
$$x_i^2 = 1 \qquad f_i(x_i^2) = 1$$
$$x_i^3 = 2 \qquad f_i(x_i^3) = 8$$

The differences can now be calculated.

j	1	2	3
Δx_i^j	1/2	1/2	1
Δf_i^j	1/8	7/8	7

Using the differences, the exact equation for x_i and the approximating equation for $f_i(x_i)$ can be written as

$$x_i = 1/2\ D_i^1 + 1/2\ D_i^2 + 1 \cdot D_i^3$$

and

$$f_i \cong 1/8\ D_i^1 + 7/8\ D_i^2 + 7 \cdot D_i^3$$

Some sample calculations comparing the approximate and the exact representations of $f_i(x_i)$ are given in Table 2.1. Notice that when x_i is outside the interval domain ($x_i = 4$) the approximation becomes very poor.

The development so far has implicitly assumed $x_i^0 = 0$ and $f_i(x_i^0) = 0$. This is not generally true. The general expressions for x_i and $f_i(x_i)$ given that $x_i^0 \neq 0$ and $f_i(x_i^0) \neq 0$ with n_i partitions are

$$x_i = x_i^0 + \sum_{j=1}^{n_i} \Delta x_i^j \, D_i^j \tag{7}$$

and

$$f_i \cong f_i\,(x_i^0) + \sum_{j=1}^{n_i} \Delta f_i^j \, D_i^j \tag{8}$$

This is extremely important when working with logarithmic and exponential functions.

Example 3

Approximate $f_i(x_i) = e^{x_i}$ with two uniform partitions over (0,2) (see Fig. 2.2). Notice that when $x_i^0 = 0$ the value of $e^{x_i} = 1.0$. The piecewise linear approximation function must have the general form of equation (8). The function values at the partition boundaries are

$x_i^0 = 0$	$f_i(x_i^0) = 1$
$x_i^1 = 1$	$f_i(x_i^1) = 2.7183$
$x_i^2 = 2$	$f_i(x_i^2) = 7.3891$

The differences are now calculated.

j	1	2
Δx_i^j	1.0	1.0
Δf_i^j	1.7183	4.6708

(given x_i)	D_i^1	D_i^2	D_i^3	f_i(approximate)	f_i(exact)
1	1	1	0	1	1
$\frac{3}{4}$	1	$\frac{1}{2}$	0	$\frac{9}{16} \tilde{=} .563$	$\frac{27}{64} \tilde{=} .422$
$\frac{3}{2}$	1	1	$\frac{1}{2}$	$4\frac{1}{2}$	$3\frac{3}{8}$
4	1	1	3	22	64

Table 2.1. Sample calculations comparing some exact and approximate values.

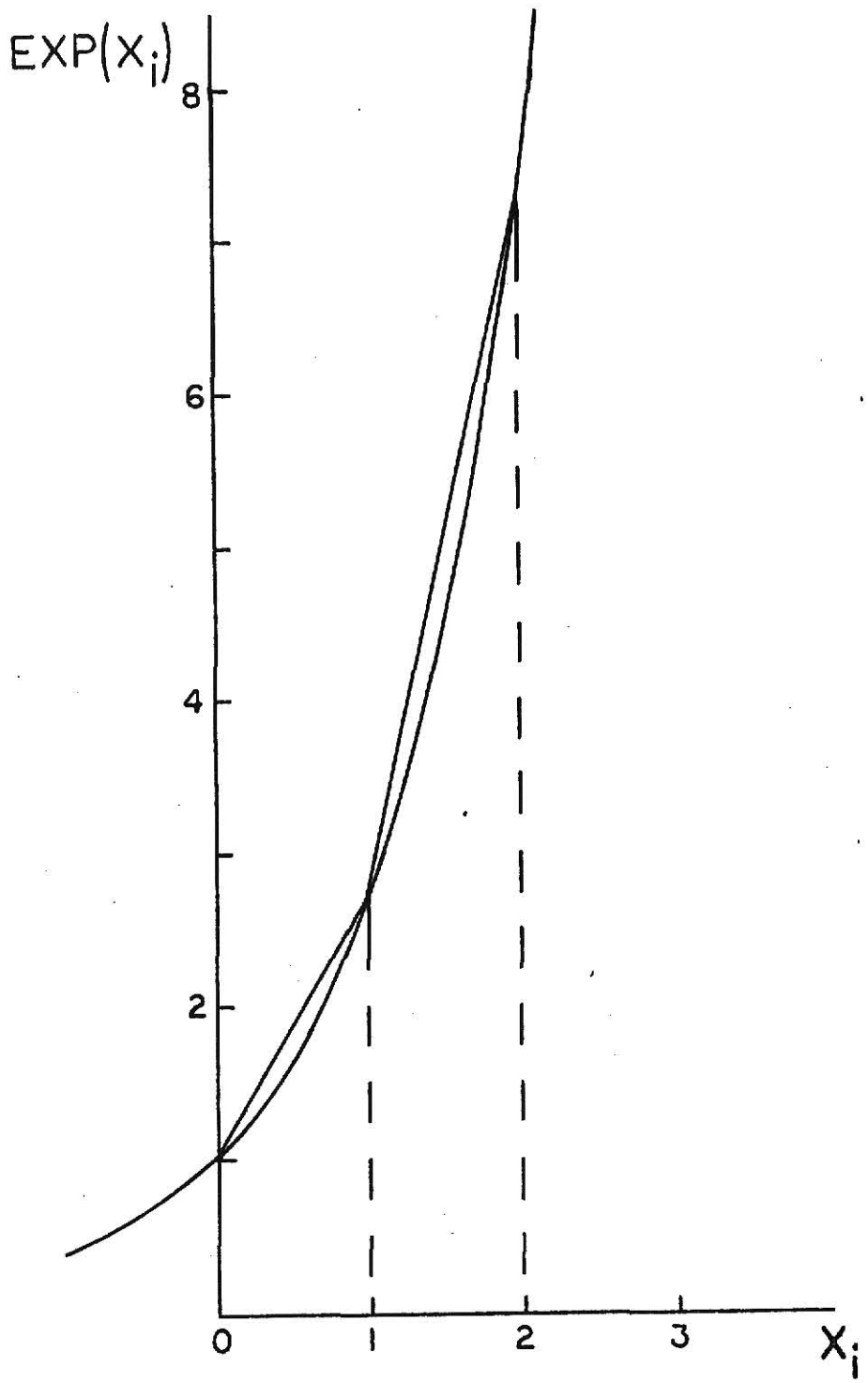


Fig. 2.2. Exp(x_i) with two uniform partitions over (0,2).

The piecewise linear functions are

$$x_i = 1.0\ D_i^1 + 1.0\ D_i^2$$

and

$$e^{x_i} \cong 1.0 + 1.7183\ D_i^1 + 4.6708\ D_i^2$$

Now consider a constraint of the form

$$e^{x_i} \leq b_k$$

The above approximation is applied in the following manner.

$$1.0 + 1.7183\ D_i^1 + 4.6708\ D_i^2 \leq b_k$$

or

$$1.7183\ D_i^1 + 4.6708\ D_i^2 \leq b_k - 1.0$$

The constant term is moved to the right hand side of the inequality without altering its net effect. The constraint is now a linear function subject to the "restricted-basis-entry rule."

2.3 THE SEPARABLE PROGRAMMING PROBLEM

The separable programming problem as defined before is

maximize (or minimize)

$$c(\bar{x}) = \sum_{i=1}^{m} f_i(x_i) \qquad (2)$$

subject to the constraints

$$\sum_{i=1}^{m} g_{ki}\ (x_i) \leq b_k, \qquad k = 1, 2, \ldots, p \qquad (3)$$

$$x_i \geq 0$$

The number of variables equals m and the number of constraints, p.

If n_i is the number of partitions for the variable x_i then the variable x_i and the approximations for the functions $f_i(x_i)$ and $g_{ki}\ (x_i)$ can be written as

$$x_i = x_i^0 + \sum_{j=1}^{n_i} \Delta x_i^j \, D_i^j \tag{9}$$

$$f_i(x_i) \cong f_i(x_i^0) + \sum_{j=1}^{n_i} \Delta f_i^j \, D_i^j \tag{10}$$

$$g_{ki}(x_i) \cong g_{ki}(x_i^0) + \sum_{j=1}^{n_i} \Delta g_{ki}^j \, D_i^j \tag{11}$$

$k = 1, 2, \ldots, p,$ and $i = 1, 2, \ldots, m$

The approximate form of the problem becomes maximize (or minimize)

$$c = \sum_{i=1}^{m} \sum_{j=1}^{n_i} \Delta f_i^j \, D_i^j + \sum_{i=1}^{m} f_i(x_i^0) \tag{12}$$

subject to the constraints

$$\sum_{i=1}^{m} \sum_{j=1}^{n_i} \Delta g_{ki}^j \, D_i^j \leq b_k - \sum_{i=1}^{m} g_{ki}(x_i^0), \qquad k = 1, 2, \ldots, p \tag{13}$$

$$x_i - \sum_{j=1}^{n_i} \Delta x_i^j \, D_i^j = x_i^0, \qquad i = 1, 2, \ldots, m \tag{14}$$

$$0 \leq D_i^j \leq 1, \qquad j = 1, 2, \ldots, n_i \tag{15}$$

$$x_i \geq 0 \tag{16}$$

If $f_i(x_i^0) \neq 0$ for any i value, equation (12) has a constant term, $\sum_{i=1}^{m} f_i(x_i^0)$, which may be dropped without effect on the optimum solution. If any $g_{ki}(x_i^0) \neq 0$ then the bound, b_k, must be adjusted. This is important because the problem changes if they are dropped. And finally, if $x_i^0 \neq 0$ the grid equations, equation (14), will not be equal to zero. The number of original constraints equals p, the number of grid equations equals m, and the number of constraints due to the piecewise linear approximation funtions, $D_i^j \leq 1$, equals $\sum_{i=1}^{m} n_i$. The problem now has $p + m + \sum_{i=1}^{m} n_i$ constraints.

The separable programming problem represented by equations (12) through (16) is now a linear programming problem that is subject to the "restricted-basis-entry rule." Any basis transformation cannot violate the "restricted-basis-entry rule", hence, the origin of this title. Put simply, a variable D_i^j cannot enter the basis unless all of the previous D_i^j values are equal to one and all of the following D_i^j values are zero. This rule insures that the approximated functions are represented correctly.

Example 4.

A simple separable programming problem shall be solved. The problem is maximize

$$c(\bar{x}) = (2x_1 - (x_1)^2) + (x_2 - (x_2)^2)$$

subject to the constraint

$$(x_1)^2 + 2(x_2)^2 \leq 1$$

$$x_1 \geq 0, \qquad x_2 \geq 0$$

The functions of this problem are separable.

$$f_1(x_1) = 2x_1 - (x_1)^2, \qquad f_2(x_2) = x_2 - (x_2)^2$$

$$g_{11}(x_1) = (x_1)^2, \qquad g_{12}(x_2) = 2(x_2)^2$$

The variable domains are chosen as (0, 1) and (0, 1/2) for x_1 and x_2, respectively. The number of partitions for x_1 is $n_1 = 4$ and for x_2 is $n_2 = 2$. The partitions are uniform for simplicity. The function values at the interval boundaries and the calculated differences are given in Table 2.2. The approximating equations are determined as

Function values

j	x_1^j	f_1^j	g_{11}^j	x_2^j	f_2^i	g_{12}^j
0	0	0	0	0	0	0
1	$\frac{1}{4}$	$\frac{7}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{3}{16}$	$\frac{1}{8}$
2	$\frac{1}{2}$	$\frac{3}{4}$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{2}$
3	$\frac{3}{4}$	$\frac{15}{16}$	$\frac{9}{16}$			
4	1	1	1			

Function differences

j	Δx_1^j	Δf_1^j	Δg_{11}^j	Δx_2^j	Δf_2^j	Δg_{12}^j
1	$\frac{1}{4}$	$\frac{7}{16}$	$\frac{1}{16}$	$\frac{1}{4}$	$\frac{3}{16}$	$\frac{1}{8}$
2	$\frac{1}{4}$	$\frac{5}{16}$	$\frac{3}{16}$	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{3}{8}$
3	$\frac{1}{4}$	$\frac{3}{16}$	$\frac{5}{16}$			
4	$\frac{1}{4}$	$\frac{1}{16}$	$\frac{7}{16}$			

Table 2.2. The function values and differences for example 4.

$$f_1(x_1) \cong \frac{7}{16} D_1^1 + \frac{5}{16} D_1^2 + \frac{3}{16} D_1^3 + \frac{1}{16} D_1^4$$

$$f_2(x_2) \cong \frac{3}{16} D_2^1 + \frac{1}{16} D_2^2$$

$$g_{11}(x_1) \cong \frac{1}{16} D_1^1 + \frac{3}{16} D_1^2 + \frac{5}{16} D_1^3 + \frac{7}{16} D_1^4$$

$$g_{12}(x_2) \cong \frac{1}{8} D_2^1 + \frac{3}{8} D_2^2$$

The grid equations are

$$x_1 = \frac{1}{4} D_1^1 + \frac{1}{4} D_1^2 + \frac{1}{4} D_1^3 + \frac{1}{4} D_1^4$$

$$x_2 = \frac{1}{4} D_2^1 + \frac{1}{4} D_2^2$$

It must be remembered that all $D_i^j \leq 1$. This series of constraints must be incorporated in the final L.P. problem. The final problem set-up is maximize

$$c = \frac{7}{16} D_1^1 + \frac{5}{16} D_1^2 + \frac{3}{16} D_1^3 + \frac{1}{16} D_1^4 + \frac{3}{16} D_2^1 + \frac{1}{16} D_2^2$$

subject to the constraints

$$\frac{1}{16} D_1^1 + \frac{3}{16} D_1^2 + \frac{5}{16} D_1^3 + \frac{7}{16} D_1^4 + \frac{1}{8} D_2^1 + \frac{3}{8} D_2^2 \leq 1$$

$$x_1 - \frac{1}{4} D_1^1 - \frac{1}{4} D_1^2 - \frac{1}{4} D_1^3 - \frac{1}{4} D_1^4 = 0$$

$$x_2 - \frac{1}{4} D_2^1 - \frac{1}{4} D_2^2 = 0$$

$$0 \leq D_i^j \leq 1, \qquad i = 1, 2, \qquad j = 1, \ldots, n_i$$

$$x_i \geq 0$$

This problem is easily solved by incorporating the "restricted-basis- entry rule" with any standard simplex tableau. The problem may be solved in five iterations. The starting tableau is given for reference in Table 2.3. In the starting tableau notice that slack variable S_1 is for the original constraint. Slack variables S_i, $i = 2, \ldots, 7$, are for the separable variable conditions, $D_i^j \leq 1$. Also notice that it is imperative that the first iteration involves D_1^1 or D_2^1. If the first iteration involves D_1^1 then the second iteration can involve D_2^1, or D_1^2 if $D_1^1 = 1$. The iteration process is given in Table 2.4. The final optimum solution is

$$x_1 = \frac{3}{4} = .75000$$

$$x_2 = \frac{11}{24} = .45833$$

$$c(\bar{x}) = \frac{113}{96} = 1.17708$$

It is apparent that the total number of arithmetic calculations could become astronomical for multidimensional problems involving many constraints. An upper bound of the number of differences calculated is

$$\sum_{i=1}^{m} n_i p + \sum_{i=1}^{m} n_i$$

where

m = number of variables used in separable functions

n_i = number of partitions for variable i

p = number of nonlinear constraints plus 1 (for a nonlinear objective function)

c_i	basis	b_k	0 x_1	0 x_2	$\frac{7}{16}$ D_1^1	$\frac{5}{16}$ D_1^2	$\frac{3}{16}$ D_1^3	$\frac{1}{16}$ D_1^4	$\frac{3}{16}$ D_2^1	$\frac{1}{16}$ D_2^2	0 S_1	0 S_2	0 S_3	0 S_4	0 S_5	0 S_6	0 S_7
0	S_1	1	0	0	$\frac{1}{16}$	$\frac{3}{16}$	$\frac{5}{16}$	$\frac{7}{16}$	$\frac{1}{8}$	$\frac{3}{8}$	1						
0	x_1	0	1	0	$-\frac{1}{4}$	$-\frac{1}{4}$	$-\frac{1}{4}$	$-\frac{1}{4}$	0	0	0						
0	x_2	0	0	1	0	0	0	0	$-\frac{1}{4}$	$-\frac{1}{4}$	0						
0	S_2	1			1							1					
0	S_3	1				1							1				
0	S_4	1					1							1			
0	S_5	1						1							1		
0	S_6	1							1							1	
0	S_7	1								1							1
$c = 0$			0	0	$-\frac{7}{16}$	$-\frac{5}{16}$	$-\frac{3}{16}$	$-\frac{1}{16}$	$-\frac{3}{16}$	$-\frac{1}{16}$	0	0	0	0	0	0	0

Table 2.3. Starting simplex tableau. (All blank spaces are zero.)

Iteration Number	Vector into Basis	Vector out of Basis
1	D_1^1	S_2
2	D_1^2	S_3
3	D_2^1	S_6
4	D_1^3	S_4
5	D_2^2	S_1

Table 2.4. Iteration process for the revised simplex algorithm. Notice that no basis transformation violates the "restricted-basis-entry rule."

For the reliability problem [45] worked later involving five variables, three constraints, a nonlinear objective function, and $n_i = 11$ for all i, the total number of differences calculated for the given solution were 275. The problem of arithemic is what justifies a FORTRAN routine to aid in separable programming efforts.

2.4 A REVIEW OF IBM MPS/360 VERSION 2, LINEAR AND SEPARABLE PROGRAMMING

MPS/360 is a set of procedures designed to solve problems involving linear and separable programming. The user controls the procedures by means of a control language program. This section is to briefly review the necessities with emphasis on problem formulation and data input.

Figure 2.3 shows an MPS/360 control language program which formulates the problem. The statements shall be briefly discussed. The "PROGRAM" and "INITIALZ" statements initiate and initialize the system. They are required control statements. The "MOVE (XDATA, 'name')" statement defines the name of the input data. Closely related is the "MOVE (XPBNAME, 'name')" statement. "CONVERT" is the input procedure. It must be supplied with the name of the data and the problem name by the two previous "MOVE" statements . The optional procedure "BCDOUT" is a data echo check procedure. The next two "MOVE" statements define the objective function (XOBJ) and the right hand side (XRHS). They are required for linear and separable programming problems. "TITLE" is an optional statement supplying page headings. The next procedure is "SETUP." This procedure transforms the problem into the form that can be optimized by an optimization procedure. In this case the optimization procedure is "PRIMAL." In "SETUP" a "BOUND" vector is defined as "SEPBOUND." The problem is maximization since the parameter "MAX" is specified (default is minimization). "SOLUTION" is the final output from "PRIMAL" and the program is ended with

```
CONTROL PROGRAM COMPILER - MPS/360 V2-M8

C001                    PROGRAM
C002                    INITIALZ
C064                    MOVE(XDATA,'DATA-SET')
C065                    MOVE(XPBNAME,'EXAMPLE')
C066                    CONVERT
C067                    BCDOUT
C068                    MOVE(XOBJ,'OBJECT')
C069                    MOVE(XRHS,'LIMITS')
C070                    TITLE('EXAMPLE')
C071                    SETUP('BOUND','SEPBOUND','MAX')
C072                    PRIMAL
C073                    SOLUTION
C074                    EXIT
C075                    PEND
```

Fig. 2.3. Listing of an MPS/360 Control Language Program.

"EXIT" and "PEND." In review, the control language program shown in Fig. 2.3 defines a problem named EXAMPLE with input data named DATA-SET. The objective function is OBJECT and the right hand side is LIMITS. This problem, titled EXAMPLE, will have some of its variables bounded by a vector called SEPBOUND and then it will be maximized.

The input data are delivered, for the purposes of this thesis, to MPS/360 in five sections. They are NAME, ROWS, COLUMNS, RHS, and BOUNDS. MPS/360 distinguishes each section by its name. They must be supplied in the given order and each section is defined by the above names, left-justified, starting in card column one. Each section follows a general card format.

Figure 2.4 shows the format for the general MPS/360 input data card. It is divided into six fields with each field being designated for a special purpose. Field 1 is the code field containing information relating to the data on the card. If the data card happens to define a row then the code will indicate N, L, G, or E for "objective function", "less than or equal to", "greater than or equal to", or "equal to", respectively. Fields 2, 3, and 5 are name fields. They can contain an alphabetic name that is 1-8 characters in length and is left-justified. Fields 4 and 6 are numeric fields with a FORTRAN format equivalent of F12.X.

The next two examples demonstrate the input data format. The 'MARKER' cards necessary for distinguishing the special separable variables are described in the second example.

EXAMPLE 5

This example describes how the data is formulated for the simple linear problem of maximize $y = 3X_1 + X_2$

	Field 1	Field 2	Field 3	Field 4	Field 5	Field 6
Columns	2-3	5-12	15-22	25-36	40-47	50-61
Type of Contents	code	name	name	value	name	value

Fig. 2.4. General card format for MPS/360 Linear and Separable Programming [37].

subject to the constraints

$$X_1 + X_2 \leq 2$$

$$0 \leq X_1 \text{ and } X_2 \leq 3/2$$

The input data are shown in Fig. 2.5 and consists of five parts: NAME, ROWS, COLUMNS, RHS, and BOUNDS. The name of the data is defined in the NAME, section. The ROWS section defines the rows of the L.P. problem to MPS/360. A row is coded as N, L, G, or E (objective, $\leq$, $\geq$, or =) and given a name. In this case OBJECT is the objective function and ROW1 is the constraint. The COLUMNS section defines the elements of the L.P. matrix by variable (column). The RHS section defines the right hand side of the constraints. In this case the RHS vector is named LIMITS. The last section is BOUNDS. The variables are given upper bounds by UP and the BOUNDS vector is named BOUND. The input data is concluded by ENDATA.

Example 6.

Example 4 is worked using MPS/360. The control language program is shown in Fig. 2.3. Figure 2.6 shows the MPS/360 data echo check. Separable data are denoted by 'MARKER' cards. A 'MARKER', 'SEPORG' card occurs prior to each set of separable variables as shown. Each set is given a name. They are SET1 and SET2 in this example. At the end of all of the separable data occurs a 'MARKER', 'SEPEND' card. This card tells MPS/360 that no more separable variables follow. The 'MARKER' card name is contained in field 2, the 'MARKER' symbol is contained in field 3 and the 'SEPORG' or 'SEPEND' symbol is contained in field 5. All linear data occur before or after the separable data. The 'MARKER' cards are the only differences when using MPS/360 for separable programming. The final solution is shown in Fig. 2.7.

	Columns											
	1	1	2	2	3	3	4	4	5	5	6	
12345	0	5	0	5	0	5	0	5	0	5	0	
NAME		DATA-SET										
N	OBJECT											
L	ROW1											
COLUMNS												
	X1		OBJECT		3.0		ROW1			1.0		
	X2		OBJECT		1.0		ROW1			1.0		
RHS												
	LIMITS		ROW1		2.0							
BOUNDS												
UP	BOUND		X1		1.5							
UP	BOUND		X1		1.5							
ENDATA												

Fig. 2.5. Sample MPS/360 input data. Data are assumed zero unless specified otherwise.

```
                EXECUTOR.     MPS/360 V2-M8

NAME          DATA-SET
ROWS
 N  OBJECT
 L  ROW1
 E  GRID1
 E  GRID2
COLUMNS
    X1        GRID1          1.00000
    X2        GRID2          1.00000
    SET1      'MARKER'                  'SEPORG'
    D11       OBJECT          .43750    ROW1          .06250
    D11       GRID1      -    .25000
    D12       OBJECT          .31250    ROW1          .18750
    D12       GRID1      -    .25000
    D13       OBJECT          .18750    ROW1          .31250
    D13       GRID1      -    .25000
    D14       OBJECT          .06250    ROW1          .43750
    D14       GRID1      -    .25000
    SET2      'MARKER'                  'SEPORG'
    D21       OBJECT          .18750    ROW1          .12500
    D21       GRID2      -    .25000
    D22       OBJECT          .06250    ROW1          .37500
    D22       GRID2      -    .25000
    ENDSET    'MARKER'                  'SEPEND'
RHS
    LIMITS    ROW1           1.00000
BOUNDS
 UP SEPBOUND  D11            1.00000
 UP SEPBOUND  D12            1.00000
 UP SEPBOUND  D13            1.00000
 UP SEPBOUND  D14            1.00000
 UP SEPBOUND  D21            1.00000
 UP SEPBOUND  D22            1.00000
ENDATA
```

Fig. 2.6. Input echo check from "BCDOUT".

Fig. 2.7. MPS/360 solution for Example 4.

EXAMPLE

SOLUTION (OPTIMAL)

TIME = 0.30 MINS. ITERATION NUMBER = 5

...NAME...	...ACTIVITY...	DEFINED AS
FUNCTIONAL	1.17708	OBJECT
RESTRAINTS		LIMITS
BOUNDS....		SEPBOUND

EXAMPLE

SECTION 1 - ROWS

	NUMBER	...ROW..	AT	...ACTIVITY...	SLACK ACTIVITY	..LOWER LIMIT.	..UPPER LIMIT.	.DUAL ACTIVITY
	1	OBJECT	BS	1.17708	1.17708-	NONE	NONE	1.00000
	2	ROW1	LL	1.00000	.	NONE	1.00000	.16667-
A	3	GRID1	EQ	.	.	.	.	.
A	4	GRID2	EQ	.	.	.	.	.

EXAMPLE

SECTION 2 - COLUMNS

	NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
	5	X1	BS	.75000	.	.	NONE	.
	6	X2	BS	.45833	.	.	NONE	.
	7	D11	LL	1.00000	.43750	.	1.00000	.42708
	8	D12	LL	1.00000	.31250	.	1.00000	.28125
	9	D13	LL	1.00000	.18750	.	1.00000	.13542
	10	D14	LL	.	.06250	.	1.00000	.01042-
	11	D21	LL	1.00000	.18750	.	1.00000	.16667
	12	D22	BS	.83333	.06250	.	1.00000	.

2.5 A FORTRAN SUBROUTINE FOR GENERATING SEPARABLE PROGRAMMING DATA TO BE USED WITH MPS/360

One difficulty in using the separable programming subroutine of MPS/360 is calculating the values of Δx_i^j, Δf_i^j, and Δg_{ki}^j of equations (9), (10), and (11) and punching these values in the input data format. This section presents a FORTRAN program which performs the necessary calculations and punches the data. The general flow chart is shown in Fig. 2.8.

There are three main loops within the flow chart logic. The first loop controls the variable, x_i, i = 1, ..., m, and the computational process on it. Each variable is admitted to the computational process and is partitioned according to four input parameters. The first is x_i^0, the starting point. The next three parameters are used to partition a predetermined feasible region uniformly. A partition is automatically created between x_i^0 and a specified lower bound, unless the distance between them is zero. From the lower bound to an upper bound the user specifies the number of uniform partitions. So each variable requires x_i^0, a lower bound, an upper bound, and the number of uniform partitions. The second loop controls the partitioning process of the variable, x_i, and is contained within the first loop. It is at this point that a special separable variable is created, one for each partition. After the partition is created the program punches the data according to the MPS/360 format. The third loop calculates the function differences, Δf_i^j or Δg_{ki}^j, which are the coefficients for the special separable variables, D_i^j. Within this loop a subroutine, named FUNC, defines the separable functions of the problem. For the main program listing, refer to Appendix 1.

The general flow chart for FUNC is shown in Fig. 2.9. The subroutine

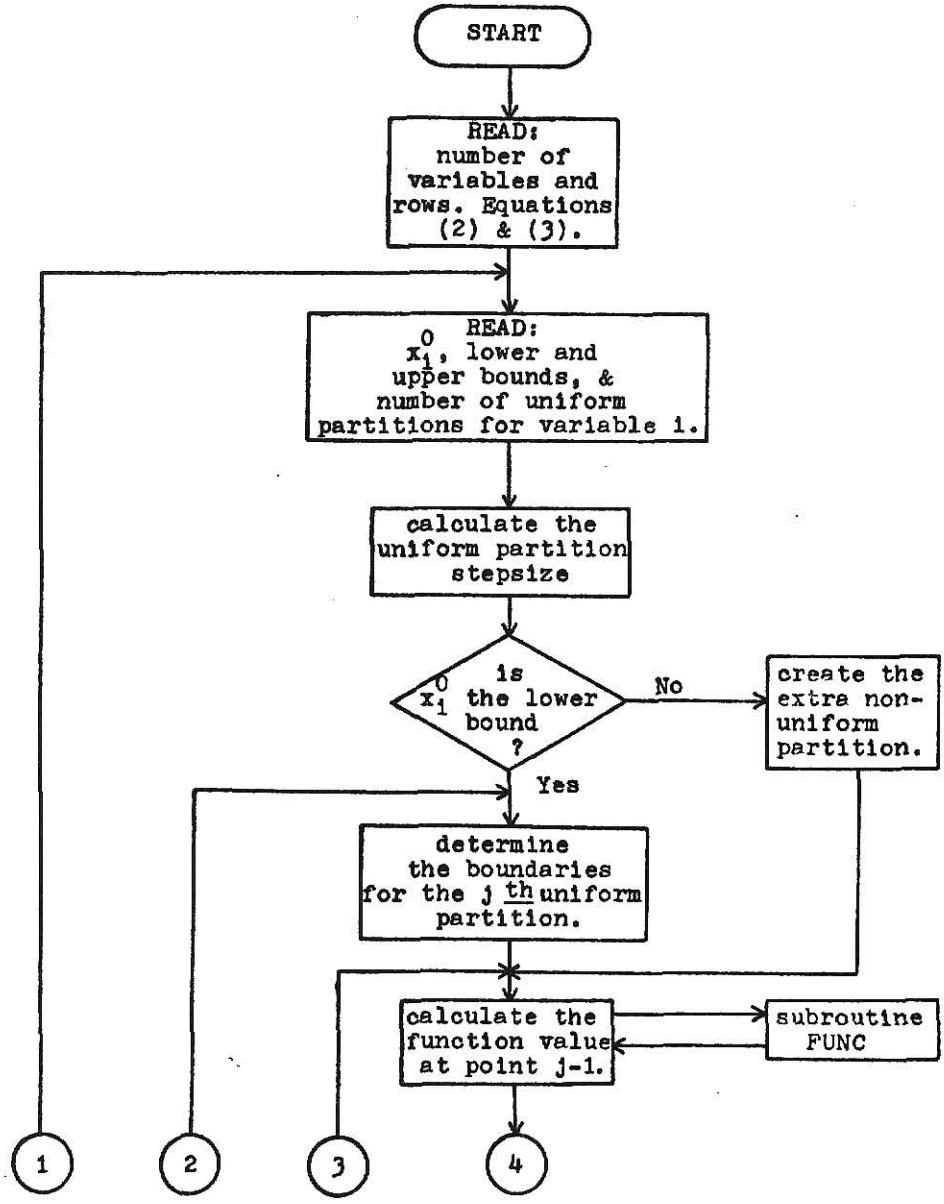


Fig. 2.8. The flow chart of the program that generates the separable programming data.

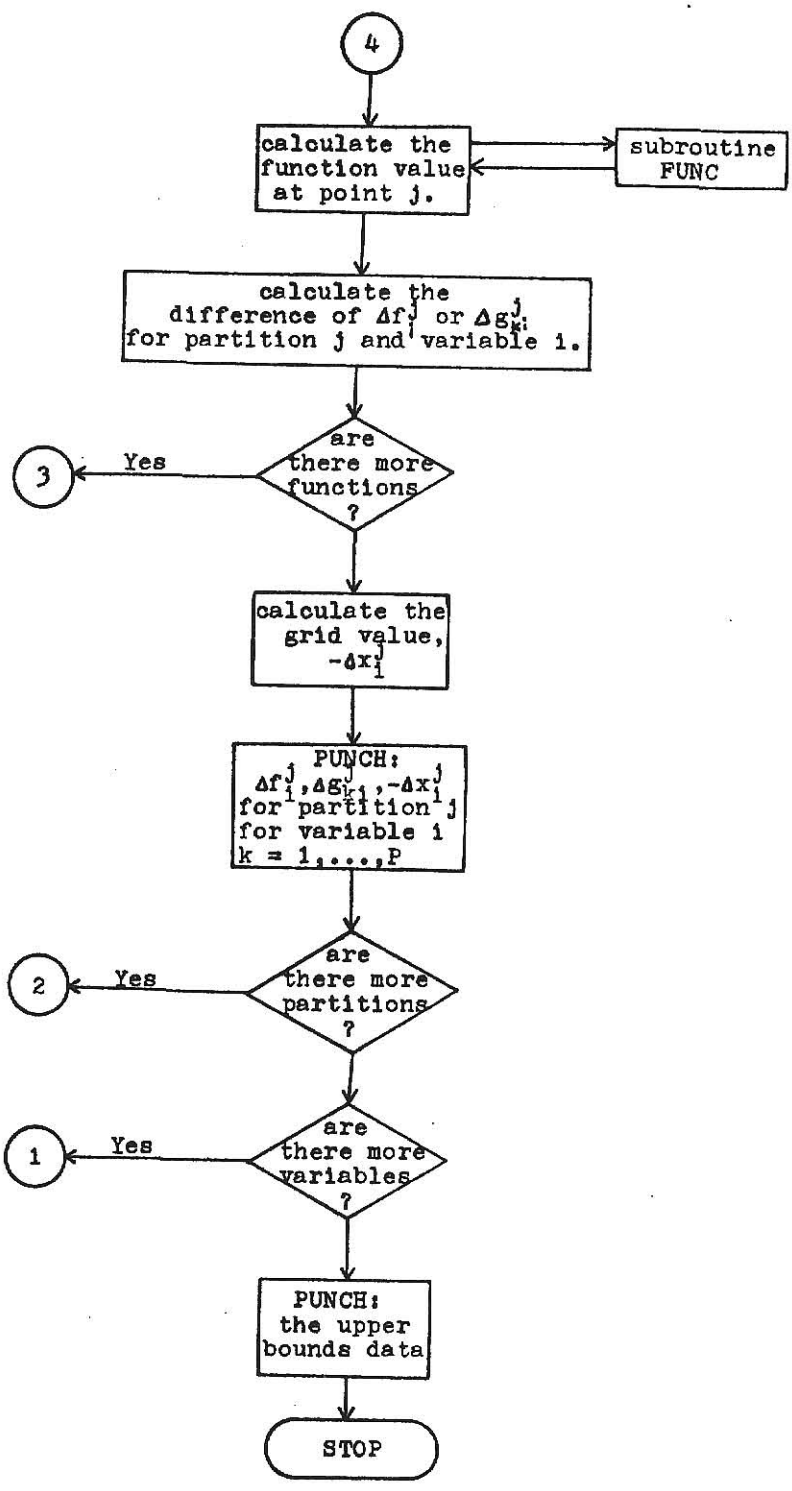


Fig. 2.8. (continued)

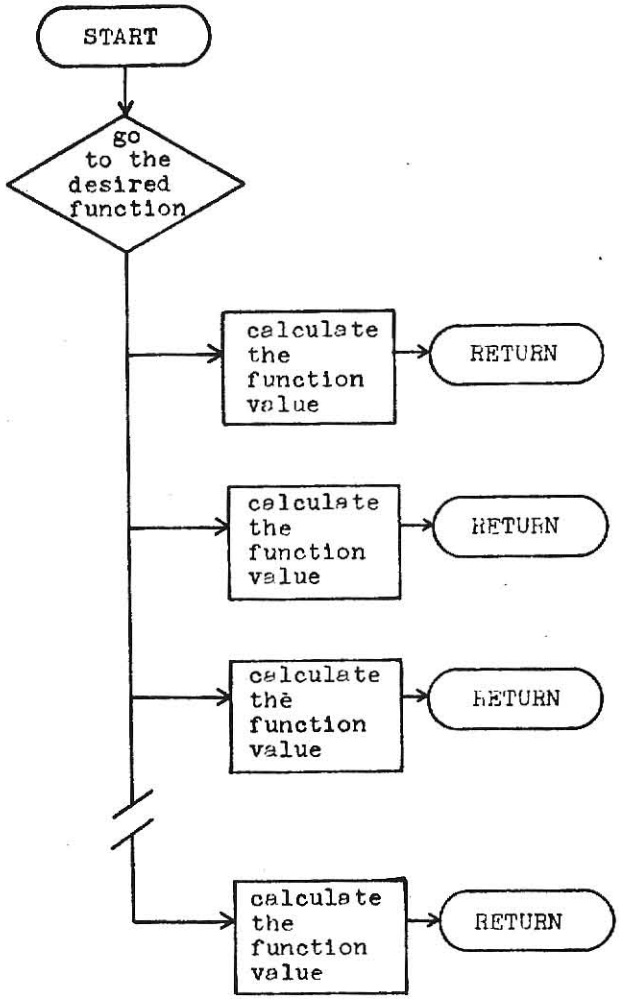


Fig. 2.9. Flow chart for the subroutine FUNC.

contains the separable functions in the form of a matrix, F(I, J). The first row of the matrix, I = 1, is for the objective function (or possible the first constraint if the objective function is linear). The remaining rows are for the constraint functions. The matrix colums represent funtions for the variable, $x_1, \ldots, x_m$. Consider Example 4. Figure 2.10 shows the separable function breakdown with row 1 as the objective function and row 2 as the constraint. When FUNC is called by the main program it is supplied with the desired row and column number. FUNC returns the computed value for that function. This may be done in several ways. A practical procedure is to use a "locator function" and a computed GO TO statement to locate the correct function. This is demonstrated in Fig. 2.11. The arithmetic statement (J - 1) *NROWS + I, where I is the row, J is the column, and NROWS is the total number of rows in the matrix, will locate F(I,J) when the matrix is ordered linearly by columns. The main program supplies the desired row and column numbers and the value of the dummy variable, x. FUNC returns the desired function value. Figure 2.12 exemplifies the construction of FUNC for Example 4. Note that it is not necessary to treat F(I,J) as a matrix, but it aids the programmer.

The next item is the input data. The main program must be supplied two parameter values for initialization purposes and four values for each problem variable. The first two are the dimensions of the separable function matrix, F(I, J). They are NFUN, the number of variables, and NROWS, the number of matrix rows. They are read only once on the first data card. A set of the next four values is needed for each variable. They are START, INTER, ALOW, and UPPER. START is the initial starting point, X_i^0. Its value is usually zero unless the user is working with a function not defined at

	column 1	column 2
row 1	$F(1, 1) = 2x_1 - (x_1)^2$	$F(1, 2) = x_2 - (x_2)^2$
row 2	$F(2, 1) = (x_1)^2$	$F(2, 2) = 2(x_2)^2$

Fig. 2.10. The matrix of separable functions, F(I, J), for example 4.

```
      SUBROUTINE FUNC(I,J,X,F,NFUN,NROWS)
      DIMENSION F(50,50)
      MM=(J-1)*NROWS+I
      GO TO(1,2,...),MM
    1 F(1,1)=
      RETURN
    2 F(2,1)=
      RETURN
    3 F(3,1)=
      RETURN
         .
         .
         .
      END
```

Fig. 2.11. A proper construction of FUNC.

```
      SUBROUTINE FUNC(I,J,X,F,NFUN,NROWS)
      DIMENSION F(50,50)
      GO TO(1,2,3,4),MM
    1 F(1,1)=2.0*X-X**2
      RETURN
    2 F(2,1)=X**2
      RETURN
    3 F(1,2)=X-X**2
      RETURN
    4 F(2,2)=2.0*X**2
      RETURN
      END
```

Fig. 2.12. FUNC for Example 4.

zero. The other three values are concerned with partitioning only the region where the optimum solution is expected to be located. Between ALOW and UPPER, the lower and upper bounds, there will occur INTER uniform partitions. The total number of partitions will be either INTER or INTER + 1. If ALOW is not equal to START an extra partition is created between them. The formats for the input variables are given in Table 2.5 and in the comment section of the main program. It is not necessary but recommended that START $\leq$ ALOW < UPPER. Figure 2.13 shows some sample input data for Example 4.

The output of the program is given in three parts. The first and easiest to identify is the data echo check. All of the input data are echo checked for user convenience. The second part is the generated separable programming data. It is punched in the correct MPS/360 format. Lastly is the bounds section of the data. This punched data is necessary to satisfy the condition that the special separable variables, D_i^j, must be less than or equal to one. (The upper bound of the last special separable variable of each set may be removed.) Figure 2.14 shows the echo check and a printout of the data cards punched for Example 4. Notice that all of the 'MARKER' cards are punched and that the bounds section is separated from the rest of the deck by a line of asterisks for easy separation. ROW101 is the objective function and ROW102 is the constraint. The matrix is always related to the data as follows.

Matrix	Data
row 1	ROW101
row 2	ROW102
row 3	ROW103
.	.
.	.
.	.
row N	ROW100+N

N = # matrix rows = NROWS

FORTRAN SYMBOLS	EXPLANATION	FORMAT
NFUN	Number of separable variables	(2I5)
NROWS	Number of matrix (F(I,J)) rows	
START	Initial starting point (X_1^o)	(F10.0,I10,2F10.0)
INTER	Number of uniform partitions between ALOW and UPPER	
ALOW	Lower bound of the uniform region	
UPPER	Upper bound of the uniform retion	

Table 2.5. Input data formats for the FORTRAN program.

		Columns				
		1	2	3	4	5
	12345	0	0	0	0	0
card 1.	2	2				
	↑	↑				
	(NFUN)	(NROWS)				
card 2.		0.0	4	0.0	1.0	
card 3.		0.0	2	0.0	0.5	
		↑	↑	↑	↑	
		(START)	(INTER)	(ALOW)	(UPPER)	

Fig. 2.13. Sample input data to generate separable data for Example 4.

```
    SET101      'MARKER'                    'SEPORG'
    P1001       ROW101        0.43750       ROW102        0.06250
    P1001       GRID101      -0.25000
    P1002       ROW101        0.31250       ROW102        0.18750
    P1002       GRID101      -0.25000
    P1003       ROW101        0.18750       ROW102        0.31250
    P1003       GRID101      -0.25000
    P1004       ROW101        0.06250       ROW102        0.43750
    P1004       GRID101      -0.25000
    SET102      'MARKER'                    'SEPORG'
    P1005       ROW101        0.18750       ROW102        0.12500
    P1005       GRID102      -0.25000
    P1006       ROW101        0.06250       ROW102        0.37500
    P1006       GRID102      -0.25000
    ENDSET      'MARKER'                    'SEPEND'
**********************************************************************
 UP SEPBOUND    P1001         1.0
 UP SEPBOUND    P1002         1.0
 UP SEPBOUND    P1003         1.0
 UP SEPBOUND    P1004         1.0
 UP SEPBOUND    P1005         1.0
 UP SEPBOUND    P1006         1.0
```

```
ECHO CHECK FOR NFUN & NROWS
NFUN =     2
NROWS=     2

ECHO CHECK FOR VARIABLE     1
START=        0.0
INTER=          4
ALOW =        0.0
UPPER=        1.00

ECHO CHECK FOR VARIABLE     2
START=        0.0
INTER=          2
ALOW =        0.0
UPPER=        0.50
```

Fig. 2.14. FORTRAN output for Example 4. The Echo Check is printed and the separable data is punched.

The grid equations are GRID101 and GRID102. The grid equation rows in general are

variable 1	GRID101
variable 2	GRID102
.	.
.	.
.	.
variable m	GRID100+m

m = # variables = NFUN

The special variables (D_i^j) are P1001, P1002, etc. For each variable there will be either INTER or INTER+1 special variables depending on the values of START and ALOW.

With the program set-up and ready to go the user might want to know approximately how many data cards will be punched.

An upper bound may be established.

$$\text{the number of cards} \leq \sum_{i=1}^{m} n_i \left\{\frac{P}{2}\right\}^* + \sum_{i=1}^{m} n_i + (m+2)$$

P = number of matrix rows (NROWS)

n_i = number of partitions for variable i (either INTER or INTER+1)

m = number of variables used in separable functions (NFUN)

* - rounded up to the nearest integer.

There are minor limitations on this program. They are

NFUN $\leq$ 899, NROWS $\leq$ 899

and

$$\sum_{i=1}^{m} n_i \leq 8999$$

This will allow a problem with 400 variables, averaging 20 partitions per variable, and 400 constraints to be worked without altering the program (except for increasing the dimension of F(50, 50) and Row (50), and the possiblity of exceeding core). The output could consist of 1,616,402 cards. This enormous output could be overcome. The MPS/360 data could be stored sequentially on a direct access device. The MPS/360 control language program could then receive the data directly without reading enormous amounts of cards. The given FORTRAN routine can easily be altered to do this type of work, but the subject will not be pursued further this thesis.

Example 7

Example 4 is worked using the proposed subroutine. Figure 2.15 shows the MPS/360 control language program. The results are shown in Fig. 2.16. Remember how to determine which row is the objective function and which grid goes with a given variable.

2.6 SOLUTION TO A RELIABILITY MODEL

The following problem is from an article titled "Systems Reliability Subject to Multiple Nonlinear Constraints" [45], and will be discussed in Chapter 5. The problem is maximize

$$S = \ln(1 - .2^{x_1}) + \ln(1 - .15^{x_2}) + \ln(1 - .10^{x_3}) + \ln(1 - .35^{x_4}) + \ln(1 - .25^{x_5})$$

subject to the following constraints.

$$x_1^2 + 2x_2^2 + 3x_3^2 + 4x_4^2 + 2x_5^2 \leq 110$$

```
CONTROL PROGRAM COMPILER - MPS/360 V2-M8

C001                    PROGRAM
C002                    INITIALZ
C064                    MOVE(XDATA,'DATA-SET')
C065                    MOVE(XPBNAME,'EXAMPLE')
C066                    CONVERT
C067                    BCDOUT
C068                    MOVE(XOBJ,'ROW1C1')
C069                    MOVE(XRHS,'LIMITS')
C070                    TITLE('EXAMPLE')
C071                    SETUP('BOUND','SEPBOUND','MAX')
C072                    PRIMAL
C073                    SOLUTION
C074                    EXIT
C075                    PEND
```

Fig. 2.15. MPS/360 Control Language Program for Example 4. Notice that the objective function is defined as ROW101.

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

Fig. 2.16. MPS/360 solution for example 4 using the data generated by the FORTRAN program.

EXAMPLE

SOLUTION (OPTIMAL)

TIME = 0.28 MINS. ITERATION NUMBER = 5

...NAME...	...ACTIVITY...	DEFINED AS
FUNCTIONAL	1.17708	ROW101
RESTRAINTS		LIMITS
BOUNDS....		SEPBOUND

EXAMPLE

SECTION 1 - ROWS

	NUMBER	...ROW..	AT	...ACTIVITY...	SLACK ACTIVITY	..LOWER LIMIT.	..UPPER LIMIT.	.DUAL ACTIVITY
	1	ROW101	BS	1.17708	1.17708-	NONE	NONE	1.00000
	2	ROW102	UL	1.00000	.	NONE	1.00000	.16667-
A	3	GRID101	EQ	.	.	.	.	.
A	4	GRID102	EQ	.	.	.	.	.

EXAMPLE

SECTION 2 - COLUMNS

NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
5	X1	BS	.75000	.	.	NONE	.
6	X2	BS	.45833	.	.	NONE	.
7	P1C01	UL	1.00000	.43750	.	1.00000	.42708
8	P1C02	UL	1.00000	.31250	.	1.00000	.28125
9	P1C03	UL	1.00000	.18750	.	1.00000	.13542
10	P1C04	LL	.	.06250	.	1.00000	.01042-
11	P1C05	UL	1.00000	.18750	.	1.00000	.16667
12	P1C06	BS	.83333	.06250	.	1.00000	.

$$7(x_1 + e^{x_1/4}) + 7(x_2 + e^{x_2/4}) + 5(x_3 + e^{x_3/4}) + 9(x_4 + e^{x_4/4})$$
$$+4(x_5 + e^{x_5/4}) \leq 175$$

$$7(x_1 e^{x_1/4} + 8x_2 e^{x_2/4} + 8x_3 e^{x_3/4} + 6x_4 e^{x_4/4} + 9x_5 e^{x_5/4} \leq 200$$

Notice that START (x_i^0) cannot equal zero for any x_i since the objective function is a logarithmic sum. This means that the bounds 110, 175, and 200 must be adjusted and the objective function value will be off by a constant (see equations (12) thru (14)). Let START = 1.0 for each variable. With this starting value the adjusted bounds are 98, 101.992, and 151.218, respectively. Figure 2.17 shows the FORTRAN input data echo check. The subroutine FUNC is given in Appendix 2.

The MPS/360 control language program used to solve this problem is given in Appendix 2. This program is somewhat different from the programming used to solve Example 4. The procedure is

1. solve the problem with the special separable variables set at their lower bounds,
2. solve the problem with the special separable variables set at their upper bounds (this is the MPS/360 default procedure),
3. solve the problem by the dual algorithm with the special variables set at their upper bounds.

The procedure is recommended when using MPS/360 [37] to determine the existance of a local optimum solution, if it exists. Separable programming, at its best, will guarantee only a local optimum. One reason is that, unlike linear inequality constraints, nonlinear inequality constraints do not necessarily form a convex set. A second reason is that a nonlinear function is not

```
ECHC CHECK FOR NFUN & NROWS
NFUN =     5
NROWS=     4

ECHO CHECK FOR VARIABLE     1
START=       1.00
INTER=         10
ALOW =       2.00
UPPER=       3.00

ECHO CHECK FOR VARIABLE     2
START=       1.CC
INTER=         10
ALCW =       2.CC
UPPER=       3.CC

ECHC CHECK FOR VARIABLE     3
START=       1.00
INTER=         10
ALOW =       1.5C
UPPER=       2.50

ECHO CHECK FOR VARIABLE     4
START=       1.CC
INTER=         10
ALCW =       3.CC
UPPER=       4.CC

ECHC CHECK FOR VARIABLE     5
START=       1.CC
INTER=         1C
ALCW =       2.50
UPPER=       3.5C
```

Fig. 2.17 The FORTRAN data echo check for the reliability problem.

necessarily concave or convex. The only way to guarantee a stationary point is a global maximum is for a function to be concave, or if it is a global minimum the function to be convex. Since the linear approximation function of a separable nonlinear function will reflect its particular concave and convex properties, separable programming will, at its best, produce a local optimum solution. Notice that the problem is not encountered in Example 4 because the constraint forms a convex set and the objective function is a concave function producing a global maximum. The solution of the reliability problem is determined to yield a global maximum because each of the three procedures yields the same result. The control language program data echo check and the solution outputs are shown in Appendix 2.

The solution to the problem is

$x_1 = 2.70000$

$x_2 = 2.32929$

$x_3 = 2.10000$

$x_4 = 3.50000$

$x_5 = 2.80000$

This is compared to the given solution [45].

$x_1 = 2.6000$

$x_2 = 2.2816$

$x_3 = 2.0075$

$x_4 = 2.6882$

$x_5 = 3.3981$

It must be remembered that separable programming is an approximate method depending on the fineness of the grid equations for accuracy. The uniform grid for this solution is only .10. The effects of grid size on problem accuracy is dependent on the properties of the approximated functions.

2.7 THE APPLICATION OF THE SEPARABLE PROGRAMMING ALGORITHM TO THE GEOMETRIC PROGRAMMING PROBLEM

Geometric programming is a relatively new optimization technique. The method was originally developed by Duffin, Peterson, and Zener [15]. They defined the "posynomial" optimization problem which was later extended by Passy [40], and Wilde and Beightler [46] to include negative coefficients. The generalized problem is defined as

minimize

$$y_0 = \sum_{t=1}^{T_0} \sigma_{0t}\, c_{0t} \prod_{n=1}^{N} x_n^{a_{0tn}},\quad \sigma_{0t} \equiv \pm 1,\ c_{0t} > 0$$

subject to the constraints

$$y_m = \sum_{t=1}^{T_m} \sigma_{mt}\, c_{mt} \prod_{n=1}^{N} x_n^{a_{mtn}} \leq \sigma_m \equiv \pm 1$$

$$\sigma_{mt} \equiv \pm 1, \qquad m = 1, \ldots, M$$

$$c_{mt} > 0,\quad X_n > 0$$

This problem is termed the primal problem. It is greatly simplified by working with the associated dual problem, then solving for the primal variables in terms of the dual variables. The dual program is [46]

maximize

$$V(\bar{w}) = \sigma \left(\prod_{m=1}^{M} \prod_{t=1}^{T_m} \left(\frac{c_{mt}\, w_{m0}}{w_{mt}} \right)^{\sigma_{mt}\, w_{mt}} \right)^{\sigma}$$

subject to the linear constraints

$$\sum_{t=1}^{T_0} \sigma_{0t} w_{0t} = \sigma \equiv \pm 1 \qquad \text{(normality condition)}$$

$$\sum_{m=0}^{M} \sum_{t=1}^{T_m} \sigma_{mt} a_{mtn} w_{mt} = 0, \quad n = 1, \ldots, N \qquad \text{(orthogonality condition)}$$

$$w_{m0} \equiv \sigma_m \sum_{t=1}^{T_m} \sigma_{mt} w_{mt} \geq 0, \quad m = 1, \ldots, M$$

$$w_{mt} \geq 0$$

The following conditions are assumed.

$$w_{00} \equiv 1$$

$$\lim_{w_{mt} \to 0} \left(\frac{c_{mt} w_{m0}}{w_{mt}}\right)^{\sigma_{mt} w_{mt}} = 1$$

The relationship between the primal and the dual variables are

$$c_{0t} \prod_{n=1}^{N} x_n^{a_{0tn}} = w_{0t} \sigma V(\bar{w}), \quad t = 1, \ldots, T_0$$

and

$$c_{mt} \prod_{n=1}^{N} x_n^{a_{mtn}} = \frac{w_{mt}}{w_{m0}}, \qquad t = 1, \ldots, T_m, \quad m = 1, \ldots, M$$

Note that these equations are linear in $\ell n(X_n)$ and can be easily solved.

For the case when

$$\sigma = 1$$

$$\sigma_m = 1, \qquad m = 1, \ldots, M$$

$$\sigma_{mt} = 1, \quad t = 1, \ldots, T_m$$

a strict inequality relationship exists between y_0 and $V(\bar{w})$ as

$$y_0(\bar{X}) \geq V(\bar{w})$$

At the optimum solution, $\bar{X}^*$,

$$y_0(\bar{X}^*) = V(\bar{w}^*)$$

Under this condition the solution is a global optimum. This is not true of the generalized version which yields only a local optimum solution [46].

The beauty of geometric programming occurs when the total number of terms is one greater than the number of variables. In this case the solution is determined by solving the linear constraints without reference to the objective function.

Let

$$T = \sum_{m=0}^{M} T_m$$

The total degrees of difficulty is defined as

$$D = T - N - 1$$

When D is greater than zero the dual problem is not so easily optimized. The remaining portion of this section describes how to apply separable programming to optimize the dual geometric program with D degrees of difficulty.

Since the constraints are linear there is no problem in applying a separable programming algorithm to them. The only nonlinear portion is the objective function which is not obviously separable. The dual objective function can be made separable by a simple linear transformation.

$$V(\bar{w}) = \sigma \left(\prod_{m=0}^{M} \prod_{t=1}^{T_m} \left(\frac{c_{mt}}{w_{mt}}\right)^{\sigma_{mt} w_{mt}} \prod_{m=0}^{M} \prod_{t=1}^{T_m} (w_{m0})^{\sigma_{mt} w_{mt}} \right)^{\sigma}$$

$$= \sigma \left(\prod_{m=0}^{M} \prod_{t=0}^{T_m} \left(\frac{c_{mt}}{w_{mt}}\right)^{\sigma_{mt} w_{mt}} \prod_{m=0}^{M} w_{m0} \sum_{t=1}^{T_m} \sigma_{mt} w_{mt} \right)^{\sigma}$$

Since $\sigma_m = \pm 1$,

$$\sigma_m w_{m0} = \frac{w_{m0}}{\sigma_m} = \sum_{t=1}^{T_m} \sigma_{mt} w_{mt}$$

$$w_{00} \equiv 1$$

The final result is

$$V(\bar{w}) = \sigma \left(\prod_{m=0}^{M} \prod_{t=1}^{T_m} \left(\frac{c_{mt}}{w_{mt}}\right)^{\sigma_{mt} w_{mt}} \prod_{m=1}^{M} w_{m0}^{\sigma_m w_{m0}} \right)^{\sigma}$$

The linear transformation is

$$w_{m0} = \sigma_m \sum_{t=0}^{T_m} \sigma_{mt} w_{mt}$$

If $\sigma = 1$ then $V(\bar{w})$ is made separable by taking its natural log. If $\sigma = -1$ then the objective function can be maximized by minimizing (the positive function)

$$- V(\bar{w})$$

This function can also be made separable by taking its natural log.

Example 8

Minimize

$$y_0 = 4X_1 + 10X_2 + 4X_3 + 2(X_1^2 + X_2^2)^{\frac{1}{2}}$$

subject to the constraint

$$X_1X_2X_3 \geq 100 \ , \ X_i \geq 0$$

The problem may be formulated as a geometric programming problem by allowing

$$X_4 = (X_1^2 + X_2^2)^{\frac{1}{2}}$$

It now has the posynomial form of Duffin, Peterson, and Zener [15].

Minimize

$$y_0 = 4X_1 + 10X_2 + 4X_3 + 2X_4$$

subject to the constraints

$$y_1 = X_1^2 \, X_4^{-2} + X_2^2 \, X_4^{-2} \leq 1$$

$$y_2 = \frac{100}{X_1X_2X_3} \leq 1$$

The inequality constraint

$$X_1^2 + X_2^2 \leq X_4^2$$

is not unreasonable since the objective function is minimized only when $X_1^2 + X_2^2 = X_4^2$. The problem has 7-4-1 = 2 degrees of difficulty. The dual problem becomes maximize the product function

$$V = \left(\frac{4}{w_{01}}\right)^{w_{01}} \left(\frac{10}{w_{02}}\right)^{w_{02}} \left(\frac{4}{w_{03}}\right)^{w_{03}} \left(\frac{2}{w_{04}}\right)^{w_{04}} \left(\frac{1}{w_{11}}\right)^{w_{11}}$$

$$\left(\frac{1}{w_{12}}\right)^{w_{12}} \left(\frac{100}{w_{21}}\right)^{w_{21}} (w_{11} + w_{12})^{(w_{11}+w_{12})} (w_{21})^{w_{21}}$$

subject to the constraints

$$w_{01} + w_{02} + w_{03} + w_{04} = 1$$

$$w_{01} + 2w_{11} - w_{21} = 0$$

$$w_{02} + 2w_{12} - w_{21} = 0$$

$$w_{03} - w_{21} = 0$$

$$w_{04} - 2w_{11} - 2w_{12} = 0$$

To put the objective function in the separable form let

$$A_1 = w_{11} + w_{12}$$

$$A_2 = w_{21}$$

The objective function to be maximized becomes

$$\ell nV = w_{01} \ell n\left(\frac{4}{w_{01}}\right) + w_{02} \ell n\left(\frac{10}{w_{02}}\right) + w_{03} \ell n\left(\frac{4}{w_{03}}\right) + w_{04} \ell n\left(\frac{2}{w_{04}}\right)$$

$$+ w_{11} \ell n\left(\frac{1}{w_{11}}\right) + w_{12} \ell n \left(\frac{1}{w_{12}}\right) + w_{21} \ell n\left(\frac{100}{w_{21}}\right) + A_1 \ell n(A_1) + A_2 \ell n(A_2)$$

In addition to the previous linear constrants the constraints

$$A_1 - w_{11} - w_{12} = 0$$

$$A_2 - w_{12} = 0$$

are imposed on the problem.

A FORTRAN program was developed to generate data to solve dual geometric programming problems using MPS/360 and is given in Appendix 3. A solution was determined with a final uniform grid size of 0.0005. Values for the dual variables were determined as

$$w_{01} = 0.23117$$

$$w_{02} = 0.30500$$

$$w_{03} = 0.33333$$

$$w_{04} = 0.13050$$

$$w_{11} = 0.05108$$

$$w_{12} = 0.01417$$

$$w_{21} = 0.33333$$

The dual objective function optimal value was given as

$$V = \text{Exp}\ (4.47719) = 87.98708$$

Solving for the primal variables involved the solution of the following simultaneous equations (quite trivial for this simple example)

$$4X_1 = (0.23117)(87.98708)$$

$$10X_2 = (0.30500)(87.98708)$$

$$4X_3 = (0.33333)(87.98708)$$

$$2X_4 = (0.13050)(87.98708)$$

yielding,

$$X_1 = 5.085$$

$$X_2 = 2.684$$

$$X_3 = 7.332$$

$$X_4 = 5.741 \cong (X_1^2 + X_2^2)^{\frac{1}{2}}$$

Substituting these values into the primal objective function yielded

$$y_0 = 87.990$$

Since this is a posynomial problem

$$87.98708 < V(\bar{w}^*) = y_0(\bar{X}^*) < 87.990$$

The given solution is apparently very close to the true optimum.

2.8 CONCLUDING REMARKS

There are several important topics that should be mentioned. One already briefly mentioned in section 2.6 on the reliability problem is the subject of local versus global optimum solutions. Another is the question of convergence by employing finer grids for better approximations. Finally there is the question of creating separability. These topics have not been developed because they are problem dependent. That is, the methods employed in solving or alleviating the above problems cannot be generalized to include all cases.

There are two reasons why separable programming will yield a local optimum solution. One is that a nonlinear separable inequality constraint does not necessarily constitute a convex set. Likewise, the piecewise linear approximation function does not produce a convex set. It is basic to linear programming theory that a set of linear constraints necessarily form a convex set. How, then, can a piecewise linear approximation function form a non-convex set? The "restricted basis entry rule" is the reason. This rule

makes the piecewise linear approximation function nonlinear in nature. This function is not linear but psuedo-linear. A similar condition occurs with the objective function, the second reason why a local optimum solution can occur. The piecewise linear approximation function will in general be concave over one region and convex over another. The result is the possibility of a local optimum solution. When the inequality constraints form a convex set and the objective function is concave for a maximum and convex for a minimum, the solution will always be global. In general though, the separable programming algorithm produces, at its best, a local optimum solutuion. For an example involving a local optimum solution refer to the MPS/360 Linear and Separable Programming User's Manual [37], chapter 5.

Grid size is another important consideration in separable programming. Two things must be considered when determining grid size. The first is computational time and the second is computational precision. In theory, a sequence of piecewise linear approximation functions exists that converge uniformly to the separable function. In reality, we are limited in time and precision and thus, can settle only for an approximation. A good procedure is to use approximately 20 partitions per variable and solve the problem. Use this solution to partition a region around the approximate solution and solve the problem again. If the solution changes sufficiently try this again, but remember, as the grid becomes finer the coefficients of the special separable variables become smaller and the problem will suffer from lack of machine precision. Because of this, the proposed FORTRAN rountine will allow no more than 50 partitions per variable. Also, more partitions mean more iterations for the revised simplex method. Computational time can become excessive.

A final consideration when using separable programming is that it can be extended to other nonlinear problems by creating separability. The reliability problem of section 2.6 and the geometric programming extension are the result of a transformation. The objective functions were both originally a product function and were made separable by taking their logarithm. Transformation techniques are discussed in the literature [31, 39] and can be important since many nonlinear programming techniques are not as simple as separable programming.

In closing, separable programming is a powerful nonlinear programming technique with the same sensitivity analysis potential as linear programming. As with any nonlinear technique it will yield at least a local optimum solution, if it exists. With the proposed FORTRAN routine and MPS/360, separable programming can be performed with little difficulty. Its main disadvantage is precision, but this is of little consequence since most engineering problems need only a good approximation. Separable programming should be considered as a solution technique when a problem involves separable functions or functions that can be made separable by a transformation.

CHAPTER THREE

INTRODUCTION TO THE GENERALIZED REDUCED GRADIENT METHOD

3.1 INTRODUCTION

Attempts to solve nonlinear programming problems have resulted in many algorithms that work perfectly for special cases. Examples are separable programming, quadratic programming, geometric programming, and the Wolfe reduced gradient method[47,48]. In an attempt to develope a procedure that will handle the general nonlinear programming problem, the generalized reduced gradient method (GRG) has been proposed by Abadie and Carpentier [3, 4, 5]. The algorithm is generalized from the Wolfe reduced gradient method[47, 48].

The Wolfe reduced gradient method solves problems with a nonlinear objective function and linear constraints. It classes the variables as independent and dependent, and substitutes into the objective function the expressions obtained from the linear constraints in independent variables for the dependent variables. This essentially reduces the original problem to an unconstrained problem of reduced dimension. From this point, a variety of optimization techniques may be used. Applying the same concept to a nonlinear set of constraints complicates matters, but it is possible by using numerical methods.

The GRG method has been studied extensively and coded in FORTRAN by Abadie and Guigou [1, 6, 7, 29, 30]. Three generations of programs have been developed. The first was an experimental code called GRG 66 which was followed by the second code, GRG 69. Both procedures compared favorably to other nonlinear programming codes by ranking highly in the Colville study [7, 10]. GRG 69 was an exceptional standout among the other participants. An improved code, GREG, is the result of the previous work and promises to remain among the highly regarded nonlinear programming procedures.

3.2 DEFINITION OF NOTATION

Notation used in the generalized reduced gradient (GRG) algorithm is explicitly defined in this section. Symbols representing functions, vectors, matrices, gradients, and Jacobians are necessary to describe the algorithm. An N-dimensional vector, $\bar{X}$, is represented as

$$\bar{X} = \begin{pmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_N \end{pmatrix}$$

The transpose of a vector, $\bar{X}$, is $\bar{X}^T$. The objective function is represented by

$$f_0(\bar{X})$$

and a vector function of M constraints is given as

$$\bar{f}(\bar{X}) = 0$$

where

$$\bar{f}(\bar{X}) = \begin{pmatrix} f_1(\bar{X}) \\ f_2(\bar{X}) \\ \cdot \\ \cdot \\ \cdot \\ f_M(\bar{X}) \end{pmatrix}$$

Notice that a bar above a symbol indicates the presence of a vector. The gradient of a function, $f_i(\bar{X})$, is given as a row vector

$$\frac{\partial f_i}{\partial \bar{X}} = \left(\frac{\partial f_i}{\partial X_1}, \frac{\partial f_i}{\partial X_2}, \ldots, \frac{\partial f_i}{\partial X_N} \right), \qquad i = 0, 1, \ldots, M$$

The "gradient of a vector function", is represented by

$$\frac{\partial \bar{f}}{\partial \bar{X}} = \begin{pmatrix} \frac{\partial f_1}{\partial \bar{X}} \\ \frac{\partial f_2}{\partial \bar{X}} \\ \vdots \\ \frac{\partial f_M}{\partial \bar{X}} \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial X_1} & \frac{\partial f_1}{\partial X_2} & \cdots & \frac{\partial f_1}{\partial X_N} \\ \frac{\partial f_2}{\partial X_1} & \frac{\partial f_2}{\partial X_2} & \cdots & \frac{\partial f_2}{\partial X_N} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_M}{\partial X_1} & \frac{\partial f_M}{\partial X_2} & \cdots & \frac{\partial f_M}{\partial X_N} \end{pmatrix}$$

This operation results in an M × N matrix.

Another symbol used is related to the real number inequality relations. The notation

$$\bar{a} \leq \bar{X} \leq \bar{b}$$

implies

$$a_i \leq X_i \leq b_i , \qquad i = 1, \ldots, N$$

This relation forms the boundary conditions for the problem variables.

3.3 STATEMENT OF THE PROBLEM

The general nonlinear programming problem may be defined in the following form . Maximize

$$f_0(\bar{X})$$

subject to the constraints

$$\bar{f}(\bar{X}) = 0$$

$$\bar{a} \leq \bar{X} \leq \bar{b}$$

Note that an inequality constraint

$$G_i(\bar{X}) \leq 0$$

can be redefined as

$$f_i(\bar{X}) = G_i(\bar{X}) + s_i = 0, \qquad s_i \geq 0$$

The addition of a slack variable, s_i, is analogous to the linear programming technique.

Let the set formed by the constraint functions be described as

$$V = \{\bar{X} \mid f_i(\bar{X}) = 0,\ i = 1, \ldots, M\}$$

and the set formed by the boundary conditions as

$$P = \{\bar{X} \mid a_j \leq X_j \leq b_j, \quad j = 1, \ldots, N\}$$

The solution set of the problem becomes

$$S = V \cap P$$

the intersection of V and P. For a solution to exist, the set S must not be empty.

3.4 THE PRINCIPLE OF THE REDUCED GRADIENT

The generalized reduced gradient algorithm is based on a very basic optimization procedure which transforms a constrained optimization problem into one that is unconstrained. This is done by dividing the solution vector components into two groups, independent and dependent. The dependent variables, denoted by the dummy vector $\bar{y}$, are solved for in terms of the independent variables, represented by $\bar{x}$, via the constraint functions.

In terms of dependent and independent variables the constraint may be written as

$$\bar{f}(\bar{X}) = \bar{f}(\bar{x}, \bar{y}) = 0$$

The strategy of representing $\bar{y}$ in terms of $\bar{x}$ attempts to find a set of functions such that

$$\bar{y} = \phi(\bar{x})$$

The number of components of $\bar{y}$ is M, and the number of components of $\bar{x}$ is

(N-M). The vector function, $\bar{\phi}(\bar{x})$, is M-dimensional. Values for $\bar{y}$ are then substituted into the objective function such that

$$f_0(\bar{X}) = f_0(\bar{x}, \bar{y}) = f_0\left[\bar{x}, \bar{\phi}(\bar{x})\right] = F(\bar{x})$$

The problem is simplified to maximize

$$F(\bar{x})$$

subject to the constraint

$$\bar{a} \leq \bar{X} \leq \bar{b}$$

Notice that the dimension of the solution vector $\bar{x}$ of F, is only (N-M). The problem has been "reduced."

Example 1

Maximize

$$f_0(\bar{X}) = (X_1 - X_1^2) + (X_2 - X_2^2)$$

subject to the constraints

$$X_1^2 + X_2^2 \leq 1$$

$$0 \leq X_1 \leq 5$$

$$0 \leq X_2 \leq 3$$

The constraint, $f_1(\bar{X})$, may be defined as an equality by adding a slack variable, X_3.

$$f_1(\bar{X}) = X_1^2 + X_2^2 + X_3 - 1 = 0$$

$$0 \leq X_1 \leq 5$$

$$0 \leq X_2 \leq 3$$

$$0 \leq X_3 \leq \infty$$

Now, choose X_1 as the dependent variable with the assumption that the boundary condition, $0 \leq X_1 \leq 5$ will not be violated at the optimum solution. Thus,

$$y_1 = X_1 = \sqrt{1 - X_2^2 - X_3}$$

Substituting this into the objective function the problem becomes

maximize

$$F(\bar{x}) = F(X_2, X_3) = \sqrt{1 - X_2^2 - X_3} - (1 - X_2^2 - X_3) + (X_2 - X_2^2)$$

subject to the conditions

$$0 \leq X_2 \leq 3$$

$$0 \leq X_3 \leq \infty$$

The gradient of this problem and what is termed the "reduced gradient" of the original problem is

$$\frac{\partial F}{\partial \bar{x}} = \left\{ \frac{\partial F}{\partial X_2}, \frac{\partial F}{\partial X_3} \right\}$$

$$= \left\{ \frac{-X_2}{\sqrt{1 - X_2^2 - X_3}} + 1, \frac{\frac{1}{2}}{\sqrt{1 - X_2^2 - X_3}} + 1 \right\}$$

The term "reduced gradient" is used because the gradient of the original objective function, $f_0(\bar{X})$, has N components while the gradient of $F(\bar{x})$ has a "reduced" dimension of (N-M).

The conditions that determine an optimum solution, $\bar{x}^*$, are (for all j)

$$\frac{\partial F}{\partial x_j^*} = 0 \qquad \text{if} \qquad a_j < x_j^* < b_j$$

$$\frac{\partial F}{\partial x_j^*} \leq 0 \qquad \text{if} \qquad x_j^* = a_j$$

$$\frac{\partial F}{\partial x_j^*} \geq 0 \qquad \text{if} \qquad x_j^* = b_j$$

The optimal solution may be found trivially for this problem by solving the equations

$$\frac{-X_2}{\sqrt{1 - X_2^2 - X_3}} + 1 = 0$$

and

$$\frac{-\frac{1}{2}}{\sqrt{1 - X_2^2 - X_3}} + 1 = 0$$

This yields

$$X_2 = \frac{1}{2}$$

and

$$X_3 = \frac{1}{2}$$

By substituting these values into the constraint function it is determined that $X_1 = \frac{1}{2}$, satisfying the condition $0 \leq X_1 \leq 5$.

The direction of movement in the GRG algorithm is always determined by the independent variables and the resulting reduced gradient. The objective function may be written as

$$f_0(\bar{X}) = f_0(\bar{x}, \bar{y}) = F(\bar{x})$$

The reduced gradient is

$$\frac{\partial F}{\partial \bar{x}} = \frac{\partial f_0}{\partial \bar{x}} + \frac{\partial f_0}{\partial \bar{y}} \frac{\partial \bar{y}}{\partial \bar{x}}$$

$$\frac{\partial F}{\partial \bar{x}} = \left(\frac{\partial f_0}{\partial x_1} \cdots \frac{\partial f_0}{\partial x_j} \cdots \frac{\partial f_0}{\partial x_{N-M}}\right) + \left(\frac{\partial f_0}{\partial y_1} \cdots \frac{\partial f_0}{\partial y_i} \cdots \frac{\partial f_0}{\partial y_M}\right) \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots\cdots & \frac{\partial y_1}{\partial x_{N-M}} \\ \vdots & \frac{\partial y_i}{\partial x_j} & \vdots \\ \frac{\partial y_M}{\partial x_1} & \cdots\cdots & \frac{\partial y_M}{\partial x_{N-M}} \end{pmatrix}$$

The matrix, $\frac{\partial \bar{y}}{\partial \bar{x}}$, is difficult to determine explicitly. This can be calculated indirectly from the constraints.

$$\bar{f}(\bar{X}) = \bar{f}(\bar{x}, \bar{y}) = 0$$

$$\frac{\partial \bar{f}}{\partial \bar{x}} + \frac{\partial \bar{f}}{\partial \bar{y}} \frac{\partial \bar{y}}{\partial \bar{x}} = 0$$

$$\frac{\partial \bar{f}}{\partial \bar{x}} = - \left(\frac{\partial \bar{f}}{\partial \bar{y}}\right)^{-1} \left(\frac{\partial \bar{f}}{\partial \bar{x}}\right)$$

$$= \begin{pmatrix} \frac{\partial f_1}{\partial y_1} \cdots \frac{\partial f_1}{\partial y_M} \\ \vdots \\ \frac{\partial f_M}{\partial y_1} \cdots \frac{\partial f_M}{\partial y_M} \end{pmatrix}^{-1} \begin{pmatrix} \frac{\partial f_1}{\partial x_1} \cdots \frac{\partial f_1}{\partial x_{(N-M)}} \\ \vdots \\ \frac{\partial f_M}{\partial x_1} \cdots \frac{\partial f_M}{\partial x_{(N-M)}} \end{pmatrix}$$

The reduced gradient in the final form becomes,

$$-T_g = \frac{\partial F}{\partial \bar{x}} = \frac{\partial f_0}{\partial \bar{x}} - \frac{\partial f_0}{\partial \bar{y}} \left(\frac{\partial \bar{f}}{\partial \bar{y}}\right)^{-1} \left(\frac{\partial \bar{f}}{\partial \bar{x}}\right) \qquad (1)$$

Example 2

The reduced gradient for example 1 will be computed directly from the reduced gradient formula. Let

$$x_1 = X_2$$

$$x_2 = X_3$$

$$y_1 = X_1$$

$$\frac{\partial f_0}{\partial \bar{x}} = \left[\frac{\partial f_0}{\partial x_1}, \frac{\partial f_0}{\partial x_2}\right] = \left[\frac{\partial f_0}{\partial X_2}, \frac{\partial f_0}{\partial X_3}\right] = [1 - 2X_2, 0]$$

$$\frac{\partial f_0}{\partial \bar{y}} = \left[\frac{\partial f_0}{\partial y_1}\right] = \left[\frac{\partial f_0}{\partial X_1}\right] = [1 - 2X_1]$$

$$\frac{\partial \bar{f}}{\partial \bar{y}} = \left[\frac{\partial f_1}{\partial y_1}\right] = \left[\frac{\partial f_1}{\partial X_1}\right] = [2X_1]$$

$$\frac{\partial \bar{f}}{\partial \bar{x}} = \left[\frac{\partial f_1}{\partial x_1}, \frac{\partial f_1}{\partial x_2}\right] = \left[\frac{\partial f_1}{\partial X_2}, \frac{\partial f_1}{\partial X_3}\right] = [2X_2, 1]$$

The reduced gradient is

$$\frac{\partial F}{\partial \bar{x}} = [1 - 2X_2, 0] - [1 - 2X_1][2X_1]^{-1}[2X_2, 1]$$

$$= \left[1 - \frac{X_2}{X_1}, 1 - \frac{1}{2X_1}\right]$$

To put it in terms of Example 1 with $X_1 = \sqrt{1 - X_2^2 - X_3}$, the reduced gradient becomes

$$\frac{\partial F}{\partial \bar{x}} = \left[\frac{-X_2}{\sqrt{1 - X_2^2 - X_3}} + 1, \frac{\frac{1}{2}}{\sqrt{1 - X_2^2 - X_3}} + 1 \right]$$

There are two problems in the development of the GRG algorithm. The first is indicated in example 1. If the boundary condition for the dependent variable, $y_1 = X_1$, would have been, say, $1 \leq X_1 \leq 5$, the method would have yielded an optimal solution outside of the boundary, P. In the development of the GRG algorithm this is one condition that must be corrected. The second problem is the inversion of the square matrix, or Jacobian, $\frac{\partial \bar{f}}{\partial \bar{y}}$. This matrix must be non-singular.

As for the non-singularity property it is interesting to note the Implicit Function Theorem [8, 13, 36] which is relevant to the reduced gradient.

The Implicit Function Theorem

Suppose there exists a M-dimensional vector function

$$\bar{f}(\bar{X}) = 0$$

with continuous first order partial derivatives. The vector $\bar{X}^T = [X_1, X_2, \ldots, X_N]$ $(N > M)$ is partitioned into $\bar{X} = [\bar{x}, \bar{y}]$, where

$$\bar{x}^T = [x_1, \ldots, x_{N-M}]$$

and

$$\bar{y}^T = [y_1, \ldots, y_M]$$

Also assume that the determinent of the Jacobian, $\frac{\partial \bar{f}}{\partial \bar{y}}$, is not zero.

Then there exists one, and only one, vector function of M-dimension such that

$$\bar{y} = \bar{\phi}(\bar{x})$$

and each component has continuous first order partial derivatives.

This theorem does not insure a functional relation between $\bar{y}$ and $\bar{x}$ if the Jacobian is singular. It also states the existance, but not the form, of the function, $\bar{\phi}(\bar{x})$. How the values are obtained is not mentioned. In reality, they may not be solved for analytically as in Example 1, but numerical methods do exist making it possible to develop a generalized procedure.

It is worthwhile to note the relation of the Lagrange multipliers to the reduced gradient. The Lagrangian function is

$$L(\bar{X}, \bar{u}) = f_0(\bar{X}) - \bar{u}^T \bar{f}(\bar{X})$$

The Kuhn-Tucker condition for optimality at $\bar{X}$ is

$$\frac{\partial L}{\partial X_j} = \begin{cases} \leq 0 \text{ if } X_j = a_j \\ \geq 0 \text{ if } X_j = b_j \\ = 0 \text{ if } a_j < X_j < b_j \end{cases}$$

Therefore

$$\frac{\partial L}{\partial \bar{x}} = \frac{\partial f_0}{\partial \bar{x}} - \bar{u}^T \frac{\partial \bar{f}}{\partial \bar{x}} = \bar{V}^T \tag{2}$$

$$\frac{\partial L}{\partial \bar{y}} = \frac{\partial f_0}{\partial \bar{y}} - \bar{u}^T \frac{\partial \bar{f}}{\partial \bar{y}} = 0 \tag{3}$$

From equation (3) the vector of Lagrange multipliers is given as

$$\bar{u}^T = \frac{\partial f_0}{\partial \bar{y}} \left[\frac{\partial \bar{f}}{\partial \bar{y}}\right]^{-1} \tag{4}$$

Substituting equation (4) into equation (2) yields

$$\frac{\partial L}{\partial \bar{x}} = \frac{\partial f_0}{\partial \bar{x}} - \frac{\partial f_0}{\partial \bar{y}} \left[\frac{\partial \bar{f}}{\partial \bar{y}}\right]^{-1} \left[\frac{\partial \bar{f}}{\partial \bar{x}}\right] = \bar{V}^T$$

which is the reduced gradient given by equation (1).

3.5 THE GENERALIZED REDUCED GRADIENT ALGORITHM

The GRG algorithm depends on partitioning the solution vector, $\bar{X}$, into $\bar{x}$ and $\bar{y}$ (N-M and M-dimensioanl vectors, respectively). The partitioned vector, $\bar{x}$, is referred to as the set of independent variables. The vector, $\bar{y}$, is denoted as containing the dependent variables, which are often called basic variables in analogies to linear programming theory. The behavior of the dependent variables is of extreme importance to the GRG algorithm. The non-degenerency assumption of the algorithm is that for a given iteration there exits a set of dependent variables that are contained within the boundary conditions, P, and the Jacobian, $\frac{\partial \bar{f}}{\partial \bar{y}}$, is non-singular.

Using the above information, the basic GRG algorithm can be stated in five steps [2].

Step 1

The direction of movement is calculated at the starting point $\bar{X}^0$ and is denoted as $\bar{h}^0$. It is calculated in three sub-steps.

Compute the reduced gradient at the point $\bar{X}^0 = [\bar{x}^0, \bar{y}^0]$.

$$\bar{g}^{0T} = \frac{\partial F}{\partial \bar{x}^0} = \frac{\partial f_0}{\partial \bar{x}^0} - \frac{\partial f_0}{\partial \bar{y}^0} \left[\frac{\partial \bar{f}}{\partial \bar{y}^0}\right]^{-1} \left[\frac{\partial \bar{f}}{\partial \bar{x}^0}\right] \qquad (1)$$

Step 1.2

The projected reduced gradient, $\bar{p}^0$, is now calculated to satisfy the Kuhn-Tucker condition. For each component of the independent vector, $\bar{x}$,

$$p_i^0 = \begin{cases} 0 \text{ if } x_i = \text{lower bound and } g_i^0 \leq 0 \\ 0 \text{ if } x_i = \text{upper bound and } g_i^0 \geq 0 \\ g_i^0 \text{ otherwise} \end{cases}$$

$$i = 1,2, \ldots, N-M$$

Step 1.3

At this point the projected reduced gradient may become the direction of movement with $\bar{h}^0 = \bar{p}^0$. It may also be modified in several ways. The rapidly convergent conjugate gradient methods of Davidon, Fletcher and Powell, and Fletcher and Reeves may be used to modify the final direction of movement. A restriction on the choice is that $\bar{h}^0 \cdot \bar{p}^0 > 0$.

Step 2

It is desired to remain within the feasible region at all times. Since this is not possible, a direction of movement is picked that will remain close to the feasible region. The step from $\bar{x}^0$ is represented by

$$\bar{x}^0 + \theta \bar{h}^0$$

Likewise, for $\bar{y}^0$, the step is

$$\bar{y}^0 + \theta \bar{k}^0$$

The vector, $\bar{k}^0$, determines the direction of movement for $\bar{y}^0$.

Step 2.1

A desired movement would be along the surface of the constraints, that is tangent to them. This is accomplished by finding the tangent to

$\bar{f}(\bar{x}^0 + \theta\bar{h}^0, \bar{y}^0 + \theta\bar{k}^0) = 0$ at the point $[\bar{x}^0, \bar{y}^0]$. It becomes

$$\frac{\partial \bar{f}}{\partial \bar{x}^0} \bar{h}^0 + \frac{\partial \bar{f}}{\partial \bar{y}^0} \bar{k}^0 = 0$$

or

$$\bar{k}^0 = - \left[\frac{\partial \bar{f}}{\partial \bar{y}^0}\right]^{-1} \left[\frac{\partial \bar{f}}{\partial \bar{x}^0}\right] \bar{h}^0 \qquad (5)$$

Step 2.2

The problem is now to maximize

$$f_0(\bar{x}^0 + \theta\bar{h}^0, \bar{y}^0 + \theta\bar{k}^0)$$

for θ. This may be done by an one-dimensional search such as the Golden section search, Fibonacci search, or by quadratic interpolation.

Step 3

Calculate

$$\tilde{x}^1 = \bar{x}^0 + \theta\bar{h}^0, \text{ and } \tilde{y}^1 = \bar{y}^0 + \theta\bar{k}^0$$

The values for the independent variables are projected into the bounds, $\bar{a} \leq \bar{X} \leq \bar{b}$. The relation is

$$x_j^1 = \begin{cases} \text{lower bound, if } x_j^0 + \theta h_j^0 \leq \text{lower bound} \\ \text{upper bound, if } x_j^0 + \theta h_j^0 \geq \text{upper bound} \\ x_j^0 + \theta h_j^0, \text{ otherwise} \end{cases}$$

$$j = 1, \ldots, N\text{-}M$$

In general the calculated dependent variables, $\tilde{y}^1$, do not satisfy the feasibility conditions. It is hoped, though, that the values are "close" to feasibility. Indeed, if θ is small this condition exists. Usually the

vector, $\tilde{y}^1$, will be the starting point for finding $\bar{y}^1$ iteratively at step 4.

Step 4

A feasible solution is developed by solving the system

$$\bar{f}(\bar{x}^{-1}, \bar{y}) = 0$$

This may be done by an iterative method. The existance of

$$\bar{y} = \bar{\phi}(\bar{x}^{-1})$$

is insured by the Implicit Function Theorem under the non-degenerency conditions. If a component of $\bar{y}$ violates a boundary condition, a degenerency, then a change of basis occurs as described in section 7. There are two ends to this procedure.

Step 4.1

If the iterative procedure does not converge to a $\bar{y}^1$, then $\bar{x}^{-1}$ is out of the functional domain. This is alleviated by reducing θ(i.e., set $\theta = \frac{1}{2}\theta$) and returning to step 3. An example of this would be to find y given x = 2 for

$$x^2 + y^2 = 1$$

When using Newton's method, convergence of $\bar{y}^{-1}$ depends on the shape of the vector function, $\bar{f}(\bar{X})$, even if $\bar{x}^{-1}$ is within its domain. The problem is alleviated by restricting the procedure to a finite number of iterations, then reducing θ and returning to step 3.

Letting $\bar{y}^{-1}$ be the solution obtained, it must be determined if the new solution, $\bar{X}^{-1} = (\bar{x}^{-1}, \bar{y}^{-1})$, improves the objective function. Since $\bar{y}^{-1}$ is not on the tangent which guarantees improvement then it will improve the objective function only if close to $\tilde{y}^1$. If the objective function is not improved,

decrease θ by one-half and return to Step 3.

Step 5

In this step it is normal to set $\bar{X}^0 = \bar{X}^1$ and repeat the algorithm. But if for some reason a better value for θ may be determined by previous information, a return to step 3 is possible.

A flow diagram exemplifying the procedure is given in Fig. 3.1.

Normally, the starting point, $\bar{X}^0$, is a feasible solution but this is not necessary. Any non-feasible $\bar{X}^0$ can be made feasible by adding artificial variables to satisfy the equality constraints, then forcing them to zero values by penalizing the objective function. Let an artificial variable, X_{N+i}, satisfy the condition

$$0 \leq X_{N+i} \leq \infty$$

If, at the starting point, $\bar{X}^0$, the $i^{\underline{th}}$ constraint violates the equality condition and it is greater than zero, $f_i(\bar{X}^0) > 0$, it is made feasible by subtracting the artificial variable, X_{N+i}.

$$f_i(\bar{X}^0) - X_{N+i} = 0$$

For the less than zero condition, $f_i(\bar{X}^0) < 0$, the equation is formulated by adding the artificial variable.

$$f_i(\bar{X}^0) + X_{N+i} = 0$$

There are two methods for obtaining a feasible solution, the penalty method and the two-phase method. For the penalty method let M_i be a large number. The objective function to be maximized becomes

$$f_0(\bar{X}) - \sum_{i \in NF} M_i X_{N+i}, \qquad NF = \{i \mid f_i(\bar{X}) \neq 0\}$$

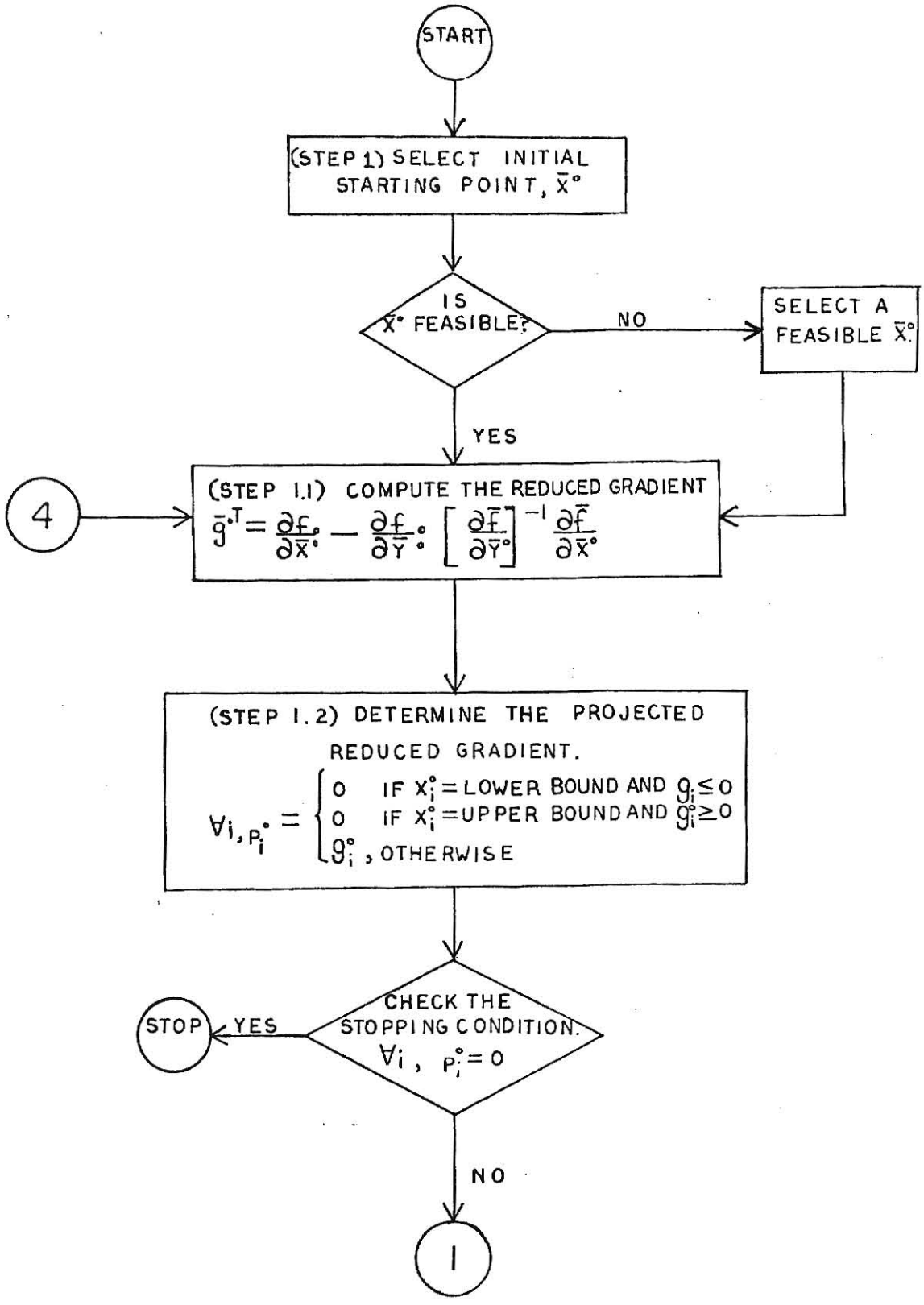


Fig. 3.1. Computer flow diagram for the GRG algorithm.

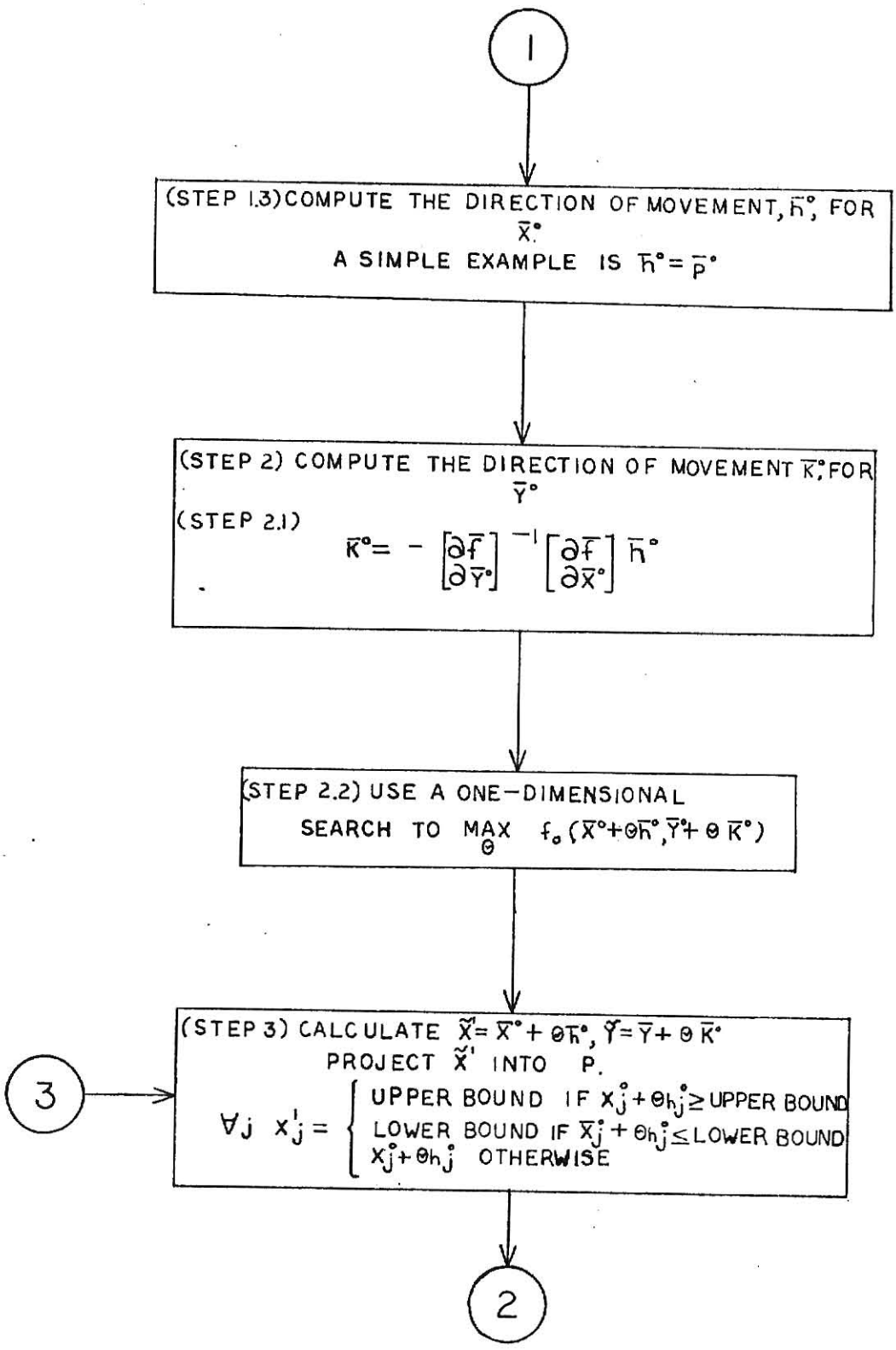


Fig. 3.1. (continued)

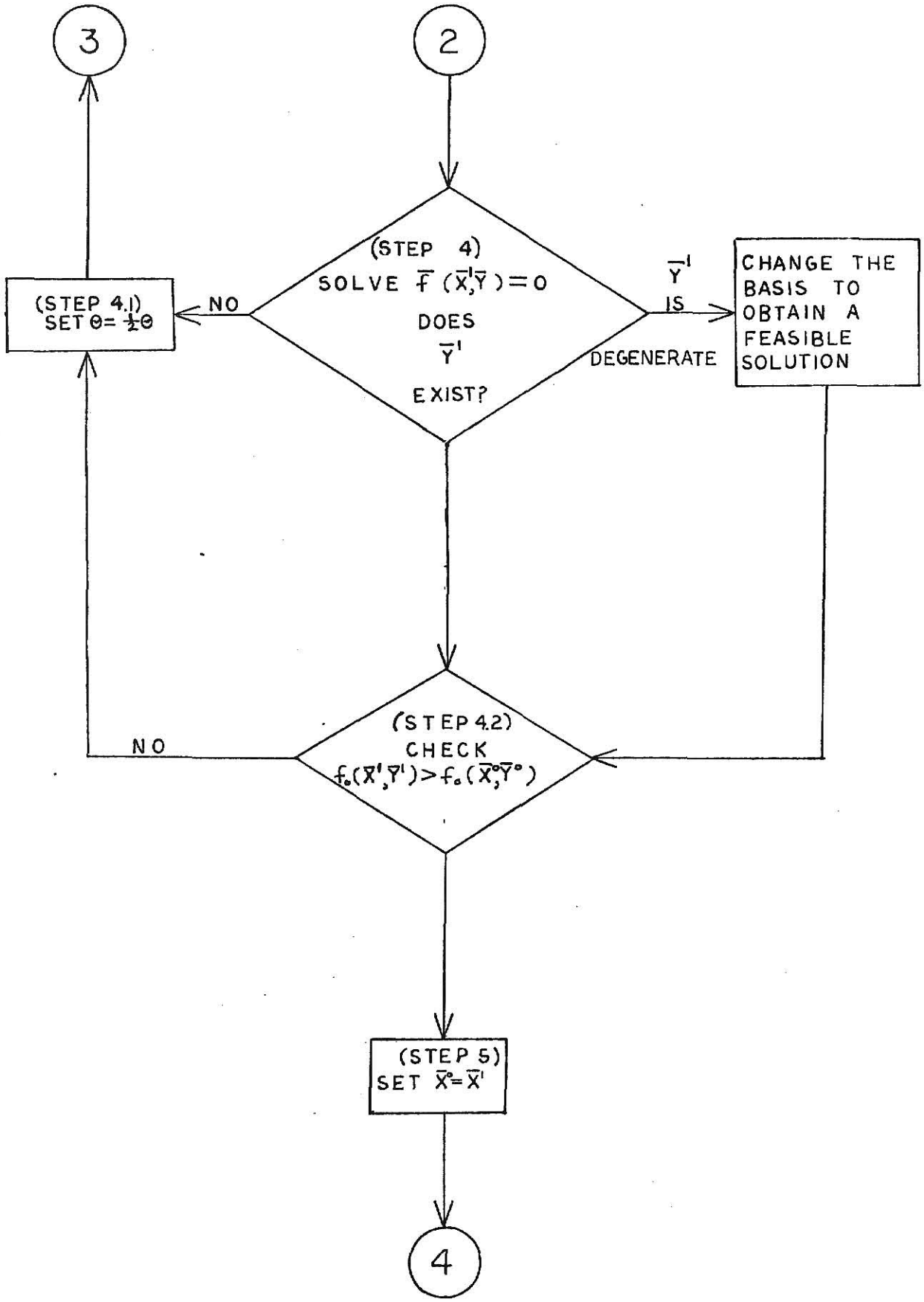


Fig. 3.1. (continued)

The artificial variables are forced to zero by the penalty, then disregarded. The two-phase method occurs in two steps. The first step is to maximize

$$- \sum_{i \in NF} X_{N+i}, \qquad NF = \{i \mid f_i(\bar{X}) \neq 0\}$$

When all of the artificial variables are zero the solution, $\bar{X}$, is feasible for $\bar{f}(\bar{X}) = 0$. In the second step the optimization problem continues normally. If it is impossible to force the artifical variables to zero values then a feasible solution may not exist. Both of the above techniques are similar to linear programming methods.

Theoretically, the stopping condition for the GRG algorithm is when $p_i^0 = 0$, $i = 1, \ldots, N-M$. This is not possible when using a finite number of digits for a real number. When convergence occurs, three appropriate stopping criteria are

1. $\| \bar{p}^0 \| = \sqrt{\sum_{i=1}^{N-M} (p_i^0)^2} < \varepsilon_1$
2. $p_i^0 < \varepsilon_2, \qquad i = 1, \ldots, N-M$
3. $\left| f_0(\bar{X}^1) - f_0(\bar{X}^0) \right| < \varepsilon_3$

3.6 A NUMERICAL EXAMPLE

A numerical example is solved for illustrating the GRG algorithm in detail.

Example 3

Maximize

$$f_0(\bar{X}) = (2X_1 - \frac{1}{2} X_1^2) + (3X_2 - \frac{1}{2} X_2^2)$$

subject to the constraints

$$f_1(\bar{X}) = X_1^2 + X_2^2 + X_3 - 1 = 0$$

$$X_i \geq 0, \qquad i = 1,2,3$$

Note that the variable X_3 is essentially a slack variable for an inequality constraint.

The problem will be solved using the GRG algorithm with the exceptions that the value of the dependent variable will be determined analytically instead of using an iterative method, and the optimization of θ will be accomplished by direct differentiation. Table 3.1 summarizes the following numerical calculations and Figures 3.2 and 3.3 show the results graphically.

The independent variables, $\bar{x}$, and the dependent variable, $\bar{y}$, are chosen as

$$\bar{x} = \begin{bmatrix} X_1 \\ X_3 \end{bmatrix}, \quad \bar{y} = [X_2]$$

The reduced gradient for the given set of independent variables is (refer to equation (1))

$$\frac{\partial F}{\partial \bar{x}} = \left[\frac{X_2(2-X_1) - X_1(3-X_2)}{X_2} , \frac{X_2 - 3}{2X_2} \right] \tag{6}$$

The formula for $\bar{k}$ is (refer to equation (5))

$$\bar{k} = \left[-\frac{X_1}{X_2} , -\frac{1}{2X_2} \right] \bar{h}$$

Maximization of $f_0(\bar{X})$ is accomplished by direct differentiation. Let,

$$X_1 = X_1^0 + \theta d_1$$

$\bar{X}^0=\begin{bmatrix}0.5\\0.5\\0.5\end{bmatrix}$	STEP 1				STEP 2		STEP 3			STEP 4			STEP 5
	$\bar{x}=\begin{bmatrix}X_1\\X_3\end{bmatrix}$	$\bar{Y}=[X_2]$	$\frac{\partial F}{\partial \bar{x}}$	$\bar{h}^0=\bar{p}^0$	$\bar{k}^0$	Θ	$\tilde{\bar{x}}^1=\bar{x}^0+\Theta\bar{h}^0$	$\tilde{\bar{Y}}^1=\bar{Y}^0+\Theta\bar{k}^0$	project $\tilde{\bar{x}}^1\rightarrow\bar{x}^1$	$\bar{Y}^1$	$f_o(\bar{X}^0)$	$f_o(\bar{X}^1)$	$\bar{X}^1\rightarrow\bar{X}^0$
1	$\begin{bmatrix}.5\\.5\end{bmatrix}$	[.5]	[-1, -2.5]	$\begin{bmatrix}-1\\-2.5\end{bmatrix}$	[3.5]	.547	$\begin{bmatrix}-.047\\-.867\end{bmatrix}$	[2.41]	$\begin{bmatrix}0\\0\end{bmatrix}$	[1]	2.25	2.50	$\begin{bmatrix}0\\1\\0\end{bmatrix}$
2	$\begin{bmatrix}0\\0\end{bmatrix}$	[1]	[2, -1]	$\begin{bmatrix}2\\0\end{bmatrix}$	[0]	1.0	$\begin{bmatrix}2\\0\end{bmatrix}$	[1]	$\begin{bmatrix}2\\0\end{bmatrix}$	DOES NOT EXIST			
2.1						.5	$\begin{bmatrix}1\\0\end{bmatrix}$	[1]	$\begin{bmatrix}1\\0\end{bmatrix}$	[0]	NO IMPROVEMENT 2.50	1.50	
2.2						.25	$\begin{bmatrix}.5\\0\end{bmatrix}$	[1]	$\begin{bmatrix}.5\\0\end{bmatrix}$	[.816]	2.50	3.10	$\begin{bmatrix}.5\\.816\\0\end{bmatrix}$
3	$\begin{bmatrix}.5\\0\end{bmatrix}$	[.816]	[.277, -1.34]	$\begin{bmatrix}.277\\0\end{bmatrix}$	[.170]	.193	$\begin{bmatrix}.554\\0\end{bmatrix}$	[.849]	$\begin{bmatrix}.554\\0\end{bmatrix}$	[.834]	3.10	3.10	$\begin{bmatrix}.554\\.834\\0\end{bmatrix}$

Table 3.1. Numerical results of Example 3.

$$X_2 = X_2^0 + \theta d_2$$

$$X_3 = X_3^0 + \theta d_3$$

where d_1, d_2, and d_3 are the directions of move for X_1^0, X_2^0, and X_3^0.

$$\frac{df_0}{d\theta} = \frac{\partial f_0}{\partial X_1}\frac{dX_1}{d\theta} + \frac{\partial f_0}{\partial X_2}\frac{dX_2}{d\theta} = 0$$

or

$$\frac{df_0}{d\theta} = (2 - X_1)\, d_1 + (3 - X_2)d_2 = 0$$

$$\frac{df_0}{d\theta} = (2 - X_1^0 - \theta d_1)\, d_1 + (3 - X_2^0 - \theta d_2)\, d_2 = 0$$

This gives

$$\theta = \frac{(2 - X_1^0)\, d_1 + (3 - X_2^0)d_2}{(d_1^2 + d_2^2)} \qquad (8)$$

Since X_2 is the dependent variable, $d_2 = k_1$, and for the independent variable, X_1, $d_1 = h_1$.

Iteration 1

Step 1

The starting point is picked as $\bar{X}^{0^T} = [0.5, 0.5, 0.5]$. Dividing this vector into independent and dependent variables gives

$$\bar{x}^0 = \begin{bmatrix} X_1 \\ X_3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

$$\bar{y}^0 = [X_2] = [0.5]$$

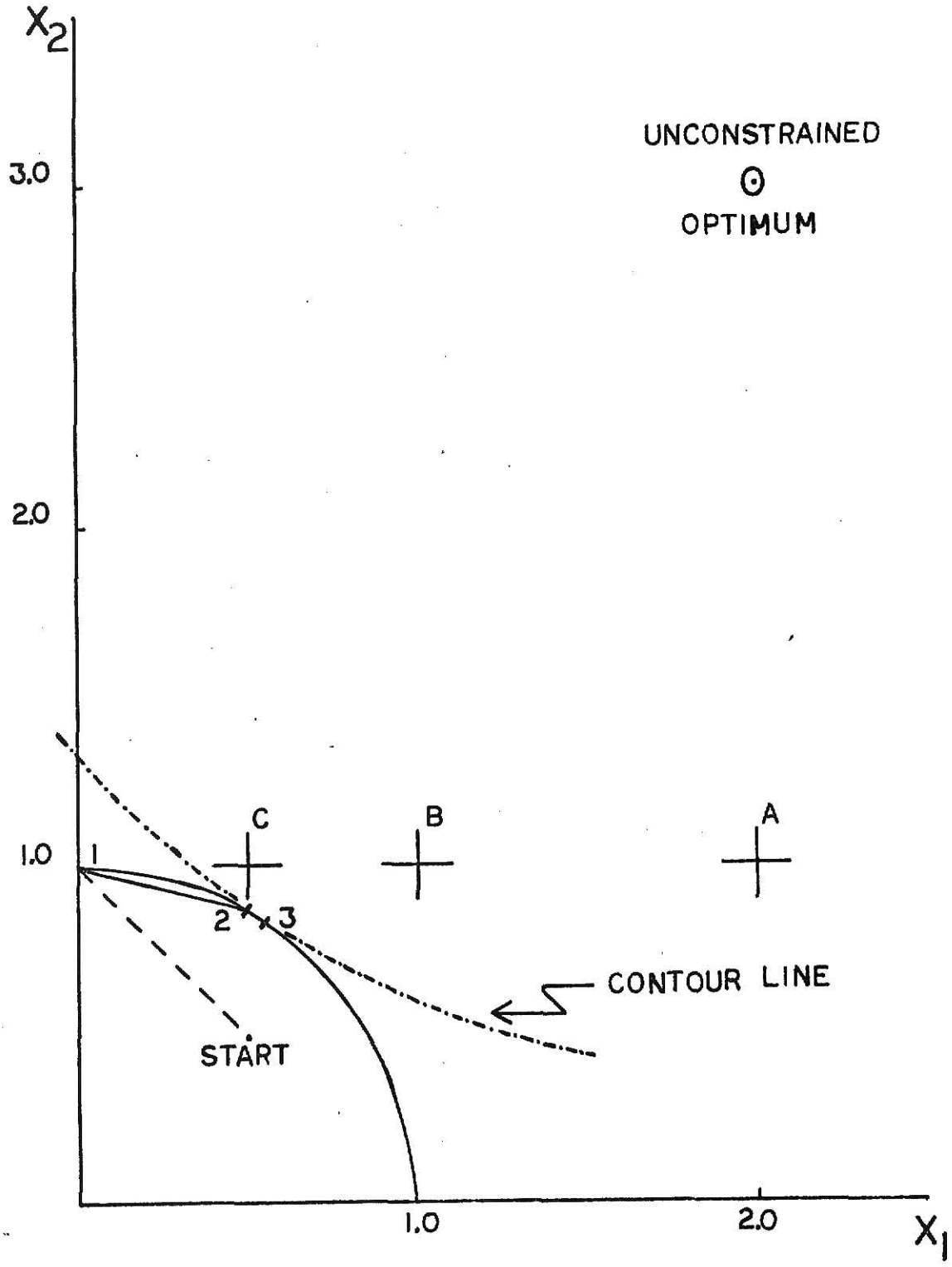


Fig. 3.2. Two-dimensional graph of the problem solving procedure of Example 3.

At this point the non-degenerency of $\bar{y}^1$ is assumed.

Step 1.1

From equation (6)

$$\bar{g}^{0^T} = \frac{\partial F}{\partial \bar{x}} = \left\{ \frac{0.5(2-0.5)-0.5(3-0.5)}{0.5} \ , \ \frac{0.5-3}{2(0.5)} \right\} = [-1.0, \ -2.5].$$

Step 1.2

$$\bar{p}^0 = \begin{bmatrix} -1.0 \\ -2.5 \end{bmatrix}$$

Step 1.3

The direction of movement is along the reduced gradient in this example.

$$\bar{h}^0 = \bar{p}^0 = \begin{bmatrix} -1.0 \\ -2.5 \end{bmatrix}$$

Step 2

Step 2.1

The direction of movement for the dependent variable is in the direction of $\bar{k}$. From equation (7),

$$\bar{k}^0 = [-1.0, \ -1.0] \begin{bmatrix} -1.0 \\ -2.5 \end{bmatrix} = [3.5]$$

Step 2.2

The optimizing value for θ is calculated from equation (8).

$$\theta = \frac{(2.0-0.5)\ (-1) + (3.0-0.5)\ (3.5)}{(-1.0)^2 + (3.5)^2} = 0.547$$

Step 3

The values for $\tilde{x}^1$ and $\tilde{y}^1$ are

$$\tilde{x}^1 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} + 0.547 \begin{bmatrix} -1.0 \\ -2.5 \end{bmatrix} = \begin{bmatrix} -0.047 \\ -0.867 \end{bmatrix}$$

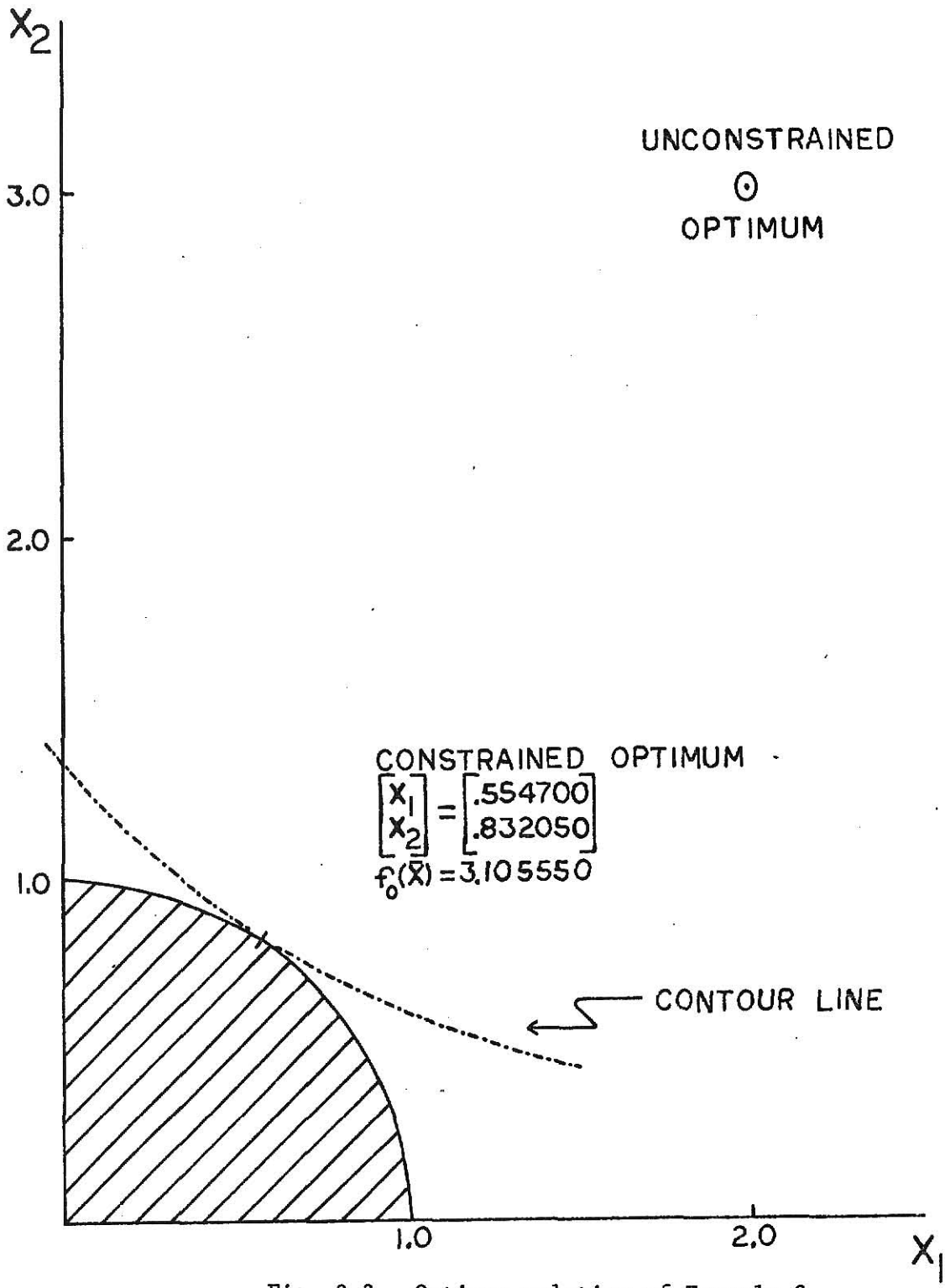


Fig. 3.3. Optimum solution of Example 3.

$$\tilde{y}^1 = [0.5] + 0.547\ [3.5] = [2.41]$$

The independent variables must be projected into the boundary set, $\bar{a} \leq \bar{X} \leq \bar{b}$. The projected independent vector is

$$\bar{x}^1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Step 4

The dependent variable is solved for via the constraint, $f_1(\bar{X}) = 0$.

$$\bar{y}^1 = \begin{pmatrix} X_2 \end{pmatrix} = \left(\sqrt{1 - X_1^2 - X_3} \right)$$

$$\bar{y}^1 = \begin{pmatrix} X_2 \end{pmatrix} = [1.0]$$

Step 4.1

This step is by-passed because convergence to $\bar{y}^1$ has occurred.

Step 4.2

In this step, improvement of the objective is checked.

$$f_0(\bar{X}^0) = 2.25 < 2.50 = f_0(\bar{X}^1)$$

Step 5

At this point, $\bar{X}^1$ becomes the starting point, $\bar{X}^0$, for the second iteration.

$$\bar{X}^0 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

Iteration 2

Since the slack variable, X_3, is zero, the next problem step is shown graphically in Fig. 3.2.

$$\bar{x}^0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \bar{y}^0 = [1]$$

Step 1.1

From equation (6),

$$\bar{g}^{0T} = \frac{\partial F}{\partial \bar{x}^0} = \left[\frac{1(2-0)-0(3-1)}{1}, \frac{1-3}{2(1)} \right] = [2, 1].$$

Step 1.2

Since $x_3^0 = 0$, the lower bound, and $g_2^0 \leq 0$, the projected reduced gradient differs from $\bar{g}^0$.

$$\bar{p}^0 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

Step 1.3

$$\bar{h}^0 = \bar{p}^0 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

Step 2

Step 2.1

From equation (7),

$$\bar{k} = [0,-1] \begin{bmatrix} 2 \\ 0 \end{bmatrix} = [0]$$

Step 2.2

From equation (8),

$$\theta = \frac{(2-0)\ (2) + (3-1)\ (0)}{(2^2 + 0)} = 1$$

Step 3

$$\tilde{x}^1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 1 \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$$

$$\tilde{y}^1 = [1] + 1\ [0] = [1]$$

$$\bar{x}^1 = \tilde{x}^1$$

The boundary condition is satisfied with $\tilde{x}^1$.

Step 4

A problem occurs at this point. A value for $\bar{y}^1$ does not exist. This is shown graphically in Fig. 3.2. Point A is located outside of the domain of $f_1(\bar{X})$.

Step 4.1

At this point, θ is reduced by one-half and the procedure returns to Step 3. This is referred to as Iteration 2.1 in Table 3.1.

Iteration 2.1

Step 3

$$\theta = 0.5$$

$$\tilde{x}^1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + 0.5 \begin{pmatrix} 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\tilde{y}^1 = [1] + 0.5\ [0] = [1]$$

$$\bar{x}^1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

The boundary condition is satisfied at this point. Refer to point B in Fig. 3.2.

Step 4

$$\bar{y}^1 = [X_2] = \sqrt{1 - X_1^2 - X_3}$$

$$\bar{y}^1 = [X_2] = [0]$$

Step 4.1

This step is by-passed because convergence to a feasible solution has occurred.

Step 4.2

The objective function has a value of 1.50 which is not an improvement over the previous solution, $\bar{X}^0$. At this time reduce θ by one-half and return to Step 3. This will be referred to as Iteration 2.2 in Table 3.1.

Iteration 2.2

Step 3

$$\theta = 0.25$$

$$\bar{x} = \begin{pmatrix} .5 \\ 0 \end{pmatrix}$$

Refer to point C in Fig. 3.2.

Step 4

Solving for the dependent variable yields

$$\bar{y}^1 = [0.816]$$

Step 4.1

This step is by-passed because convergence to a feasible solution has occurred.

Step 4.2

The value of the objective function improves to 3.10

Step 5

The next starting point is

$$\bar{X}^0 = \begin{pmatrix} 0.5 \\ 0.816 \\ 0 \end{pmatrix}$$

Iteration 3

This iteration continues smoothly in the same manner as the first iteration. Figure 3.2 indicates that convergence will be rapid. The solution is shown in Fig. 3.3 as

$$\bar{X} = \begin{pmatrix} \frac{2\sqrt{13}}{13} \\ \frac{3\sqrt{13}}{13} \\ 0 \end{pmatrix} = \begin{pmatrix} 0.554700 \\ 0.832050 \\ 0.0 \end{pmatrix}$$

The third iteration yields

$$\bar{X}^1 = \begin{pmatrix} 0.554 \\ 0.834 \\ 0.0 \end{pmatrix}$$

which is approaching the optimum. It is apparent that more than slide-rule precision is needed for a fourth iteration.

3.7 THE PSUEDO-NEWTON METHOD AND CHANGING THE BASIS

In general, numerical calculations are necessary for evaluating

$$\bar{y} = \bar{\phi}\ (\bar{x})$$

in step 4 of the GRG algorithm. It is desired that an iterative method produce a sequence, $\{\tilde{y}^1, \ldots, \tilde{y}^t\}$, that satisfies the stopping conditon

$$||\bar{f}(\bar{x}^1, \tilde{y}^t)|| < \varepsilon$$

where the symbol, $||\bar{f}(\bar{x}^1, \tilde{y}^t)||$, indicates the norm of the vector $\bar{f}(\bar{x}^1, \tilde{y}^t)$. Upon convergence $\tilde{y}^t$ becomes $\bar{y}^1$.

One of the most efficient procedures is the generalized Newton method.

The Taylor series expansion around the point $\tilde{y}^i$ yields an approximate value for $\bar{f}(\bar{x}^1, \bar{y}^1)$ as

$$\bar{f}(\bar{x}^1, \bar{y}^1) \cong \bar{f}(\bar{x}^1, \tilde{y}^i) + \frac{\partial \bar{f}}{\partial \tilde{y}^i} (\bar{y}^1 - \tilde{y}^i)$$

Since $\bar{f}(\bar{x}^1, \bar{y}^1) = 0$, the iterative formula becomes

$$\bar{y}^1 \cong \tilde{y}^{i+1} = \tilde{y}^i - \left(\frac{\partial \bar{f}}{\partial \tilde{y}^i}\right)^{-1} \bar{f}(\bar{x}^1, \tilde{y}^i)$$

When the above approximation is good, it converges to the solution very rapidly. Sometimes it may not converge making it necessary to limit the maximum number of iterations. There are two reasons for nonconvergence. One is that the starting point is not close enough to the final solution. This is a problem caused by rapidly changing surface curvature, that is, the elements of the Jacobian, $\frac{\partial \bar{f}}{\partial \bar{y}}$, are "very" nonlinear. A second cause is that a real solution for $\bar{y}^1$ does not exist for the current values of the independent variables in $\bar{x}^1$. Both of these problems are alleviated in step 4.1 of the GRG algorithm in which θ is reduced by 1/2. This causes the independent vector, $\bar{x}^1$, and the starting point, $\tilde{y}^1$, to be closer to the feasible point $[\bar{x}^0, \bar{y}^0]$.

To determine $\bar{y}$, Abadie and Carpentier have proposed a modified Newton method and termed it the Pseudo-Newton method [2,5]. The method satisfies the boundary condition, P, by a logical procedure. When a degenerency occurs with a component, y_i^t, converging outside of P, the value of that component is set to its lower or upper bound. The degenerate, dependent component, y_i^t, is then interchanged with an independent component, x_j, thus changing the composition of the independent vector, $\bar{x}$, and the dependent vector, $\bar{y}$. The procedure has been termed by Abadie and Carpentier as changing the basis [2,5]. With the new basis the generalized Newton method is used to determine

a new value $\tilde{y}^t$.

Improvement of the objective function is not insured when a change of basis occurs during an iteration. Improvement is insured for values of θ less than a maximum value θ^* in the direction of the reduced gradient. But when a change of basis occurs the independent vector, $\bar{x}$, changes and is no longer related to the reduced gradient of the previous basis. For this reason improvement is not insured regardless of how many times θ has been reduced. Abadie has noted that the condition can cause the GRG 66 and GRG 69 codes to cycle [2]. He has studied the degenerency problem extensively [1] and has implemented an anticycle procedure in the GREG code.

Example 4

Suppose a hypothetical objective function is subject to the constraints

$$x_1^2 + x_2^2 \leq 4$$

$$x_1^2 + x_3 \leq 1$$

$$x_i \geq 0 \qquad i = 1,2,3$$

The constraints are pictured in Fig. 3.4 and, by using slack variables, have the form

$$x_1^2 + x_2^2 + x_4 = 4$$

$$x_1^2 + x_3 + x_5 = 1$$

$$x_i \geq 0, \; i = 1,2,3,4,5.$$

Assume the i^{th} iteration yields (point 1 in Fig. 3.4)

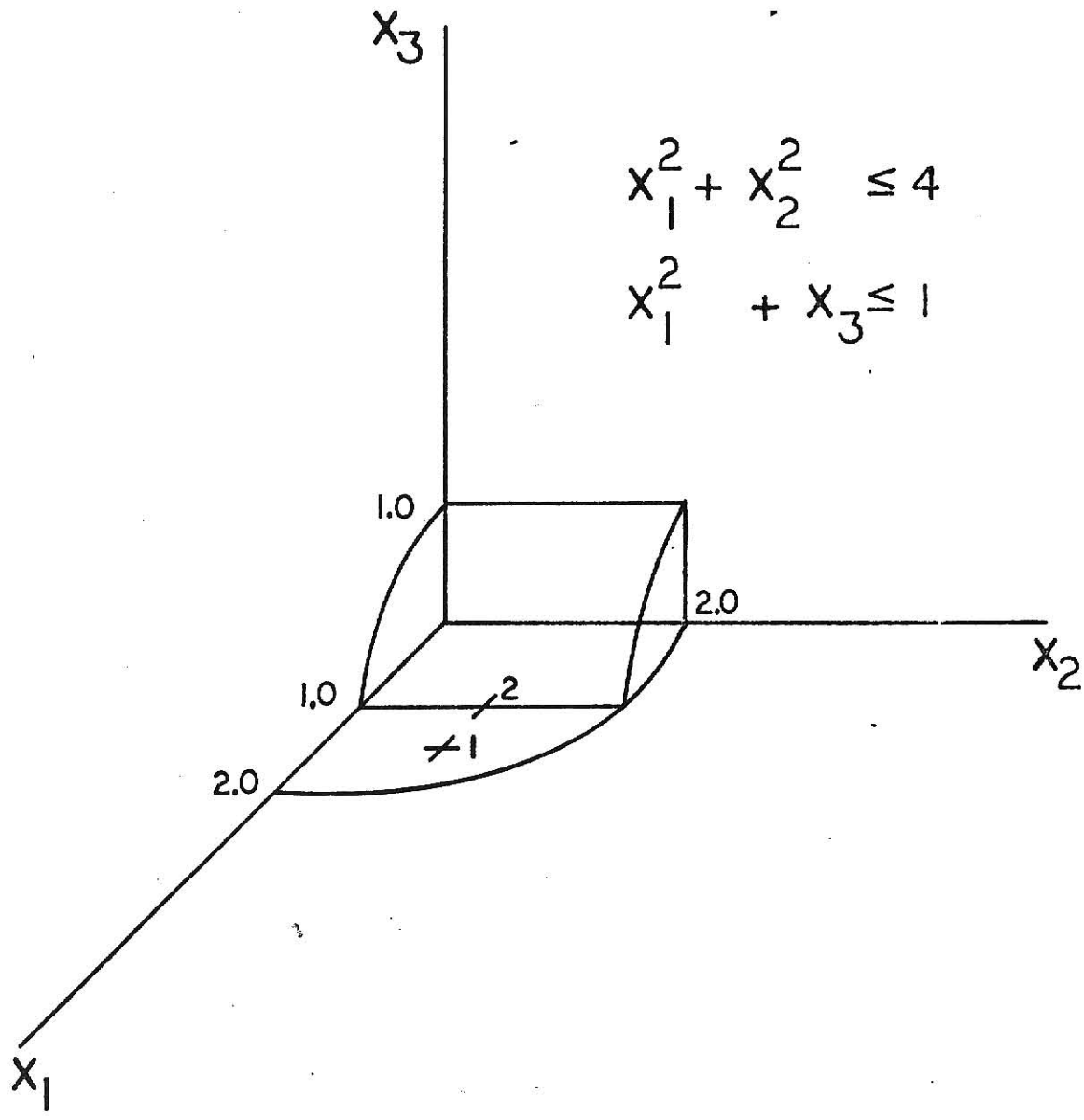


Fig. 3.4. Obtaining a feasible solution, point 2, by changing the basis in Example 4.

$$\bar{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 3/2 \\ 1 \\ 0 \end{pmatrix}$$

and, by solving for $\bar{y}$

$$\bar{y} = \begin{pmatrix} X_4 \\ X_5 \end{pmatrix} = \begin{pmatrix} 3/4 \\ -5/4 \end{pmatrix}$$

The variable X_5 violates a boundary condition and this must be corrected before proceeding. Let $X_5 = 0$ and make it an independent variable in place of X_1. This yields

$$\bar{x} = \begin{pmatrix} X_2 \\ X_3 \\ X_5 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Solving for $\bar{y}$ yields

$$\bar{y} = \begin{pmatrix} X_1 \\ X_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

a feasible solution (point 2 in Fig. 3.4). The new point becomes the next starting point only if the objective function is improved. This criterion depends on both the "shape" of the objective function and how "close" the point is to the point produced by the reduced gradient.

To prevent degenerate situations Abadie and Guigou [2,6] have recommended replacing a dependent variable, y_i, with an independent variable, x_j that maximizes the expression

$$\min \{|\alpha_{ij}|(b_j - x_j),\ |\alpha_{ij}|(x_j - a_j)\} \text{ , for } \forall j$$

$$\alpha_{ij} = \left(\frac{\partial \bar{f}}{\partial \bar{y}}\right)^{-1} \left(\frac{\partial \bar{f}}{\partial \bar{x}}\right)$$

The intention is to choose a value located far from its boundaries. This conjecture, though, has not been proven [2].

3.8 CONCLUDING REMARKS

The GRG algorithm is an attempt at the solution of the general continuous nonlinear programming problem. It is restricted to problems consisting of continuous constraints that have continuous first order partial derivatives. This restriction is a result of the Implicit Function Theorem cited eariler. The objective function is not as restricted in structure as the constraints. It is required to be continuous with the first order partial derivatives defined at each point. But, the first order partial derivatives do not necessarily have to be continuous if the objective function is monotonically increasing (or decreasing). This allows the use of a piecewise linear monotonically increasing (or decreasing) objective function.

The method is powerful because it possesses all of the convergence properties of the popular gradient techniques for unconstrained nonlinear optimization problems. Its main restriction is in step 4 of the algorithm. A method must exist for reentry into the feasible domain that converges rapidly, is reliable, and will control degenerate situations which are inherent in most problems. It is obvious that the re-entry procedure adds to the numerical difficulties that exist in large computer coded algorithms and that the GRG method is not easily coded and applied to large scale problems. These difficulties are largely technical, though, and can be alleviated with time and experience. The GRG method is theoretically a powerful technique and it appears that the only limitation to its application will be the computer code that represents it.

CHAPTER FOUR

INTRODUCTION TO THE GREG PROGRAM

4.1. INTRODUCTION

The generalized reduced gradient algorithm is coded in FORTRAN IV in the GREG program [30]. The program has been developed, by Abadie and his associates of Electricité de France. The GREG program consists of a main program, nine permanent or internal subroutines, and four user supplied, temporary or external subroutines. A typical source deck consists of approximately 2500 cards, requires approximately 150 K (K = 1024 bytes) of storage and 18 minutes of CPU time to compile by the IBM FORTRAN IV (G) compiler. For execution it requires, with the present dimensioning approximately 120 K of storage. The main program and the permanent subroutines have been compiled and stored in a partitioned data set. This simple step sharply reduces the CPU time, and consequently the cost. It also allows the program to be ran in less than 128 K of storage.

As presently dimensioned the GREG program will handle problems involving up to 50 inequality and/or equality constraints.

The maximum number of variables a problem may have depends on the type of constraints involved. The program automatically provides slack and artificial variables. There is one slack variable added for each inequality constraint. For a given constraint, if the starting point, $\bar{X}^0$, is not feasible, an artificial variable is supplied with an appropriate penalty attached to the objective function. Therefore, the dimension which represents the total number of variables a problem has depends on the following four numbers.

NV = the number of variables in the original problem.

NIN = the number of inequality constraints and, therefore, the number of slack variables added.

NIN1 = the number of inequality constraints not satisfied at the starting point and, therefore, the number of artificial variables added for this reason.

NIN4 = the number of equality constraints not satisfied at the starting point and, therefore, the number of artificial variables added for this reason.

The code is presently dimensioned with the following two constraints.

$$NV + NIN + NIN1 + NIN4 \leq 150$$

and

$$NV \leq 100$$

Use of the GREG program is approached in four steps. The first is developing the four user supplied, temporary subroutines. These subroutines, basically, define the problem. Next is organizing the input data. This step is extremely important when using the GREG program. The program is extremely sensitive to certain parameter values and they must be choosen carefully. The third step involves applying the necessary job control language. Along with the access procedures, the job control language is given that was used to load the program into the system. The title headings are in French and shall be translated as a finally.

4.2. THE GREG USER SUPPLIED SUBROUTINES

The user supplied subroutines must define the optimization problem in the form of maximize

$$f_0(\bar{X}), \quad \bar{X} = (X_j \mid j = 1, \ldots, NV)$$

subject to the constraints

$$f_i(\bar{X}) \leq \text{ or } = 0, \quad i = 1, \ldots, NC$$

$$a_j \leq X_j \leq b_j, \quad j = 1, \ldots, NV$$

Any nonlinear programming problem may be put into this form. The four subroutines that describe this problem set-up to the GREG program are PHIX,

CPHI, JACOB, and GRADFI. PHIX defines the objective function, CPHI the constraint function, JACOB the gradients of the constraint functions, and GRADFI the gradient of the objective function. Each of the four subroutines performs a unique task. They are referred to many times during the execution of the program and warrant careful programming considerations.

Each of the user supplied subroutines must contain a set of "COMMON" statements that are commensurate to the internal GREG subroutines. Figure 4.1 shows a list of the GREG common block definition statements. An error in this portion of the programming can result in completely erroneous results.

The subroutines are called in the following order.

1. PHIX
2. CPHI
3. JACOB
4. GRADFI

With the variable, IT, which is the iteration counter, the subroutines can be used to initialize values used within them by "READ", "DATA", or arithmetic statements. When the subroutines are called for the first time, IT = -1. By checking the condition, if IT < 0, the initializations are performed on the first call to the subroutine. The programming for the following problem is shown in Figs. 4.2 through 4.5. The problem shall serve as an example in the description of the four external, user supplied subroutines.

Maximize

$$f_0(\bar{X}) = \sum_{j=1}^{5} \ell n\ (1 - \exp(B_j \cdot X_j))$$

subject to the constraints

```
      DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
     1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
      COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
      COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NT0,NIN1,NIN2,NIN3,NIGEGA  50
     1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
     2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
      COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
     1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                               GEGA  90
      COMMON     IDIREC,DELTFI,ETA,JK0,LC,YSORT                          GEGA 100
```

Fig. 4.1. GREG common block definition statements.

$$f_1(\bar{X}) = \sum_{j=1}^{5} CC_{1j} \cdot X_j^2 \qquad \leq 110$$

$$f_2(\bar{X}) = \sum_{j=1}^{5} CC_{2j} \, (X_j + \exp\,(X_j/4)) \leq 175$$

$$f_3(\bar{X}) = \sum_{j=1}^{5} CC_{3j} \cdot X_j \exp\,(X_j/4) \quad \leq 200$$

PHIX

The external, user-supplied subroutine PHIX defines the objective function to the GREG program. This value is stored in the FORTRAN variable PHI, and is described in terms of the FORTRAN vector array, XC(J). Only the original problem variables are used. That is, J ranges from one to NV. The penalties due to the artifical variables are added to PHI automatically in an internal subroutine.

If it is necessary to initialize constants used in defining PHI, the variable IT may be used as a logical indicator so constant values are initialized only once. Since PHIX is the first subroutine called, it may also be used to initialize constants used in CPHI, JACOB, and GRADFI as long as they have been declared in a common block definition statement. Figure 4.2 shows an example of PHIX.

CPHI

CPHI defines the constraint functions as previously defined ($\leq$ or = 0). The values are stored in the vector array VC(I), I = 1, ..., NC, and in terms of the original problem variables, XC(J), J = 1, ..., NV. The constraints must be ordered with inequalities first and equalities second. Let,

```
C001            SUBROUTINE PHIX
C002            DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)   GEGA  20
               1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)          GEGA  30
C003            COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB         GEGA  40
C004            COMMON     NV,NC,NK,NEC,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
               1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICOB,JCOB,KCGEGA  60
               2CB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
C005            COMMON     KFONC,KGRAC,KCONT,KINV1,KINV2,KCOBA,KJACO,KMAX1,KMAX2,KGEGA  80
               1REN1,KREN2,KINV,KCOBA1,KREN11,KREN21                              GEGA  90
C006            COMMON     ICIREC,DELTFI,ETA,JKO,LC,YSORT                         GEGA 100
C007            COMMON B(5),CC(3,5)
C008            IF(IT)100,101,101
C009       100  B(1)=ALOG(0.2)
C010            B(2)=ALOG(0.15)
C011            B(3)=ALOG(0.1)
C012            B(4)=ALOG(0.35)
C013            B(5)=ALOG(0.25)
C014       101  PHI=0.0
C015            DO102I=1,5
C016       102  PHI=PHI+ALOG(1.0-EXP(B(I)*XC(I)))
C017            RETURN
C018            END
```

Fig. 4.2. An example of PHIX.

NIN = the number of inequality constraints,

NEG = the number of equality constraints.

Then, VC(I), I = 1, ..., NIN, are inequality constraints and VC(I), I = NIN+1, ..., NIN+NEG, are equality constraints. The total number of constraints is

NC = NIN + NEG

Like PHIX, constants may be initialized in CPHI by using IT as a logical indicator. Since CPHI is the second procedure call by GREG, it may be used to initalize values in JACOB and GRADFI. An example of CPHI is shown in Fig. 4.3.

JACOB

The subroutine JACOB defines the gradients of the constraint functions. The partial derivative $\frac{\partial f_i}{\partial x_j}$ is stored in the matrix array A(i,j). The rows of the matrix represent each constraint function, $f_i(\bar{X})$, i = 1, ..., NC, in the same order as sequenced in CPHI. The partial derivatives are represented in terms of the FORTRAN variable XC(j), j = 1, ..., NV.

Constant values may also be initialized in JACOB. It is the third subroutine referred to and may be used to initialize values in GRADFI. An example of JACOB is shown in Fig. 4.4.

GRADFI

The fourth and final user-supplied subroutine is GRADFI. This subroutine defines the gradient of the objective function in terms of the array XC(J), J = 1, ..., NV. The component values are stored in the vector array C(J), J = 1, ..., NV.

Like the other subroutines, initialization may be accomplished in GRADFI, but only for this subroutine since it is the last one called for initialization purposes. Fig. 4.5 shows an example of GRADFI.

```
C001              SUBROUTINE CPHI
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KO,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFCNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                        GEGA 100
C007              COMMON B(5),CC(3,5)
C008              IF(IT)100,101,101
C009         100  CC(1,1)=1.0
C010              CC(2,1)=7.0
C011              CC(3,1)=7.0
C012              CC(1,2)=2.0
C013              CC(2,2)=7.0
C014              CC(3,2)=8.0
C015              CC(1,3)=3.0
C016              CC(2,3)=5.0
C017              CC(3,3)=8.0
C018              CC(1,4)=4.0
C019              CC(2,4)=9.0
C020              CC(3,4)=6.0
C021              CC(1,5)=2.0
C022              CC(2,5)=4.0
C023              CC(3,5)=9.0
C024         101  VC(1)=0.0
C025              DO102J=1,5
C026         102  VC(1)=VC(1)+CC(1,J)*XC(J)**2
C027              VC(1)=VC(1)-110.0
C028              VC(2)=0.0
C029              DO103J=1,5
C030         103  VC(2)=VC(2)+CC(2,J)*(XC(J)+EXP(XC(J)/4.0))
C031              VC(2)=VC(2)-175.0
C032              VC(3)=0.0
C033              DO104J=1,5
C034         104  VC(3)=VC(3)+CC(3,J)*XC(J)*EXP(XC(J)/4.0)
C035              VC(3)=VC(3)-200.0
C036              RETURN
C037              END
```

Fig. 4.3. An example of CPHI.

```
C001              SUBROUTINE JACOB
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFCNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     ICIREC,DELTFI,ETA,JKO,LC,YSORT                        GEGA 100
C007              COMMON B(5),CC(3,5)
C008              DO101J=1,5
C009          101 A(1,J)=2.0*CC(1,J)*XC(J)
C010              DO102J=1,5
C011          102 A(2,J)=CC(2,J)*(1.0+0.25*EXP(XC(J)/4.0))
C012              DO103J=1,5
C013          103 A(3,J)=CC(3,J)*EXP(XC(J)/4.0)*(1.0+0.25*XC(J))
C014              RETURN
C015              END
```

Fig. 4.4. An example of JACOB.

```
C001              SUBROUTINE GRADFI
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004             1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  50
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  60
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  70
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  80
C006              COMMON     ICTREC,DELTFI,ETA,JKO,LC,YSORT                        GEGA  90
C007              COMMON B(5),CC(3,5)                                              GEGA 100
C008              DO101J=1,5
C009              AA=EXP(B(J)*XC(J))
C010          101 C(J)=B(J)*AA/(AA-1.0)
C011              RETURN
C012              END
```

Fig. 4.5. An example of GRADFI.

Since the constants used in PHIX will be related to those in GRADFI, and likewise for CPHI and JACOB, the initializations may usually be accomplished in PHIX and CPHI. To do this the constant variables are defined in "COMMON" statements so their values may be passed, but be cautious when doing this.

Note that the arrays A(I,J) and C(J) in the subroutines JACOB and GRADFI are initialized automatically at zero, thus eliminating the need for defining any zero values for them.

4.3. INPUT DATA FOR THE GREG PROGRAM

To use the GREG program, values for nineteen parameters, a starting point, a lower bound, and an upper bound must be established. The parameter input has been programmed in such a way that a minimum of two may be read in as input (NV and NIN or NEG) while the remaining ones take on default values. Besides this, the user may have designed some input for initialization purposes in PHIX, CPHI, JACOB, and GRADFI. It is the purpose of this section to discuss the above three types of input data.

The list of parameters and their definitions are given in Table 4.1. Each parameter is given a default value which is used if it is not changed in the parameter input list. Note that if NV is not given a value greater than zero the program will terminate without notice. The parameter input follows the rules of the IBM FORTRAN "NAMELIST" input procedure. This type of input is exemplified in Fig. 4.6. The GREG program is extremely sensitive to the values given to the parameters. It must be emphasized that the program is written in single precision arithmetic. This allows only seven significant digits and stopping criteria smaller than 1.0×10^{-7} can yield erroneous results.

Table 4.1. Parameters

	FORTRAN Program Symbols	Explanations	Default Value
1.	NV	the number of original problem variables.	0
2.	NIN	the number of inequality constraints.	0
3.	NEG	the number of equality constraints.	0
4.	NEVL	the maximum number of iterations for Newton's Method.	20*
5.	NTØ (NT zero)	the maximum number of bisections in the parabolic interpolation process when maximizing a function of a single variable.	6*
6.	ITET	the number of previous iterations used to help determine a maximum value for θ.	20*
7.	ICONJ	equals 1 if conjugate directions are desired when maximizing, otherwise zero.	1
8.	IDIAG	equals 1 if diagonal directions are desired when maximizing, otherwise zero.	0
9.	ITMAX	the maximum number of iterations.	50
10.	KFIL	equals 1 if the cost function is linear, otherwise, zero.	0
11.	KLIN	equals 1 if all of the constraints are linear, otherwise, zero.	0
12.	NCØ (NC zero)	equals zero for linear programming problems and is $\geq$ the number of constraints for nonlinear programming problems.	10
13.	ITSOR	the iteration which recording of the intermediate output starts. ($1 \leq$ ITSOR $\leq$ ITMAX)	1*

Table 4.1. (continued)

14.	ISOLSR	the solution is printed out every ISOLSR iterations. For small problems, ISOLSR = ITMAX. For large problems, ISOLSR < ITMAX.	50
15.	EPSIL	used as a criterion for the choice of a pivot in the changing and inversion of a basis. ($10^{-2} \leq$ EPSIL $\leq 10^{-1}$)	0.1E0*
16.	EPSILØ (EPSIL zero)	is used as a stopping criteria for Newton's Method. (EPSILØ $\geq 10^{-7}$)	0.1E-02
17.	EPSIL1	is used as a stopping criteria if the problem is declared convex. (EPSIL1 $\geq 10^{-7}$)	0.1E-02
18.	EPSIL2	is used as a stopping criterion by using the norm of the reduced gradient. (EPSIL2 $\geq 10^{-7}$)	0.1E-02
19.	PC	equals zero if the problem is non-convex; equals one if the problem is convex. This parameter affects only the stopping criteria.	0.0

* - it is recommended that these values are not changed.

```
                C
                O
                L
                U
                M
                N
                123456789
               ┌
first card     | &PARM NV=10,NIN=3,NEG=2,PC=1.0,
second card    | EPSIL0=0.1E-4,EPSIL2=0.1E-03,&END
```

Fig. 4.6. An example of a FORTRAN namelist input for the GREG program. The title of the namelist is PARM.

After the parameter data, the starting point, the lower bound, and the upper bound are read in that order. The format is 8G10.4 which is equivalent to 8F10.4 and 8E10.4. This allows eight values per card. If a card is filled, skip to the next card and repeat the process until all NV values are listed. Each vector, the starting point, the lower bound, and the upper bound will start on a new card. The last data to enter the program is the optional, user defined initialization input. The data has to be ordered in the same calling sequence of the external subroutines PHIX, CPHI, JACOB, and GRADFI. Figure 4.7 shows the sequence of the input data.

4.4. JOB CONTROL LANGUAGE FOR UTILIZING THE GREG PROGRAM

The job control language given in this section is all that is necessary to utilize the GREG program at Kansas State University. Details can be found in the IBM FORTRAN IV (G and H) Programmers Guide. One can also consult the Kansas State University Computing Center User's Guide [12] for details on their procedures for preserving permanent data-sets.

There are two types of procedures of interest to the GREG user. They are the load and the access procedures. The two load procedures that have been used to install GREG into the system are given in Figs. 4.8, 4.9, and 4.10. Figures 4.11, 4.12, and 4.13 show the three access procedures for running a particular problem. It is recommended that object decks be used when several runs for the same problem are made. This will substantially reduce CPU time, and consequently, the cost.

4.5 INTERPRETING THE GREG OUTPUT

Output headings for the GREG program are in French. This section translates the headings and interprets the output. The language feature was not converted because of excessive costs to recompile the program.

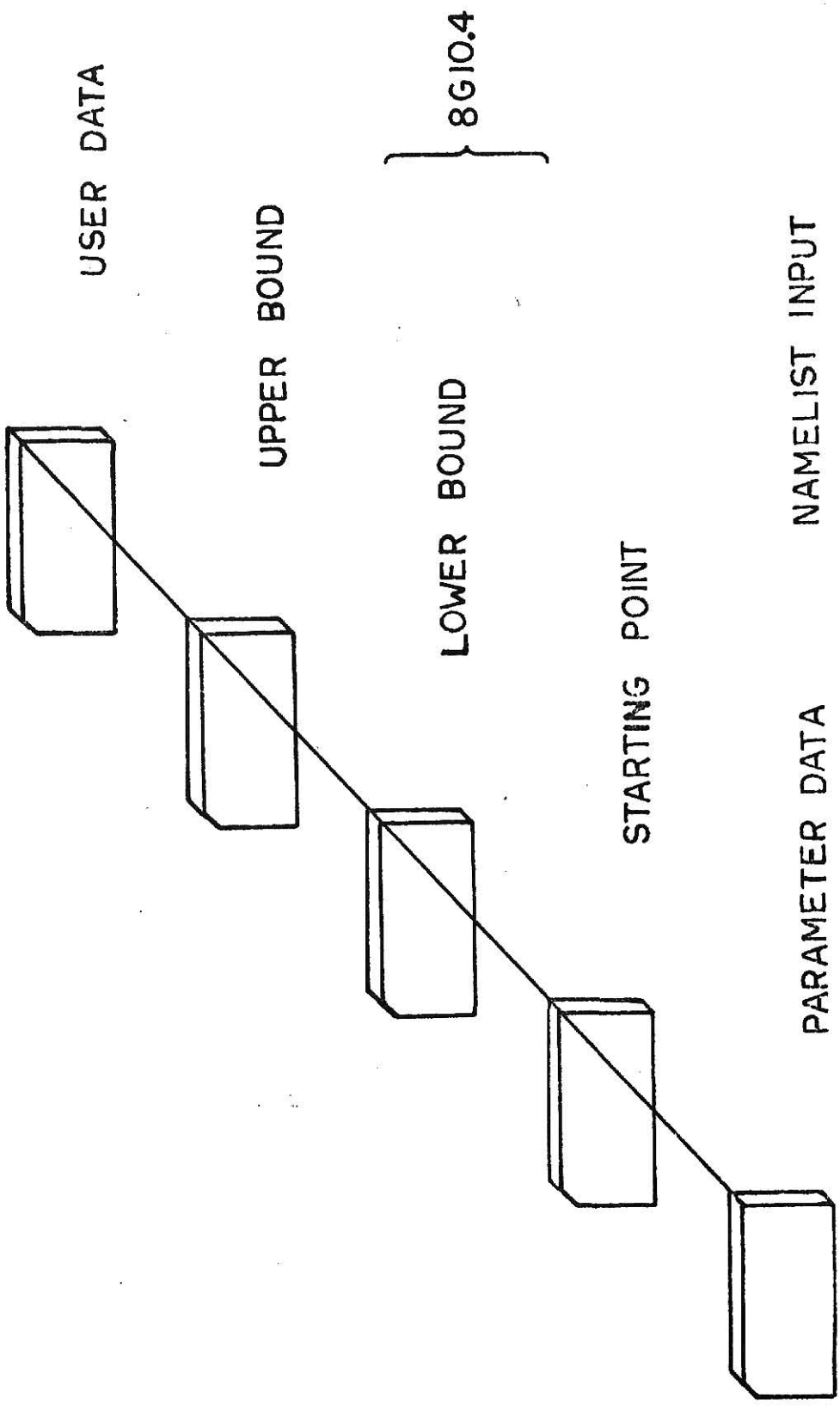


Fig. 4.7. The sequence of the input data deck.

```
//          JOB ( standard K.S.U. job card )
//        EXEC FORTGCLG
//FORT.SYSIN DD *

           complete GREG source deck including the subroutines
           PHIX, CPHI, JACOB, and GRADFI.

/*
//LKED.SYSLMOD DD DSN=yyyyyy.xxxxxxxx(name),DISP=(NEW,KEEP),UNIT=2314
//       VOL=SER=nnnnnn,SPACE=(TRK,(10,5,3),RLSE)
//GO.SYSIN DD *

           data

/*
```

Fig. 4.8. The necessary job control language for loading the GREG source deck into a partitioned data-set.

```
//          JOB ( standard K.S.U. job card )
//        EXEC FORTGLG
//LKED.SYSLMOD DD DSN=yyyyyy.xxxxxxxx(name),DISP=(NEW,KEEP),UNIT=2314
//        VOL=SER=nnnnnn,SPACE=(TRK,(10,5,3),RLSE)
//LKED.SYSIN DD *

            complete GREG object deck including the subroutines
            PHIX, CPHI, JACOB, and GRADFI.

/*
//GO.SYSIN DD *

            data

/*
```

Fig. 4.9. The necessary job control language for loading the GREG object deck into a partitioned data-set.

```
//          JOB ( standard K.S.U. job card )
//        EXEC FORTGCLG
//FORT.SYSIN DD *

            source deck

/*
//LKED.SYSLMOD DD DSN=yyyyyy.xxxxxxxx(name),DISP=(NEW,KEEP),UNIT=2314
//        VOL=SER=nnnnnn,SPACE=(TRK,(10,5,3),RLSE)
//LKED.SYSIN DD *

            object deck

/*
//GO.SYSIN DD *

            data

/*
```

Fig. 4.10. The necessary job control language for loading a combination of source and object decks into a partitioned data-set.

```
//          JOB ( standard K.S.U. job card )
//        EXEC FORTGCLG
//FORT.SYSIN DD *

            source deck for PHIX, CPHI, JACOB, and GRADFI.

/*
//LKED.LIB DD DSN=yyyyyy.xxxxxxxx,DISP=SHR,UNIT=2314,VOL=SER=nnnnnn
//LKED.SYSIN DD *
          INCLUDE LIB(name)
          ENTRY MAIN
/*
//GO.SYSIN DD *

            data

/*
```

Fig. 4.11. Access to the stored program using a source deck for PHIX, CPHI, JACOB, and GRADFI.

```
//          JOB ( standard K.S.U. job card )
//        EXEC FORTGLG
//LKED.LIB DD DSN=yyyyyy.xxxxxxxx,DISP=SHR,UNIT=2314,VOL=SER=nnnnnn
//LKED.SYSIN DD *

            object deck for PHIX, CPHI, JACOB, and GRADFI.

          INCLUDE LIB(name)
          ENTRY MAIN
/*
//GO.SYSIN DD *

            data

/*
```

Fig. 4.12. Access to the stored program via an object deck.

```
//          JOB ( standard K.S.U. job card )
//        EXEC FORTGCLG
//FORT.SYSIN DD *

            source deck

/*
//LKED.LIB DD DSN=yyyyyy.xxxxxxxx,DISP=SHR,UNIT=2314,VOL=SER=nnnnnn
//LKED.SYSIN DD *

            object deck

          INCLUDE LIB(name)
          ENTRY MAIN
/*
//GO.SYSIN DD *

            data

/*
```

Fig. 4.13. Access to the stored program via an object and a source deck for PHIX, CPHI, JACOB, and GRADFI.

Another feature of the output is that all real numbers are printed with fourteen significant digits because the program was developed on a machine with fifteen digit accuracy. Only the first seven digits are valid when using the IBM 360/50 computer. This feature was not changed because of the cost and plans to initiate a double precision version to handle the numerical difficulties inherent in some problems.

The output consists of five sections. They are

1. parameter echo check
2. starting conditions
3. intermediate output
4. final solution
5. summary of the results.

Table 4.2 translates the main French titles.

Of chief concern are the stopping conditions (NIVEAU DE SORTIE). Numerically, the stopping condition codes are from zero to eight. The descriptions follow.

0. The program is terminated when the total number of iterations is greater than or equal to a specified maximum number of iterations. The number of iterations, IT, will exceed the maximum only during the course of a cycle of conjugate directions.
1. The program is terminated when the norm of the projected reduced gradient is less than or equal to ε_2 (EPSIL2) times the norm of the projected reduced gradient of the first iteration that has no artificial variables.

2. Let

$$n = \max.\ n_1, \forall i$$

$$n_i = |p_i\ (b_i - x_i)| \text{ if } p_i > 0$$

$$n_i = |p_i\ (a_i - x_i)| \text{ if } p_i < 0$$

b_i = the upper bound for an independent variable x_i.

a_i = the lower bound for an independent variable x_i.

The program is terminated when

$$n < \varepsilon_3,\ \varepsilon_3 = \frac{\varepsilon_1}{n},\ \varepsilon_1 = \text{EPSIL1}$$

3. If the problem has been declared convex, then the program is terminated when

$$\Delta\phi < \varepsilon_1,\quad \varepsilon_1 = \text{EPSIL1}$$

$$\Delta\phi = \Sigma n_i,\ \forall i$$

4. Let $\bar{c}^T = \frac{\partial f_0}{\partial \bar{X}}$, the gradient of the objective function and $\bar{Y}^T$ = the direction of movement. This termination occurs when $c \cdot Y < 0$ (scalar product).

If $\overline{Y}$ is a conjugate direction the iteration is repeated and $\overline{Y}$ is set equal to $\overline{C}$. If the condition occurs again an inversion followed by a search for a new basis is performed. The program is terminated if $\overline{C}.\overline{Y} < 0$ occurs another time.

5. If during the course of a re-entry into the feasible region the Psuedo-Newton method yields a point satisfying the constraints, but does not improve the function value, then the re-entry point is modified by choosing a θ^* such that

$$\theta^* = \max\ (\theta/10,\ \theta_{int})$$

θ_{int} is an interpolated value between (zero and θ)

This condition is internal and is not printed. It is normally allowed to occur twice in succession after which an inversion with a new basis

is initiated. If this fails to correct the condition then the stopping condition 7 is enacted.

6. The program is terminated when

$$|f_o(\bar{x}^\circ + \theta\bar{h}^\circ, \bar{y}^\circ + \theta\bar{k}^\circ) - f_o(\bar{x}^\circ, \bar{y}^\circ)| < \varepsilon^* |f_o(\bar{x}^\circ, \bar{y}^\circ)|$$

This stopping condition occurs only if the basis consists of slack variables.

7. The program is terminated when

$$|f_o(\bar{x}^\circ + \theta\bar{h}^\circ, \bar{y}^\circ + \theta\bar{k}^\circ) - f_o(\bar{x}^\circ, \bar{y}^\circ)| < \varepsilon^* |f_o(\bar{x}^\circ, \bar{y}^\circ)|$$

but the basis does not consist of just slack variables as in condition 6. Before stopping, an inverse with a new basis is initiated after which the direction of movement is conjugate to the gradient. If this does not work then the direction is along the projected reduced gradient, after which, the program terminates. This condition is usually implemented by successive reductions of θ due to failure to re-enter the feasible region or improve the objective function.

8. The program terminates when

$$|f_o(\bar{x}^\circ + \theta\bar{h}^\circ, \bar{y}^\circ + \theta\bar{k}^\circ) - f_o(\bar{x}^\circ, \bar{y}^\circ)| < \varepsilon^* |f_o(\bar{x}^\circ, \bar{y}^\circ)|$$

Unlike condition 7, no attempt to re-enter the feasible region has failed. An attempt to override the stopping condition is effected by selecting and inverting a new basis and taking the directions of movement as the projected reduced gradient.

The desired stopping conditions are one, two, and three. Conditions six, seven, and eight indicate that precision difficulties may exist. Attempting an improvement from these stopping conditions may be desired since their presence indicates premature stopping. There are two methods.

1. Try a new starting point since the route from the present starting point must be very "flat".

2. Subtract a constant from the objective function so $\varepsilon^* \left| f(\bar{x}^\circ, \bar{y}^\circ) \right|$ will be smaller.

These methods are not guaranteed to work, but they have helped solve some problems (such as the Colville problem number 3 worked in Chapter 5). If the difficulty persists then the only alleviation is more precision.

$\varepsilon^* = 1.0 \times 10^{-12}$ for GREG.

$\varepsilon^* = 1.0 \times 10^{-6}$ for GREGR (a more realistic value for seven digit precition).

```
PARAMETRES   (parameters)

NV          5
NIN         6
NEG         0
NEVL       20
NTO         6
ITET       20
ICONJ       1
ICIAG       0
ITMAX      50
KFIL        0
KLIN        0
NCO        10
ITSCR       1
ISOLSR     10
EPSIL      0.1E 00
EPSILO     0.1E-04
EPSIL1     0.1E-02
EPSIL2     0.1E-02
PC         0.0
```

Table 4.2. Translation of the five output sections of the GREG program.

GRADIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES 5 (number of primal variables)
NOMBRE TOTAL DE VARIABLES 11 (total number of variables)
NOMBRE DE CONTRAINTES 6 (number of constraints)

EPSILON DE NEWTON 0.1000E-04 (stopping condition for Newton's method)
EPSILON TEST GRADIENT 0.1000E-02 (stopping condition for gradient test)

FONCTION ECONOMIQUE 0.1171875000000E-01
(objective function)

	BORNE INFERIEURE (lower bound)	VARIABLE NATURELLE (starting point)	BORNE SUPERIEURE (upper bound)
XI(1)	0.7800000000000E 02	X(1) 0.786199951171888E 02	XS(1) 0.1020000000000E 03
XI(2)	0.3300000000000E 02	X(2) 0.334399871826176 02	XS(2) 0.4500000000000E 02
XI(3)	0.2700000000000E 02	X(3) 0.310699920654306 02	XS(3) 0.4500000000000E 02
XI(4)	0.2700000000000E 02	X(4) 0.440999908447276 02	XS(4) 0.4500000000000E 02
XI(5)	0.2700000000000E 02	X(5) 0.352199859619146 02	XS(5) 0.4500000000000E 02

VALEUR DES CONTRAINTES
(constraint values)

C(1)	-0.211364746093756 00
C(2)	-0.917886352539066 02
C(3)	-0.111057281494146 02
C(4)	-0.889427185058596 01
C(5)	-0.487271118164066 01
C(6)	-0.127288818359386 00

Table 4.2. (continued)

```
IT  1   PHI 0.11817578125CC0E 03   DIR.GRAC.  NO  2   YN 0.35E 03   DELTAFI 0.19E 04   ETA 0.14E 04   NCDB  0   NCN  2   NITN  2
IT  2   PHI 0.11823437500CC0E 03   DIR.CONJ.  LC  3                                                       NCDB  0   NCN  2   NITN  2
IT  3   PHI 0.11828906250000E 03   DIR.CONJ.  LC  3                                                       NCDB  0   NCN  2   NITN  2
IT  4   PHI 0.25400390625000E 03   DIR.GRAC.  NO  2   YN 0.95E 03   DELTAFI 0.57E 03   ETA 0.70E 03   NCDB  2   NCN  8   NITN 20
IT  5   PHI 0.25632C31250C00E 03   DIR.GRAC.  NC  3   YN 0.83E 03   DELTAFI 0.38E 02   ETA 0.38E 02   NCDB  0   NCN  2   NITN  7
IT  6   PHI 0.29201171875000E 03   DIR.GRAC.  NO  4   YN 0.83E 03   DELTAFI 0.36E 02   ETA 0.36E 02   NCDB  0   NCN  3   NITN 12
        PHI 0.29201171875000E 03                      YN 0.81E 03   DELTAFI 0.43E-02   ETA 0.43E-02

CUREE  CU  CALCUL    165  CENTISECONDES
 (execution time)
```

NO = number of variables at their lower or upper bounds.
LC = number of conjugate cycles when using conjugate directions.
YN = norm of the reduced gradient.
DELTAFI = $\Delta\Phi$ as defined by stopping condition 3.
ETA = n as defined by stopping condition 2.
NCDB = number of changes of basis during the iteration.
NCN = number of calls to the Newton method.
NITN = number of iterations of the Newton method during the course of NCN calls.

Table 4.2. (continued)

VARIABLE NATURELLE
(primal variable)

```
X(  1) =  0.7800000000000E 02 = XI(  1)
X(  2) =  0.3300000000000E 02 = XI(  2)
X(  3) =  0.2999389644E4375E 02
X(  4) =  0.4500000000000E 02 = XS(  4)
X(  5) =  0.3677612304687E 02
```

VARIABLE DUALE ASSOCIEE
(associated reduced gradient value)

```
V(  1) = -0.489176023390638 02
V(  2) = -0.843099212646488 02
V(  3) =  0.
V(  4) =  0.266324462890638 02
V(  5) =  0.
```

CONTRAINTE
(constraint)

```
CONTRAINTE  1      -0.152587890625000E-04
CONTRAINTE  2      -0.919999847412118 02
CONTRAINTE  3      -0.111582489013678 02
CONTRAINTE  4      -0.684175109863788 01
CONTRAINTE  5      -0.499998474121098 01
CONTRAINTE  6      -0.152587890625000E-04
```

VARIABLE DUALE ASSOCIEE
(Lagrange multiplier)

```
U(  1) =  0.403186523437508 03
U(  2) =  0.0
U(  3) =  0.0
U(  4) =  0.0
U(  5) =  0.0
U(  6) =  0.809375732421888 03
```

Table 4.2. (continued)

```
NIVEAU DE SORTIE      7  (stopping condition)

NOMBRE D ITERATIONS       6  (number of iterations)

NOMBRE D APPELS DU S/P CALCULANT LES CONTRAINTES     93  (number of times the constraints were calculated)

NOMBRE D APPELS DU S/P CALCULANT LA FONCTION ECONOMIQUE     69  (number of times the objective function was calculated)

NOMBRE D APPELS DU S/P CALCULANT LE GRADIENT DE LA FONCTION ECONOMIQUE     14  (number of times the gradient was calculated)

NOMBRE D APPELS DU S/P CALCULANT L INVERSE AVEC BASE DONNEE      4  (number of inverse calculations with the given base)
NOMBRE D APPELS DU S/P CALCULANT L INVERSE AVEC RECHERCHE DE BASE      1  (number of inverse calculations with another base)

NOMBRE D APPELS DU S/P EFFECTUANT LES CHANGEMENTS DE BASE      3  (number of base changes)

NOMBRE D APPELS DU S/P CALCULANT LE JACOBIEN DES CONTRAINTES      8  (number of times the Jacobien of the constraints was calculated)

NOMBRE D APPELS DU S/P EFFECTUANT UNE RECHERCHE DE MAX SANS DICHOTOMIE     14  (number of maximums found without bisection)
NOMBRE D APPELS DU S/P EFFECTUANT UNE RECHERCHE DE MAX AVEC DICHOTOMIE      0  (number of maximums found by bisection)

NOMBRE D APPELS DU S/P EFFECTUANT LA RENTREE DANS LE DOMAINE     26
  (number of reentries into the feasible region)
                                                    NOMBRE TOTAL D ITERATIONS     74
                                                    (total number of Newton's iterations)
```

Table 4.2. (end)

CHAPTER FIVE

NUMERICAL TESTING OF THE GREG PROGRAM

5.1 INTRODUCTION

Because this study was the first implementation of the GREG code at Kansas State University it was necessary to test the program and discover any limitations it might have. For this reason a series of numerical comparisons were performed. The problems involved were of three categories. There were simple example problems, small scale problems from recent publications that were solved by other techniques, and Colville test problems. The Colville study [10] involved both GRG 66 and GRG 69, which compared favorably with other existing nonlinear programming codes. The GREG version was derived from both of these codes and should be superior in all aspects.

The GREG code was developed on the CDC 6600 computer [30] which has fifteen digit single precision arithmetic in its FORTRAN compiler. It must be emphasized that this is a distinct advantage over the IBM system 360/50 with seven digit single precision arithmetic. This means that the code, which is very sensitive to numerical precision, can behave quite differently in the IBM system 360/50 .

In performing the numerical study it was hoped that the reliability, limitations, accuracy, and speed of the program could be judged. All of the pertinent programming material is given in Appendix 4. The speed of the program is measured as the time of execution of the algorithm. It does not include compiling of the subroutines and initialization of the program. Initialization involves, basically, the adding of slack and artificial variables.

5.2 THE EXAMPLE PROBLEM OF CHAPTER THREE

The first problem worked was Example 3 of Chapter 3. The problem definition is maximize

$$f_0(\bar{X}) = (2X_1 - \frac{1}{2}X_2^2) + (3X_2 - \frac{1}{2}X_2^2)$$

subject to the constraint

$$f_1(\bar{X}) = X_1^2 + X_2^2 - 1 \leq 0$$

$$X_i \geq 0, \; i = 1,2,3$$

The iterative process in the GREG program was different from the example because it used the slack variable as the dependent variable in the starting basis. The solution was obtained in 0.5 seconds as

$$f_0 = 3.105550$$

$$X_1 = 0.5545919$$

$$X_2 = 0.8321228$$

5.3 RELIABILITY OF A COMPLEX SYSTEM

The following model was proposed by Tillman, Hwang, Fan, and Lai [35,44] in the study of the reliability of spacecraft life support systems. Two optimization problems were considered from it. The first was to maximize the system reliability subject to a weight constraint. The second was to minimize the system weight subject to a reliability constraint. The problems were optimized by the sequential unconstrained minimization technique [25]. The codes employed were the RAC program [38] and a code employing the Hooke and Jeeves pattern search [33,35]. The second code was superior to the RAC program yielding a computational time for both problems of 90.4 seconds [35].

A schematic diagram of the system is shown in Fig. 5.1 [35,44]. Each of the numbered components is assumed to have a reliability of R_i, $i = 1,2,3,4$. Assuming that component three does not fail the system unreliability is

$$Q_1 = R_3 \left((1-R_1)(1-R_4) \right)^2$$

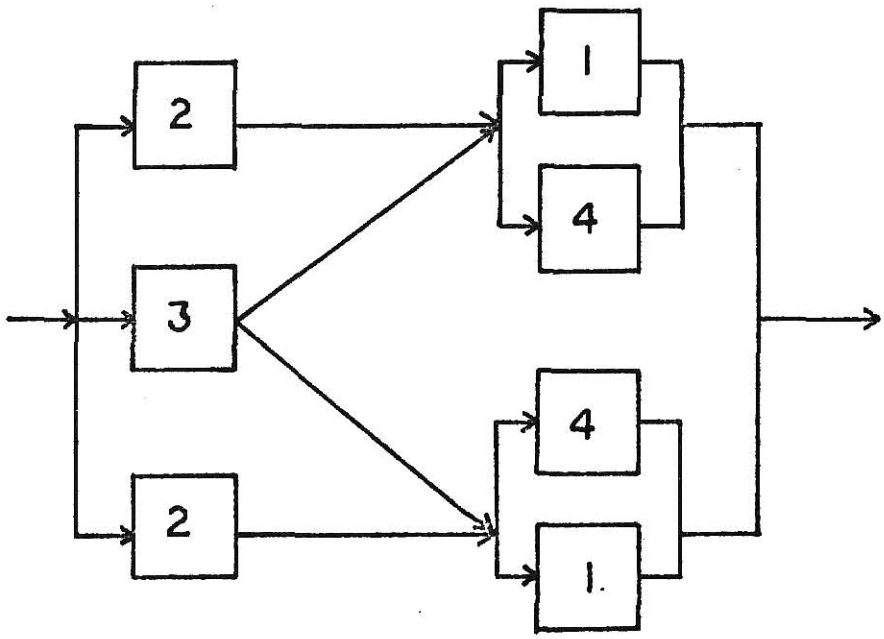


Fig. 5.1. Schematic diagram of a complex system.

If component three does fail then the probability of system failure becomes

$$Q_2 = (1 - R_3)\,(1 - R_2\,(1 - (1 - R_1)(1 - R_4))\,)^2$$

Since the total reliability of the system is the complement of the total unreliability, the reliability is

$$R_s = 1 - Q_1 - Q_2$$

The weight of the system was given as a function of the component reliability, R_i, $i = 1,2,3,4$.

$$\text{weight} = \sum_{i=1}^{4} K_i\, R_i^{\alpha_i}$$

The resultant optimization problems are

maximize

$$R_s = 1 - R_3(\,(1-R_1)(1-R_4)\,)^2 - (1-R_3)(1-R_2\,(1-(1-R_1)(1-R_4))\,)^2$$

subject to the constraint

$$K_1 R_1^{\alpha_1} + K_2 R_2^{\alpha_2} + K_3 R_3^{\alpha_3} + K_4 R_4^{\alpha_4} \leq C$$

and minimize

$$\text{weight} = K_1 R_1^{\alpha_1} + K_2 R_2^{\alpha_2} + K_3 R_3^{\alpha_3} + K_4 R_4^{\alpha_4}$$

subject to the constraints

$$1 - R_3(\,(1-R_1)(1-R_4)\,)^2 - (1-R_3)(1-R_2\,(1-(1-R_1)(1-R_4))\,)^2 \geq R_s,\ \min$$

$$R_i \geq R_{i,\,\min}\ ,\ i = 1,2,3,4$$

Numerical values for the parameters were given as

$K_1 = 100, \quad K_2 = 100, \quad K_3 = 200, \quad K_4 = 150$

$C = 800, \quad R_{s,min} = 0.9$

$\alpha_i = 0.6, \quad R_{i,min} = 0.5, \quad i = 1,2,3,4$

Numerical results for the GREG code were as follows.

First Problem

$R_s = 1.0$

$R_1 = 1.0$

$R_2 = 1.0$

$R_3 = 0.5394473$

$R_4 = 0.7974168$

Execution time = 0.35 seconds

Physically, the results may be interpreted as investing all effort into components one and two to make their reliability as high as possible. The best optimum solution by the previous methods was $R_s = 0.999997$.

Second Problem

weight = 641.8232

$R_1 = 0.5$

$R_2 = 0.8389202$

$R_3 = 0.5$

$R_4 = 0.5$

Execution time = 0.48 seconds

The best previous solution was

weight = 642.249

Total computational time for both problems was less than a second.

5.4 OPTIMUM RELIABILITY OF A MULTISTAGE PARALLEL SYSTEM

The problem in this section was optimized in Chapter Two using separable programming. It was obtained from a study by Tillman, Hwang, Fan, and Balbale [45]. It involves the optimization of a mixed system with N stages in series with components at each stage in parallel. The system is shown schematically in Fig. 5.2. The problem is to choose the number of parallel components at each stage that maximizes the systems reliability subject to a weight constraint. In reality, this problem should have an integer solution [43], but this point was neglected in the following solution since it is desired to use the example as a test problem only.

For a given stage, i, if R_i is the reliability of the components involved then

$$(1 - R_i)^{M^i}$$

is the unreliability of M^i components in parallel. The reliability is the complement of the unreliability and for N stages the total reliability is

$$R_s = \prod_{i=1}^{N} (1 - (1 - R_i)^{M^i})$$

A constraint involving weight and volume was given as

$$\sum_{i=1}^{N} p^i (M^i)^2 \leq P$$

The value of p^i is the product of weight per unit and volume per unit at stage i.

The cost constraint was given as

$$\sum_{i=1}^{N} c^i (M^i + \exp (M^i/4)) \leq C$$

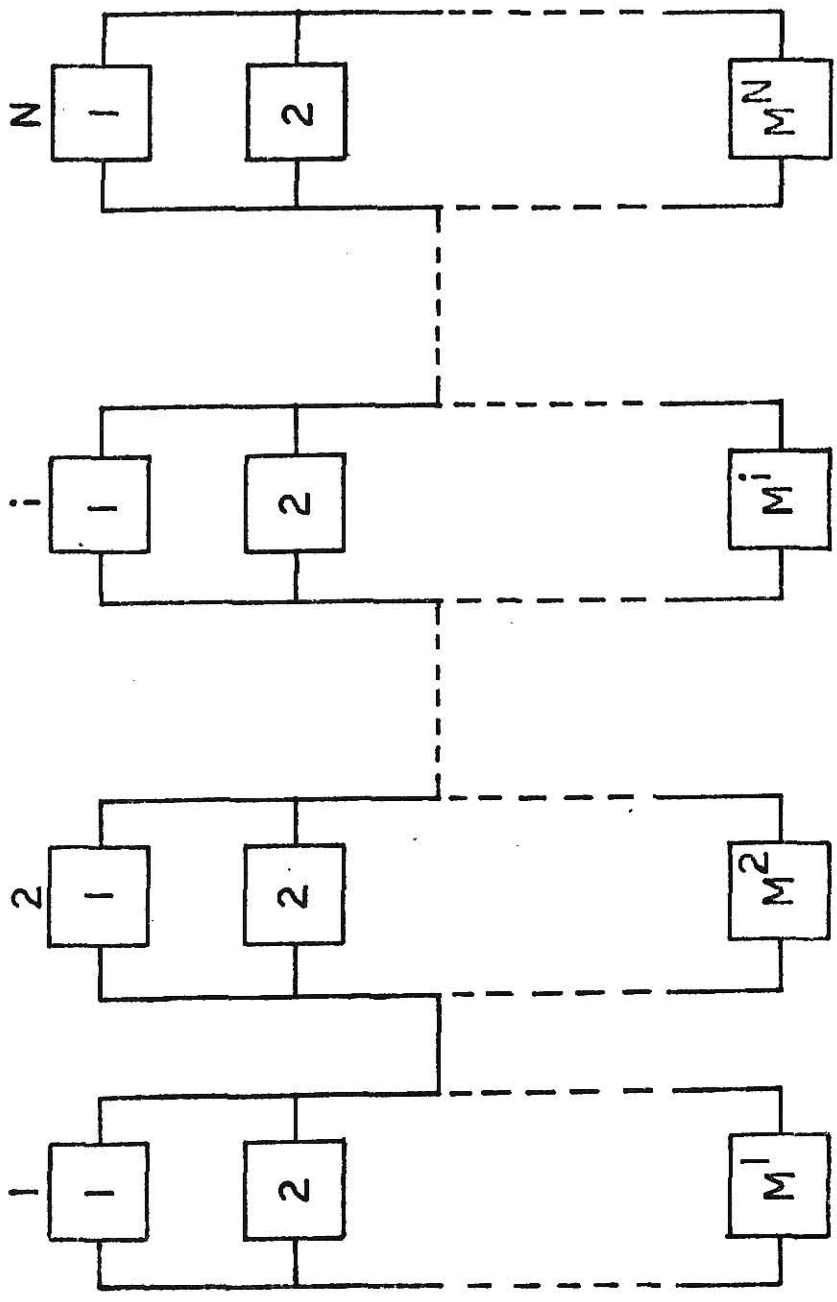


Fig. 5.2. Schematic diagram of a multistage parallel system.

The value $c^i M^i$ represents the cost of the units at stage i and $c^i \exp(M^i/4)$ is the additional cost for interconnecting parallel units.

The weight constraint was given as

$$\sum_{i=1}^{N} w^i M^i \exp(M^i/4) \leq W$$

The value $w^i M^i$ is the weight of the units at stage i and the factor $\exp(M^i/4)$ allows for the weight of interconnecting the units.

The problem was easier to work by maximizing $\ln(R_s)$. This makes the problem separable. The parameter values used were

$R_1 = 0.80, \quad p^1 = 1.0, \quad c^1 = 7.0, \quad w^1 = 7.0$

$R_2 = 0.85, \quad p^2 = 2.0 \quad c^2 = 7.0 \quad w^2 = 8.0$

$R_3 = 0.90, \quad p^3 = 3.0 \quad c^3 = 5.0 \quad w^3 = 8.0$

$R_4 = 0.65 \quad p^4 = 4.0 \quad c^4 = 9.0 \quad w^4 = 6.0$

$R_5 = 0.75 \quad p^5 = 2.0 \quad c^5 = 4.0 \quad w^5 = 9.0$

The problem is (changing M^i to X_i)

maximize

$$f_0 = \ln(1-.2^{X_1}) + \ln(1-.15^{X_2}) + \ln(1-.10^{X_3}) + \ln(1-.35^{X_4}) + \ln(1-.25^{X_5})$$

subject to the constraints

$$X_1^2 + 2X_2^2 + 3X_3^2 + 4X_4^2 + 2X_5^2 \leq 110$$

$$7(X_1 + e^{X_1/4}) + 7(X_2 + e^{X_2/4}) + 5(X_3 + e^{X_3/4}) + 9(X_4 + e^{X_4/4}) + 4(X_5 + e^{X_5/4}) \leq 175$$

$$7X_1 e^{X_1/4} + 8X_2 e^{X_2/4} + 8X_3 e^{X_3/4} + 6X_4 e^{X_4/4} + 9X_5 e^{X_5/4} \leq 200$$

The solution was determined as

$X_1 = 2.677995$

$X_2 = 2.352543$

$X_3 = 2.070096$

$X_4 = 3.530631$

$X_5 = 2.792024$

Execution time = 2.86 seconds

This is very close to the results obtained by the separable programming method in Chapter Two. The results of the separable programming method (Chapter Two) with a grid size of 0.1 were

$X_1 = 2.70000$

$X_2 = 2.32929$

$X_3 = 2.10000$

$X_4 = 3.50000$

$X_5 = 2.80000$

and the results of solution by the discrete maximum principle were [45]

$X_1 = 2.6000$

$X_2 = 2.2816$

$X_3 = 2.0075$

$X_4 = 2.6882$

$X_5 = 3.3981$

5.5 TWO PROBLEMS FROM THE COLVILLE STUDY

The next two problems were taken from "A Comparative Study of Nonlinear Programming Codes", by A.R. Colville [10]. The report is often referred to as the Colville study. The purpose of the study was to compare and evaluate

existing nonlinear programming codes.

The first problem considered is a modified version of test problem number two in the study. It was modified by Guigou [30] to exemplify usage of the GREG code. It is presented in that form. The problem is to maximize

$$f_0(\bar{X}) = \sum_{i=1}^{10} b_i X_{i+5} - \sum_{i=1}^{5}\sum_{j=1}^{5} c_{ij} X_i X_j - 2 \sum_{j=1}^{5} d_j X_j^3$$

subject to the quadratic constraints

$$\sum_{j=1}^{10} a_{i,j+5} X_{j+5} - e_i - 2 \sum_{j=1}^{5} c_{ji} X_j - 3d_i X_i^2 \leq 0$$

$$i = 1,2,3$$

$$\sum_{j=1}^{10} a_{i,j+5} X_{j+5} - e_i - 2 \sum_{j=1}^{5} c_{ji} X_j - 3d_i X_i^2 + X_{i+12} = 0$$

$$i = 4,5$$

$$0 \leq X_k \leq 100, \quad k = 1, \ldots, 17$$

Table 5.1 gives the parameter values. The problem was defined in the Colville study with five inequality constraints. The slack variables have been added to the last two constraints (X_{16} and X_{17}) to exemplify the GREG code when a nonfeasible starting point is encountered with both inequality and equality constraints.

The solution obtained was

$$f_0 = -32.34867$$

$$X_1 = 0.2994560$$

$$X_2 = 0.3332918$$

$$X_3 = 0.4006066$$

i	1	2	3	4	5
d_i	4	8	10	6	2
e_i	-15	-27	-36	-18	-12

i	1	2	3	4	5	6	7	8	9	10
b_i	-40	-2	-1/4	-4	-4	-1	-40	-60	5	1

Value of c_{ij}:

i \ j	1	2	3	4	5
1	30	-20	-10	32	-10
2	-20	39	- 6	-31	32
3	-10	- 6	10	- 6	-10
4	32	-31	- 6	39	-20
5	-10	32	-10	-20	30

Table 5.1 Parameter values for the modified Colville test problem two.

Value of a_{ij}:

i \ j	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	-16	2	0	1	0
7	0	-2	0	0.4	2
8	- 3.5	0	2	0	0
9	0	-2	0	-4	-1
10	0	-9	-2	1	-2.8
11	2	0	-4	0	0
12	- 1	-1	-1	-1	-1
13	- 1	-2	-3	-2	-1
14	1	2	3	4	5
15	1	1	1	1	1

Table 5.1 (continued)

$X_4 = 0.4287636$

$X_5 = 0.2243463$

$X_6 = X_7 = 0.0$

$X_8 = 0.5175596$

$X_9 = 0.0$

$X_{10} = 0.3081506$

$X_{11} = 0.1183264$

$X_{12} = X_{13} = 0.0$

$X_{14} = 0.1027765$

$X_{15} = X_{16} = X_{17} = 0.0$

Execution time = 12.88 seconds

The best solution in the Colville study produced an objective function value of -32.349.

The second problem is test problem number three. It involves a quadratic objective function with six quadratic constraints. The problem is defined as minimize

$$f_0(\bar{X}) = 5.3578547\ X_3^2 + 0.8356891\ X_1X_5 + 37.293239\ X_1, - 40792.141$$

subject to

$$85.334407 + 0.0056858\ X_2X_5 + 0.0006262\ X_1X_4 - 0.0022053\ X_3X_5 \begin{matrix} \geq 0 \\ \leq 92 \end{matrix}$$

$$80.51249 + 0.0071317\ X_2X_5 + 0.0029955\ X_1X_2 + 0.0021813\ X_3^2 \begin{matrix} \geq 90 \\ \leq 110 \end{matrix}$$

$$9.300961 + 0.0047026\ X_3X_5 + 0.0012547\ X_1X_3 + 0.0019085\ X_3X_4 \begin{matrix} \geq 20 \\ \leq 25 \end{matrix}$$

$78 \leq X_1, \leq 102$

$33 \leq X_2, \leq 45$

$27 \leq X_3, X_4, X_5 \leq 45$

In this problem the optimum solution was not reached by the normal procedure. It was found that the code would stop too prematurely with stopping condition number seven. The problem was that the number of desired significant digits and the magnitude of the objective function required more precision than was available. To solve the problem a constant was subtracted from the objective function to scale it. The constant value was 30373.94. The final solution was then found to be

$$f_0 = -(292.01 + 30373.94) = -30665.95$$

$$X_1 = 78.00000$$

$$X_2 = 33.00000$$

$$X_3 = 29.99389$$

$$X_4 = 45.00000$$

$$X_5 = 36.77612$$

Execution time = 1.65 seconds

The best solution given in the Colville study was -30665.5.

5.6 SOLUTION TO THE GEOMETRIC PROGRAMMING EXAMPLE OF CHAPTER TWO

For the purpose of comparison, the primal problem of the geometric programming example of Chapter Two was solved using the GREG code. The result was very close to the previous solution.

$$|f_o| = 87.98774$$

$$X_1 = 5.087201$$

$$X_2 = 2.681470$$

$$X_3 = 7.33074$$

Execution time = 0.83 seconds

5.7 CONCLUDING REMARKS

The GREG code is very successful at solving nonlinear programming problems and appears to execute very efficiently. It does have some faults. One is that it is very sensitive to numerical error and behaves very erratically without any indication of what the problems are. Usage could be aided by the implementation of an error routine. FORTRAN programs of the magnitude of GREG are extremely difficult to control. For this reason the user should be proficient at FORTRAN programming.

The programming of the external subroutines could be aided by restructuring the GREG common block. Only a few variables and vectors are needed in the user supplied subroutines PHIX, CPHI, JACOB, and GRADFI. If these variables were placed in a named common block separate from the remaining GREG common variables, it would enhance programming by relieving the user of the possibility of creating conflicting variable names. Errors of that type are extremely difficult to locate since an error message would probably point to the permanent subroutines because a variable in common was unknowingly altered.

Despite the minor difficulties, the GREG code is extremely easy to use when considering the complexity of the problems it is designed to optimize. Being coded in FORTRAN, it may be used by a large number of programmers and could be altered to meet specific, specialized needs. The program is fast, efficient, and reasonably accurate in single precision. The code is obviously an enormous contribution to nonlinear optimization.

CHAPTER SIX

APPLICATION OF THE GRG METHOD TO A COMPLEX WATER QUALITY CONTROL SYSTEM

6.1 INTRODUCTION TO THE WATER QUALITY MANAGEMENT PROBLEM

Excessive degradation of water quality in streams has resulted from the rapid industrial growth of the past two decades. The realization has become apparent that even the largest rivers can no longer be treated as infinite sinks for human and industrial waste. Several factors are involved. The most obvious to the average person is the destruction of the esthetics in a normal ecological system. A more economically oriented problem is that the quality of water passed to downstream users may require excessive treatment to be used effectively, especially in the case of drinking water. Solutions to the water quality problems are in the realm of the political and social structure of our society. The problems must be compromised from all points of view so that the socio-ecological system serves for the mutual benefit of all members and not the destruction of any member.

The systems analysis approach to water quality management will by no means transpose water quality problems from the social and political system in our society, but as indicated by past experiences, can offer invaluable insights to the decision making processes. Much value can be derived from simplified mathematical models as long as the assumptions of the model have been effectively realized.

As early as 1925, the pioneering study of Streeter and Phelps [41] first attempted to apply the mathematical modeling approach to water quality control. Since that time, numerous problems have been defined and various linear and nonlinear models applied to them. A significant step has been the development of the dispersion models [20,22,23]. Traditionally the problems involved only one water quality standard, usually DO (dissolved oxygen), at a time. The usual form of pollution has been described as BOD (biochemical oxygen demand). Recently, models have been extended to include thermal pollution and water temperature

as a quality standard [16,17,18,23].

The evolution of water quality models has imposed two major problems on the systems analysis approach. The first, and the most severe, is the general lack of data to estimate the mathematical parameters. This problem is more acute as the model becomes more complex. A second problem is the complex array of mathematical equations that describe the system and difficulties in solving them. Of course, if enough data are available complex numerical methods to perform the mathematical analysis are worth the effort. The availability of data usually determines the model used and will ultimately affect the results of any study.

In this chapter a water quality management model that includes DO and temperature as quality standards, is presented. The model was orginally proposed by Dysart and Hines [17]. Their approach was to define a multistage system with a treatment cost objective function and optimize it using a dynamic programming algorithm. The present chapter is concerned with reformulation of the model so that the generalized reduced gradient method via the GREG program can optimize it. The objective of the research was to demonstrate the effectiveness of the GRG method on water quality management problems and to demonstrate its flexibility for application to more involved models.

6.2 DO, BOD, AND TEMPERATURE RELATIONSHIPS IN FLOWING STREAMS

To develop the mathematical model, adequate transfer relationships of the state variables, temperature, BOD, and DO must be developed. The following transfer relations have been termed first order models and are extensions of the original Streeter-Phelps formulation [41].

The natual temperature of a stream is characterized by seasonable fluctuations and stochastic fluctuations within a given season. This makes it a

difficult variable to study in water quality control. One basic assumption is for any given time of the year the stream is characterized by an equilibrium temperature. When this temperature is raised by an unnatural process such as thermal pollution the stream temperature will tend to reduce to the equilibrium temperature by first order decay. The model is based, then, on the excess temperature of the stream.

$$\frac{d(T-E)}{dt} = - K(T-E)$$

T = water temperature in ^{o}F

E = equilibrium temperature in ^{o}F

t = time in days

K = temperature dissipation rate coefficient in $days^{-1}$

The analytical solution to this first order differential equation is

$$T_2 = E + (T_1-E) \exp (-K (t_2 - t_1))$$

where

T_1 = initial water temperature at an upstream point in ^{o}F

T_2 = water temperature at a downstream point in ^{o}F

t_2-t_1 = flow time from the initial starting point, t_1, to the downstream point, t_2, in days

The condition $T-E \geq 0$ is imposed on this model.

The development of the BOD transfer equation assumes that the normal BOD concentration is zero and that the dissipation of BOD follows a first order decay model.

$$\frac{dL}{dt} = -K_1 L$$

L = BOD concentration in mg/ℓ

K_1 = deoxygenation rate coefficient in $days^{-1}$

t = time in days

This differential equation has the simple solution

$$L_2 = L_1 \exp\left(-K_1(t_2-t_1)\right)$$

where

L_1 = initial BOD concentration in mg/ℓ at time = t_1

L_2 = BOD concentration in mg/ℓ at time = t_2

t_2-t_1 = flow time in days

The dissolved oxygen transfer relation is derived from the classical Streeter-Phelps formulation [41]. It assumes that a saturation level exists in the stream at a given temperature and when the dissolved oxygen is depleted below this level the deficit approaches zero by a first order decay model. It also assumes the dissolved oxygen deficit grows with the influx of BOD by a first order growth model. The relation is

$$\frac{dD}{dt} = K_1 L - K_2 D$$

where

D = DO deficit in mg/ℓ

K_2 = reaeration rate coefficient in $days^{-1}$

t = time in days

Knowing the BOD transfer equation the DO relationship becomes

$$\frac{dD}{dt} = K_1 L_1 \exp\left(-K_1(t-t_1)\right) - K_2 D$$

$$\frac{dD}{dt} + K_2 D = K_1 L_1 \exp(-K_1(t-t_1))$$

which is a first order linear differential equation. The solution is

$$D_2 = \frac{K_1 L_1}{K_2 - K_1}\left[\exp(-K_1(t_2-t_1)) - \exp(-K_2(t_2 - t_1))\right]$$

$$+ D_1 \exp(-K_2 (t_2 - t_1))$$

where

D_1 = initial DO deficit in mg/ℓ

D_2 = DO deficit in mg/ℓ at time t_2

$t_2 - t_1$ = flow time in days

D_2 is the DO deficit and it can be represented as

$$D_2 = DO_s - DO$$

where

DO_s = saturation DO level in mg/ℓ at a given temperature at time = t_2

DO = DO level in mg/ℓ at time = t_2

The DO transfer relation becomes

$$DO = DO_s - \frac{K_1 L_1}{K_2 - K_1}\left[\exp(-K_1 (t_2-t_1)) - \exp(-K_2 (t_2-t_1))\right]$$

$$-D_1 \exp(-K_2 (t_2-t_1))$$

Temperature interacts with the BOD and the DO profiles and has drastic effects on the DO saturation level for a flowing stream. These interactions have been studied intensively. The DO saturation level of a stream has been

derived as a polynomial relation [42].

$$DO_s = 14.652 - 0.41022\ TC + 0.0079910\ TC^2 - 0.000077774\ TC^3$$

where

DO_s = DO saturation level in mg/ℓ

TC = temperature in ^oC

This curve was fitted to a set of data with a multiple correlation coefficient of 0.99980.

Likewise, the coefficents K_1 and K_2 are related to temperature by the relationships [19,11]

$$K_1 = K_{1,20}\ (1.047)^{(TC-20)}$$

$$K_2 = K_{2,20}\ (1.024)^{(TC-20)}$$

where

$K_{1,20}$ = deoxygenation rate coefficient at 20 ^oC in $days^{-1}$

$K_{2,20}$ = reaeration rate coefficient at 20 ^oC in $days^{-1}$

TC = temperature in ^oC

A referral to the DO transfer relation indicates how dependent it is on the stream temperature.

The transfer relations represent a simplified model of a flowing stream and the study continued with this concept in mind. It must be realized that correct results are jeopardized by applying the parameter estimations of DO_s, K_1, and K_2 to any existing stream. Regional variations in climate, geography, and soil composition justify doubt in their validity in application to streams other than the one for which they were developed.

The following assumptions about the transfer equations are implied:

1. uniform stream width
2. uniform stream depth
3. uniform and constant flow

6.3 DEFINITION OF THE N-STAGE RIVER BASIN MODEL

A river basin may now be modeled by dividing the system into stages, one at each point of discontinuity. A point of discontinuity consists of conditions that violate the assumptions of the transfer equations, changes in water quality standards, or boundaries determined by law that define a region under separate jurisdiction. Examples of discontinuities that require a new stage are

1. organic and/or thermal waste inputs
2. a state line
3. a change in the flow regime such as changes in stream width or depth
4. increases in the flow such as tributary inputs
5. changes in water quality standards

Figure 6.1 due to Dysart and Hines [17] exemplifies a generalized river basin model.

Consider a source at a stage, n, that has both organic and thermal input. The amount of treatment that must be supplied to its pollution input depends on two factors:

1. the quality of the stream water entering the stage from upstream sources
2. the water quality standards within the given stage.

Suppose that reliable data may be acquired to estimate the following values.

XT_n = thermal input in BTU/hr at the begining of stage n

XB_n = organic, BOD, input in lbs/day at the beginning of stage n

Q_n = uniform flow within stage n in ft^3/sec

TF_n = time of flow from the beginning to the end of stage n, in days (t_2-t_1)

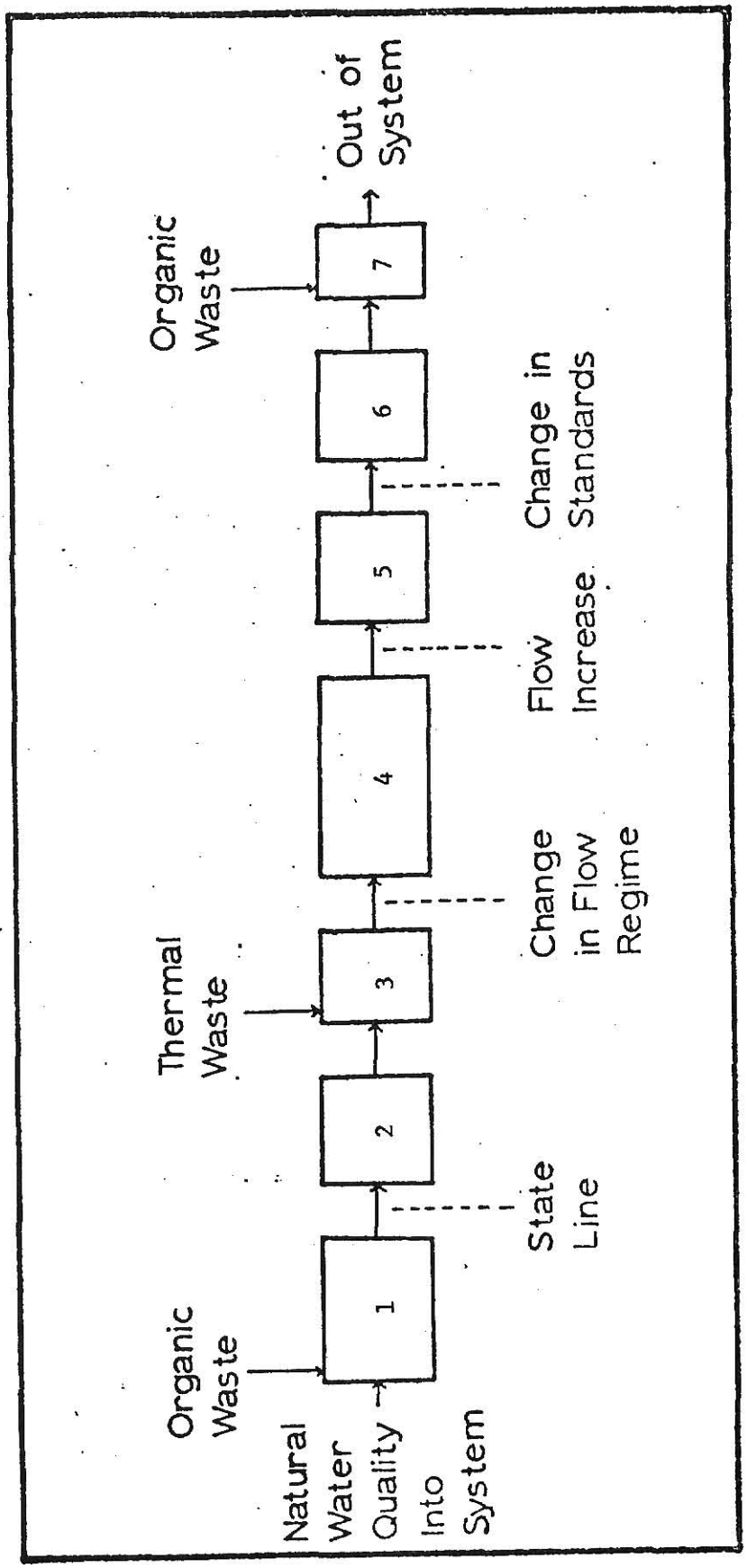


Fig. 6.1. An example of the discrete staging of the river basin model [17].

It is assumed that XT_n, XB_n, Q_n, and TF_n are constant. This is an unrealistic assumption because each value will vary daily, weekly, monthly, and seasonally. But, to formulate the optimization problem stochastic fluctuations are not considered and a static, steady state assumption is made. The rationalization is that if conditions that contribute to pollution such as low flow and high pollution input are studied then the system will behave favorably under normal conditions. It then becomes a question of how extreme should the extreme conditions be. This is a question that cannot be answered by the systems analyst. It becomes a philosophical question as to what risk of stream pollution is society willing to accept. One opinion is that as little risk as possible should be accepted. That is, very extreme conditions should be studied and treatment facilities should be designed so they will not be overloaded under the stress.

Thermal input in BTU/hr can be converted to the rise in stream temperature by the relation

$$\text{thermal input in } {}^{o}F = \frac{C_1 \cdot XT_n}{Q_n}$$

where

C_1 is a conversion constant

$$C_1 = 4.44970 \times 10^{-6} \quad \left(\frac{hr \cdot ft^3 \cdot {}^{o}F}{sec \cdot BTU}\right)$$

The conversion of BTU to the rise in water temperature in ^{o}F and of water volume to mass is relative to the water temperature at 39.1 ^{o}F. This is not an exact conversion at other temperatures, but the error is small when compared to the magnitude and significance level of a typical thermal pollution input value.

A similar expression is obtained when converting the BOD input to mg/ℓ.

$$\text{BOD input in mg/}\ell = \frac{C_2 \cdot XB_n}{Qn}$$

$$C_2 = 0.185404 \left(\frac{\text{days}\cdot\text{ft}^3\cdot\text{mg}}{\text{sec}\cdot\text{lbs}\cdot\ell}\right)$$

The values for deriving the conversion factors, C_1 and C_2, were obtained from a handbook containing physical constants [32].

The temperature rise of the stream is now represented as a function of a treatment decision variable, d(T,n), for the thermal input at stage n.

$$\text{t-rise}_n = \left[\frac{C_1 \cdot XT_n}{Q_n}\right]\left[\frac{100-d(T,n)}{100}\right]$$

The expression represents the rise in stream temperature after a cooling treatment of d(T,n) per cent.

d(T,n) = % of thermal input cooled to the equilibrium water temperature in stage n.

The maximum stream temperature occurs at the thermal pollution source and is represented as

$$\text{temp}_n = ST_n + \left[\frac{C_1 \cdot XT_n}{Q_n}\right]\left[\frac{100-d(T,n)}{100}\right]$$

where ST_n is the incoming temperature for stage n.

Three water quality constraints for each stage are considered from the above temperature equations and the DO transfer equation.

$$\text{t-rise}_n \leq TRS_n$$

$$\text{temp}_n \leq TMAX_n$$

$$\min DO_n \geq MDO_n$$

The minimum DO constraint is basic because the normal ecological process of the stream is dependent on DO concentration.

Of extreme importance to the N-stage river basin model formulation are the boundary conditions of the state variables, ST_n, SB_n, and SO_n(temperature,

BOD, and DO, respectively). The output from stage n for a state variable must equal the input for stage n+1. The boundary relations are given in terms of the state variables.

Temperature

$$ST_{n+1} = E + (T_n - E) \exp(-K \cdot TF_n)$$

$$T_n = temp_n = ST_n + \left[\frac{C_1 \cdot XT_n}{Q_n}\right]\left[\frac{100-d(T,n)}{100}\right]$$

where

ST_n = incoming temperature for stage n, in ^{o}F

ST_{n+1} = outgoing temperature for stage n and the incoming temperature for stage n+1, in ^{o}F

BOD

$$SB_{n+1} = L_n \exp(-K_1 \cdot TF_n)$$

$$L_n = SB_n + \left[\frac{C_2 \cdot XB_n}{Q_n}\right]\left[\frac{100 - d(B,n)}{100}\right]$$

where $d(B,n)$ = % treatment of BOD at the beginning of stage n

SB_n = incoming BOD for stage n, in mg/ℓ

SB_{n+1} = outgoing BOD for stage n and the incoming BOD for stage n+1, in mg/ℓ

DO

$$SO_{n+1} = DO_s - \frac{K_1 L_n}{K_2 - K_1}\left[\exp(-K_1 \cdot TF_n) - \exp(-K_2 \cdot TF_n)\right] - D_1 \exp(-K_2 \cdot TF_n)$$

D_1 = initial DO saturation level - SO_n

= initial DO deficit for stage n

SO_n = incoming DO for stage n in mg/ℓ

SO_{n+1} = outgoing DO for stage n and the incoming DO for stage n+1. For all cases, n = 1,...,N-1.

In summary, the input of pollution into a stage is represented by the constant values XT_n and/or XB_n. If a source does not exist then the appropriate input value equals zero. Each stage is constrained by three water quality constraints based on the desired water quality standards. The BOD level is implicitly controlled by the minimum DO constraint. The boundary conditions for each stage are derived from the transfer relations and the initial state variable values, ST_1, SB_1, and SO_1 are considered constant in the steady state assumption. Figure 6.2 exemplifies a four stage system.

6.4 FORMULATION OF THE MATHEMATICAL OPTIMIZATION PROBLEM FOR THE GREG PROGRAM

This section describes the development of the water quality constraints for optimization by the GREG program. The optimization problem may be defined as minimize

$$\text{cost} = \sum_{n=1}^{N} C_n$$

subject to the constraints

$$\text{t-rise}_n \leq \text{TRS}_n$$

$$\text{temp}_n \leq \text{TMAX}_n$$

$$\text{min DO}_n \geq \text{MDO}_n$$

$$n = 1, \ldots, N$$

where C_n is the treatment cost as of function of the treatment level (decision variables) for the thermal and BOD inputs of stage n. The problem can be defined in terms of the decision variables. Let

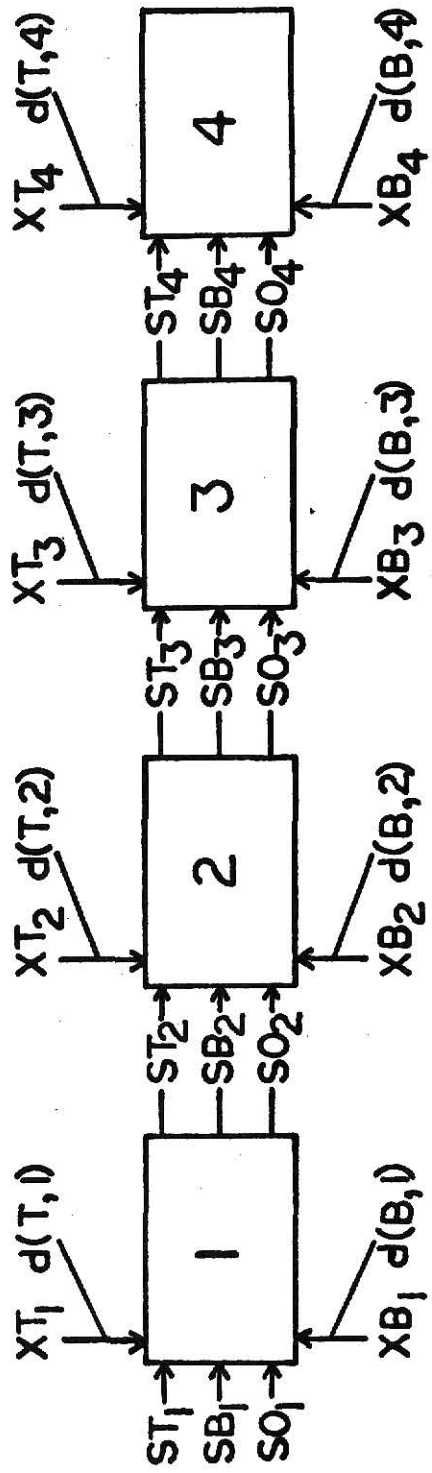


Fig. 6.2. A schematic diagram of the general river basin model with four stages. If an input does not exist in a stage then its value is appropriately set equal to zero.

$$X_i = d(T,i), \qquad X_{N+i} = d(B,i)$$

$i = 1, \ldots, N$, $\quad N$ = number of stages

The total number of decision variables is $2 \times N$. The temperature rise constraint is then given as

$$f_i(\bar{X}) = \left[\frac{C_1 \cdot XT_i}{100\ Q_i}\right] [\ 100 - X_i\ ] - TRS_i \leq 0 \qquad i = 1, \ldots, N$$

The first N constraints consist of the temperature rise constraint for each stage. The maximum temperature constraint is formulated as

$$f_{N+i}(\bar{X}) = ST_i + \left[\frac{C_1 \cdot XT_i}{100\ Q_i}\right] [\ 100 - X_i\ ] - TMAX_i \leq 0 \qquad i = 1, \ldots N$$

This set of N constraints contains the maximum temperature constraints for each stage. The final set of constraints is for the minimum DO restriction.

$$f_{2N+i}(\bar{X}) = -\min [DO] + MDO_i \leq 0 \qquad i = 1, \ldots, N$$

The total number of constraints is $3 \times N$.

Calculation of the constraints at a point, $\bar{X}$, consists of two operations. First, the boundary conditions are updated for each stage. That is, the values for ST_i, SB_i, and SO_i are calculated. Only ST_1, SB_1, and SO_1 remain constant. The values of the other state variables depend on the values of the previous decision variables. ST_i, SB_i, and SO_i are functions of X_j and X_{N+j}, $j = 1, \ldots, i - 1$.

After the boundaries are calculated the values of the constraints are determined. The calculation of the temperature rise and maximum temperature constraints is direct, but the value of the minimum DO in a stage is more difficult to obtain.

It is known that the minimum DO is between $t = 0$ and $t = TF_i$ for stage i.

The problem is to determine a value $t = t_{min}$ such that the DO concentration is minimum. A Fibonacci search is used to determine the minimum. The number of iterations is determined during the initialization of the program so that the desired precision is obtained in the calculation. The number of iterations is determined so the length of the final interval containing t_{min} is 0.0001.

Calculation of the partial derivatives is the next task in preparing the problem for the GREG program. For the temperature rise constraints the partial deriatives are determined as

$$\frac{\partial f_i}{\partial X_i} = - \left[\frac{C_1 \cdot XT_i}{100\ Q_i}\right]$$

$$\frac{\partial f_i}{\partial X_j} = 0,\ j = 1, \ldots, N \text{ and } j \neq i$$

$$\frac{\partial f}{\partial X_{N+i}} = 0,\ j = 1, \ldots, N$$

For this set of constraints the partial derivatives are all constant. Similar constant results occur with the maximum temperature constraint except the calculations involve previous decision variables.

$$\frac{\partial f_{N+i}}{\partial X_i} = - \left[\frac{C_1 \cdot XT_i}{100\ Q_i}\right]$$

$$\frac{\partial f_{N+i}}{\partial X_j} = 0,\ j > i$$

$$\frac{\partial f_{N+i}}{\partial X_j} = \frac{\partial ST_i}{\partial X_j},\ j < i$$

$$\frac{\partial f_{N+i}}{\partial X_{N+j}} = 0, \; j = 1, \ldots, N$$

The partial derivatives of the state variable, ST_i, are calculated inductively.

$$ST_i = XE_{i-1} + (T_i - XE_{i-1}) \exp(-K_{i-1} TF_{i-1})$$

$$T_i = ST_{i-1} + \frac{C_1 \cdot XT_{i-1}}{100 \, Q_{i-1}} [\, 100 - X_i \,]$$

K_{i-1} = (K) the temperature dissipation rate coefficient specified for a given stage, i - 1. This subscripting has not been used in the previous discussion, but this, and all other constants, are generally different in each stage.

Thus,

$$\frac{\partial ST_i}{\partial X_{i-1}} = - \left[\frac{C_1 \cdot XT_{i-1}}{100 \, Q_{i-1}} \right] \exp(-K_{i-1} TF_{i-1})$$

$$\frac{\partial ST_i}{\partial X_{i-k}} = - \left[\frac{C_1 \cdot XT_{i-k}}{100 \, Q_{i-k}} \right] \exp\left(- \sum_{m=1}^{k} K_{i-m} TF_{i-m} \right), \; 0 < k < i$$

By letting $j = i - k$,

$$\frac{\partial f_{N+i}}{\partial X_j} = - \left[\frac{C_1 \cdot XT_j}{100 \, Q_j} \right] \exp\left(\sum_{\ell=j}^{i-1} K_\ell \cdot TF_\ell \right), \; j < i$$

Analytic partial differentiation of the minimum DO function is considered very impractical so it is accomplished numerically. The procedure is to calculate the minimum DO at the point, $\bar{X}$, and store the value, FMIN. Then, for each

variable X_j (and X_{N+j}) the calculation is

1. FMIN1 = min [DO] when $X_j \leftarrow X_j + \Delta$

2. $$\frac{\partial f_{2N+i}}{\partial X_j} \cong -\left[\frac{\text{FMIN1} - \text{FMIN}}{\Delta}\right]$$

The minus sign comes from the minus sign in the constraint function

$$f_{2N+i} = - \min [DO_i] + MDO_i \leq 0$$

The calculations are performed only for the indices j such that $j \leq i$. Any decision, X_j and X_{N+j}, made in a downstream stage (j>i) has no effect on the previous stages and their constraints, thus indicating the partial derivatives for these cases are zero.

6.5 APPLICATION TO THE CHATTAHOOCHEE RIVER BASIN

To test the application of the GRG method to a N-stage water quality management model, data obtained from the study of Dysart and Hines [16,17,18] on the Chattahoochee River Basin below Atlanta, Georgia was utilized. A map of the system is shown in Fig. 6.3 [17]. The portion of the river under consideration in the study was bounded above at Atlanta, specifically at Clayton. The lower bound of the system was Yellowjacket Creek. There are numerous points of organic and thermal pollution input along this stretch of the river, but only the major contributers were considered in the study. The only major organic polluter is the Clayton plant. There are three major thermal polluters, McDonough, Atkinson, and Yates power plants. Four sources comprise the majority of both the organic and thermal waste in the system under consideration.

Figure 6.4 [17] summarizes the system and divides it into four steady-

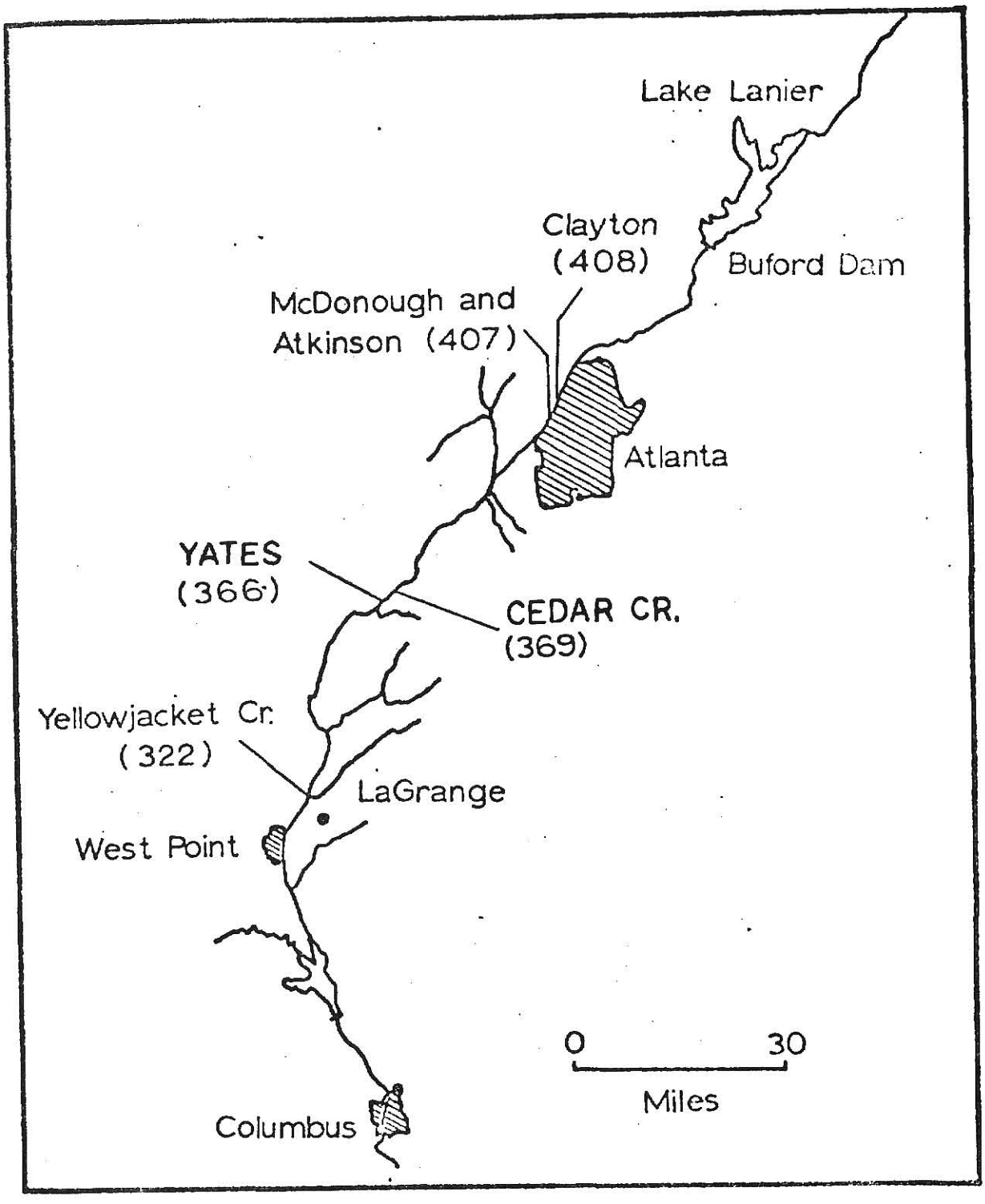
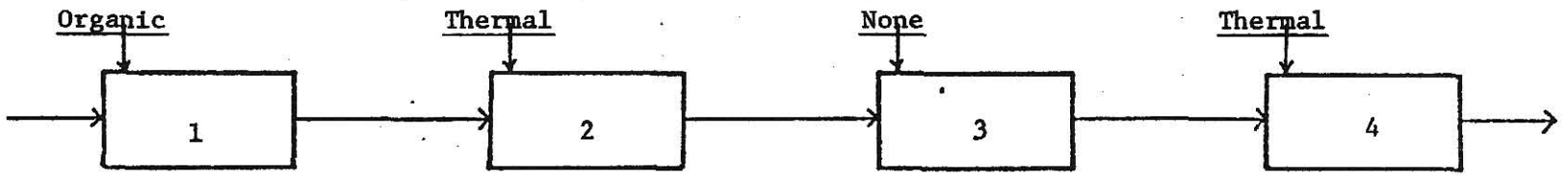


Fig. 6.3. A map of the Chattahoochee River basin [17].



Stage	Location	Waste	Water Quality Standards		
			DO(mg/ℓ)	TMAX(°F)	TRIS(°F)
1	Clayton to McDonough and Atkinson (river mile 408-407)	organic	3	93	10
2	McDonough and Atkinson to Cedar Creek (river mile 407-369)	thermal	3	93	10
3	Cedar Creek to Yates (river mile 369-366)	none	4	93	10
4	Yates to Yellowjacket Creek (river mile 366-322)	thermal	4	93	10

Fig. 6.4. The staging and the water quality standards of the river basin model [17].

state stages. Stage one describes the organic waste input and the resultant effect on the system for the Clayton plant. This stage extends to the McDonough and Atkinson plants where a discontinuity occurs with thermal discharge. These two plants are close together and can be treated as one source for stage two. The second stage extends to Cedar Creek where another discontinuity occurs. At Cedar Creek the DO standard changes as indicated in Fig. 6.4. The third stage is created and extends to the Yates plant, another thermal source. This becomes the upper bound for the fourth stage which extends to Yellowjacket Creek, the lower bound of the system. In this example, discontinuites resulting in new stages occured, for all practical reasons, at pollution sources and because of a change in the DO standard. It was not considered necessary to divide the system into more stages because of changes in flow regime or for flow increases. The model described in Fig. 6.4 is believed to represent the system within the assumptions of the water quality management river basin model.

Input data for the model for the purpose of the present study are given in Table 6.1. The data were extracted from the "One-in-Ten September" case of Dysart and Hines [17]. This means that the lowest flow of the river occuring in September during the previous ten years was used as the basis for a conservative abatement, or treatment policy for the system. It was reasoned that if the extreme case was studied for a given time of the year then policy based on it would be satisfactory for more favorable conditions. Of course, this would not be true where significant BOD concentration is created by runoff (such as feedlot runoff) in high flow conditions.

The objective function of the system was given in terms of treatment costs for the three stages with pollution input. Figure 6.5 [17] shows the cost of organic waste treatment in millions of dollars as a function of the abatement level for the Clayton plant. This function, for the purpose of this

n	Location	XT_n (BTU/hr)	XB_n (lbs/day)	Q_n (ft^3 /sec)	XE_n (^oF)	K ($days^{-1}$)	$K_{1,20}$ ($days^{-1}$)	$K_{2,20}$ ($days^{-1}$)	TF_n (days)	distance (miles)
1.	Clayton to McDonough and Atkinson	0	1.1×10^5	820	80.7	1.24	1.00	0.90	0.05	1
2.	McDonough and Atkinson to Cedar Creek	2.4×10^9	0	870	80.7	1.29	1.00	0.90	1.87	38
3.	Cedar Creek to Yates	0	0	885	80.7	1.10	1.00	0.90	0.17	3
4.	Yates to Yellowjacket Creek	1.6×10^9	0	920	80.7	1.34	1.00	0.90	2.40	44

Table 6.1. The input parameters for the "One-in-Ten September" case.

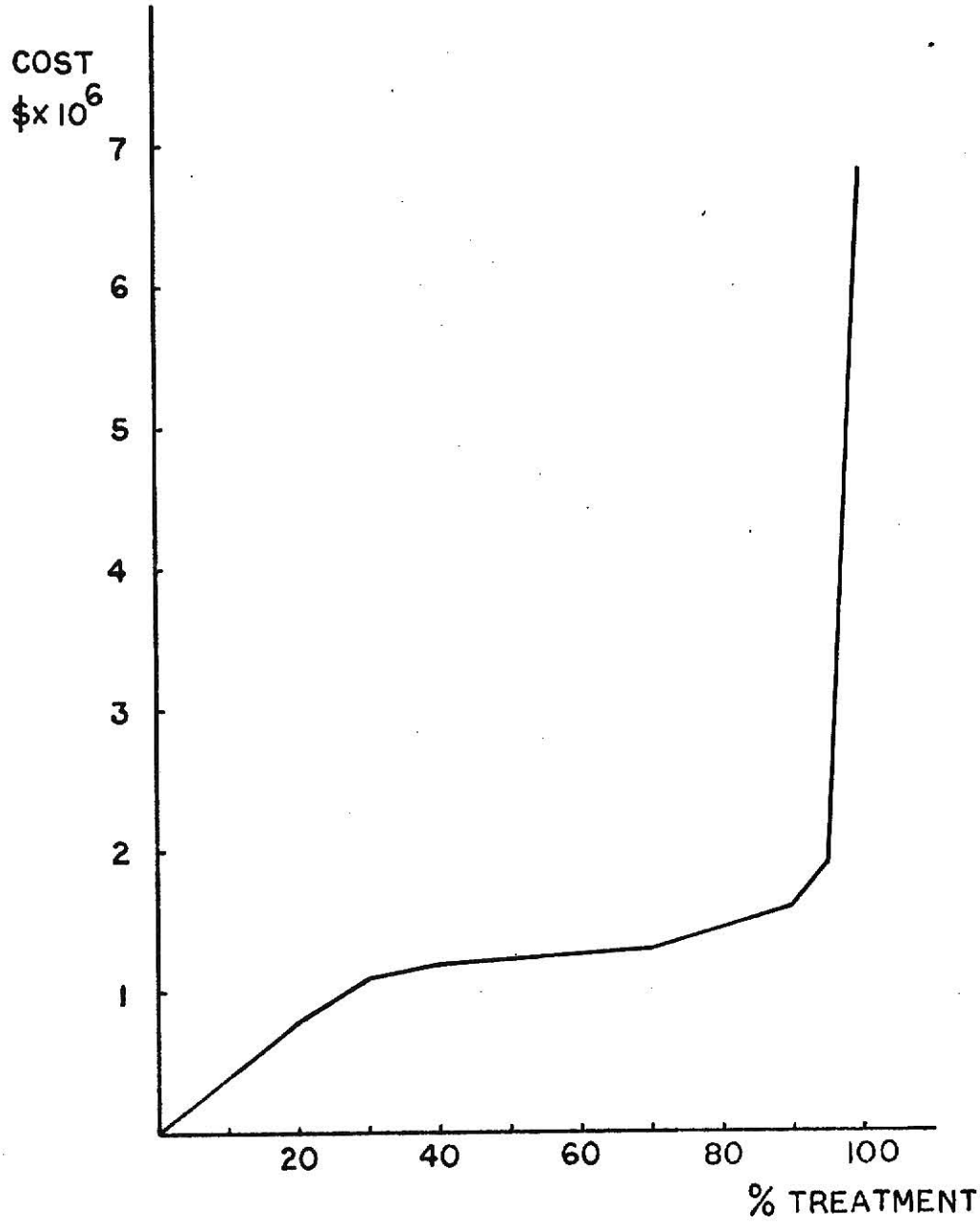


Fig. 6.5. The Clayton plant abatement cost of organic waste treatment as a function of per cent treatment.

study, is represented by a piecewise linear function.

X = % treatment

$$\text{cost} = 0.04X, \quad 0 \leq X < 20$$

$$\text{cost} = 0.03X + 0.2, \quad 20 \leq X < 30$$

$$\text{cost} = 0.01X + 0.8, \quad 30 \leq X < 40$$

$$\text{cost} = \left(\frac{0.01}{3}\right) X - \left(\frac{0.40}{3} - 1.2\right), \quad 40 \leq X < 70$$

$$\text{cost} = 0.015X + 0.25, \quad 70 \leq X < 90$$

$$\text{cost} = 0.06X - 3.8, \quad 90 \leq X < 95$$

$$\text{cost} = 0.98X - 91.2, \quad 95 \leq X < 100$$

The cost for the treatment of the thermal waste was given as (in $)

X = % treatment

$$\text{Cool Cost (McDonough and Atkinson)} = \left(\frac{817{,}000}{0.9}\right) [\, 1 - \exp(-0.023\, X)\,]$$

$$\text{Cool Cost (Yates)} = \left(\frac{575{,}000}{0.9}\right) [\, 1 - \exp(-0.023\, X)\,]$$

The objective function is separable and its gradient is easily defined for optimization with the GREG program.

The programming of the external GREG subroutines is given in Appendix 5. The results are encouraging since both accuracy and computation time are favorable when using the GRG method. The accuracy of the problem analysis is improved over the dynamic programming approach because the system is treated on a continuous basis. The main improvement, though, is that computation to an optimum solution is significantly faster than the dynamic programming approach.

The abatement policy was determined to be

% treatment = 66.9%
(Clayton)

% treatment = 18.5%
(McDonough
and
Atkinson)

% treatment = 0.0%
(Yates)

The total cost of the system abatement policy by the cost function used was $1.60 million. The cost function used for the Clayton plant was a little different from the Dysart and Hines study [17] because values for it were extrapolated from a graph. The solution to this problem from their study was

% treatment = 70%
(Clayton)

% treatment = 20%
(McDonough
and
Atkinson)

% treatment = 0.0%
(Yates)

Total cost = 1.576×10^6

The execution time of the GREG program on the IBM 360/50 computer from a feasible starting point of 100% treatment for all inputs was 39.41 seconds. From the nonfeasible starting point of 0% treatment for all sources the execution time was 30.35 seconds. The execution times cannot be compared directly but the dynamic programming approach was attributed a typical 10-15 minute execution time on a Univac 1108 computer [18]. The program was written in the ALGOL language.

The resulting DO profile is shown in Fig. 6.6. A minimum DO concentration is reached in stage two. This minimum is caused by the organic waste of stage one. The temperature profile is shown in Fig. 6.7. The two jumps in the graph are caused by the thermal inputs. The variation of K_1 and K_2, which depends on temperature, is shown in Fig. 6.8.

6.6 CONCLUDING REMARKS

The application of the GRG method to the water quality management model demonstrates its flexibility, speed, and accuracy. It serves as a pilot study to further research involving more complicated water quality models that are more realistic in describing the behavior of such a complex system. Two important results of this study were that the minimum DO and its partial derivatives were determined numerically with much success. This implies applications of the technique to an assortment of constraints. The most important application to water quality models would be the numerical integration of sumultaneous differential equations to define the constraints. This makes it conceiveable to apply the procedure to the more complex dispersion models involving BOD, DO, and temperature with less restrictive assumptions (see Fan et al. [23]).

This study also demonstrated the use of a monotonically increasing piecewise linear cost function. This demonstrated the flexibility in defining objective functions for the GRG method. It is the flexibility of the GRG method that is desireable. It has a wide variety of applications including water quality optimization models.

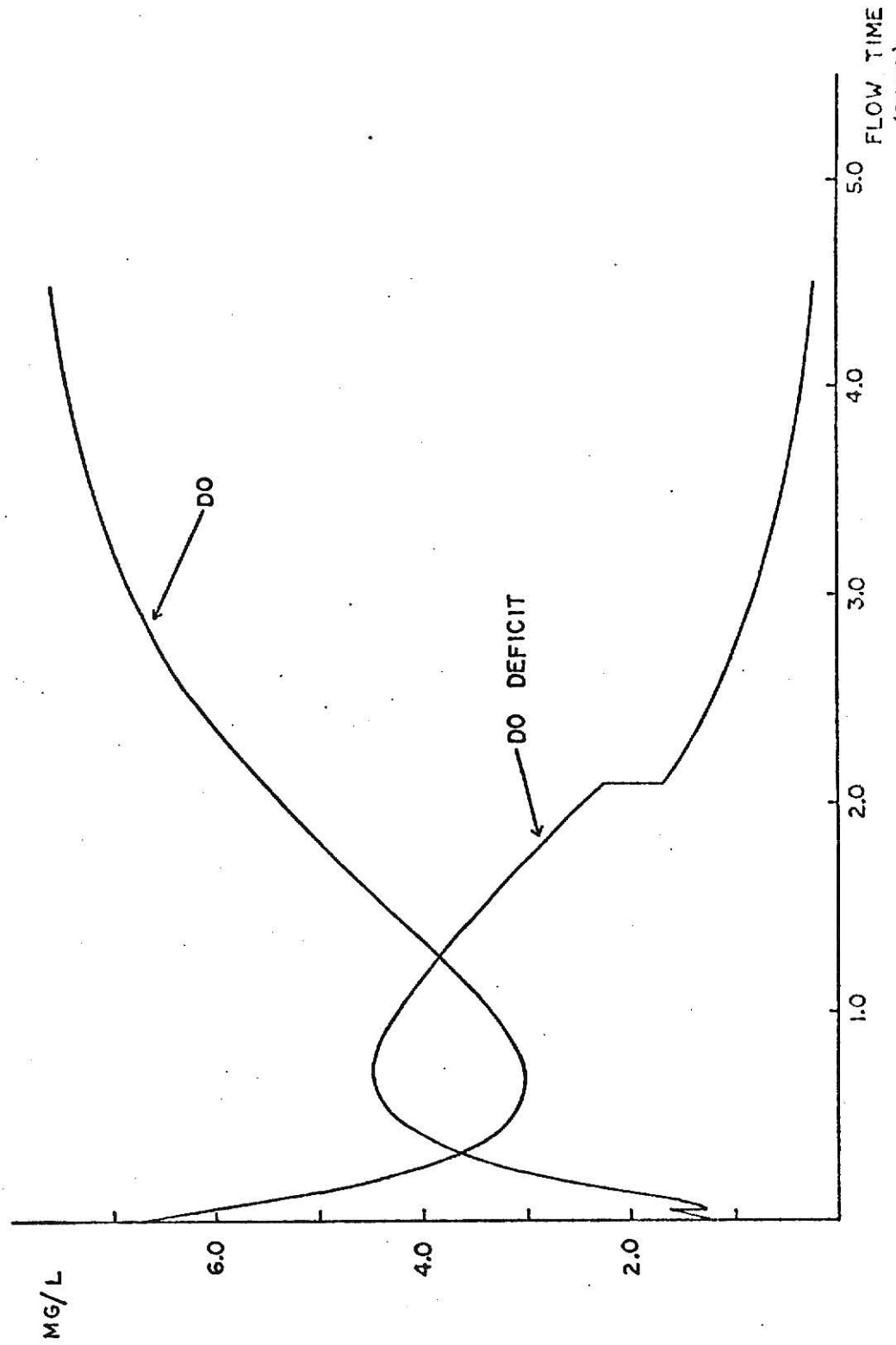


Fig. 6.6. The DO and DO deficit profiles for the system at the optimum solution. The discontinuities of the DO deficit curve represent thermal pollution input from the McDonough and Atkinson plants and the Yates plant.

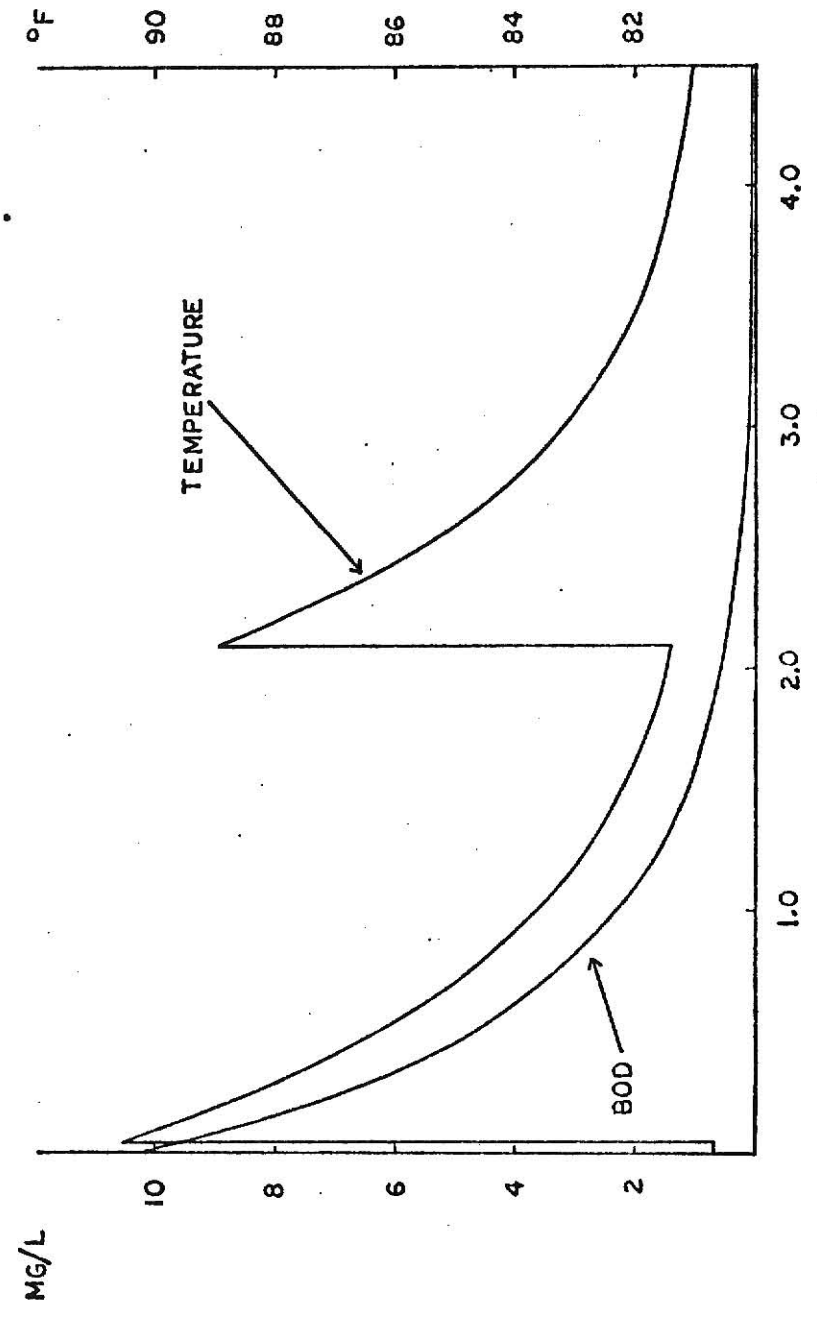


Fig. 6.7. The temperature and the BOD profiles for the system. The BOD input occurs at the Clayton plant (flow time = 0.0). The discontinuities of the temperature curve occur at the source of the two thermal inputs.

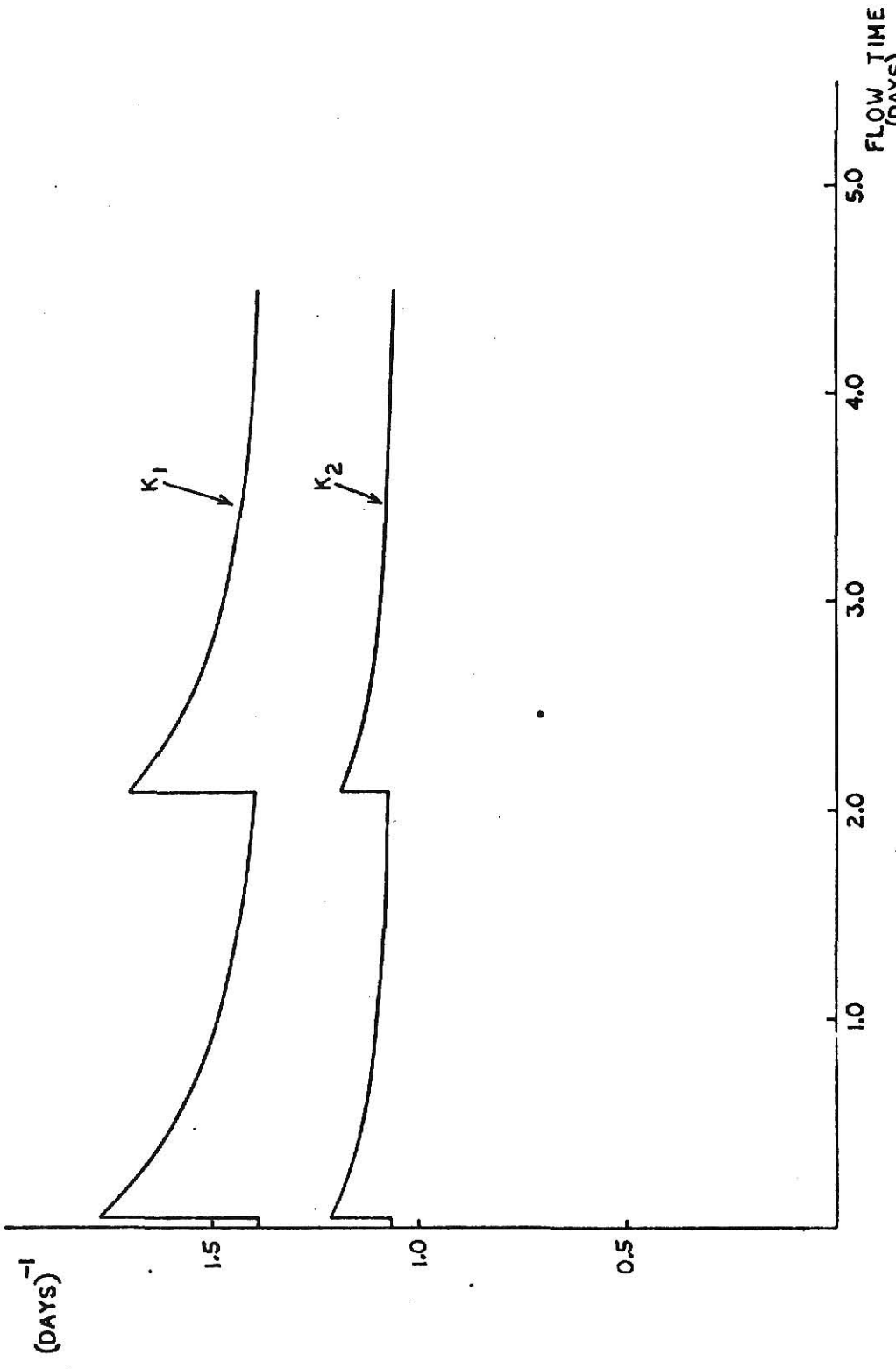


Fig. 6.8. The variation of the deoxygenation rate coefficient and the reaeration rate coefficient. Notice the close relation to the temperature curve of Fig. 6.7.

REFERENCES

1. Abadie, J., Solution des questions de dégénérescence dans la méthode GRG, EDF* note HI 143/00 du 25 Septembre 1969.

2. Abadie, J., Application of the GRG algorithm to optimal control problems, in: Integer and Nonlinear Programming, ed. J. Abadie, North Holland Publishing Co., Amsterdam, 1970.

3. Abadie, J. and J. Carpentier, Généralisation de la méthode du gradient réduit de Wolfe au cas de constraintes non-lineaires, EDF* note HR 6678, du 27 Octobre 1965.

4. Abadie, J. and J. Carpentier, Généralisation de la méthode du gradient réduit de Wolfe au cas de constraints non-lineaires, in: Proc. IFOR conf., eds. D.B. Hertz and J. Melese, Wiley, New York, 1966, pp. 1041-1053.

5. Abadie, J. and J. Carpentier, Generalization of the Wolfe reduced gradient method to the case of nonlinear constraints, in: Optimization, ed. R. Fletcher, Academic Press, New York, 1969.

6. Abadie, J. and J. Guigou, Gradient réduit généralisé, EDF* note HI 069/2 du Avril 1969.

7. Abadie, J. and J. Guigou, Numerical experiments with the GRG method, in: Integer and Nonlinear Programming, ed. J. Abadie, North Holland Publishing Co., Amsterdam, 1970.

8. Apostol, T.M., Mathematical Analysis, Addison-Wesley, Reading, Mass., 1957.

9. Beale, E.M.L., Numerical methods, in: Nonlinear Programming, ed. J. Abadie, North Holland Publishing Company, Amsterdam, 1967.

10. Colville, A.R., A comparative study of nonlinear programming codes, IBM New York Scientific Center, Report 320-2949, 1968.

11. Committee on Sanitary Engineering Research, Effect of water treatment on stream reaeration, J. Sanitary Engineering Division, Vol. 87, No. SA6, 1961, pp. 59-71.

12. Computing Center User's Guide, Kansas State University, Manhattan, Kansas, September 1971.

13. Courant, R., Differential and Integral Calculus, Vol. II, Interscience, New York, 1936.

14. Davidon, W.C., Variable metric method for minimization, A.E.C. Research and Development Report, ANL - 5990 (Rev.), 1959.

15. Duffin, R.J., E.L. Peterson, and C. Zener, Geometric Programming, Wiley, New York, 1967.

16. Dysart, B.C., III and W.W. Hines, Control of water quality in a complex natural system, IEEE Transactions on Systems Science and Cybernetics, Vol. SSC-6, No. 4, October 1970, pp. 322-329.

17. Dysart, B.C., III and W.W. Hines, Developement and Application of a Rational Water Quality Planning Model, Water Resources Center, Georgia Institute of Technology, Final Report, OWRR Project - No. A-012GA, Atlanta, Georgia, June 30, 1968.

18. Dysart, B.C., III and W.W. Hines, Water quality management and a policy model, AIIE Transactions, Vol. 3, No. 1, March 1971, pp.7-12.

19. Eckenfelder, W.W., Jr., and D.J. O'Conner, Biological Waste Treatment, Pergamon Press, Oxford, 1964.

20. Fan, L.T., R.S. Nadkarni, and L.E. Erickson, Dispersion model for a stream with several waste inputs and water intakes, Water Resources Bulletin, Vol. 7, No. 6. December 1971, pp. 1210-1220.

21. Fan, L.T., L.E. Erickson, and C.L. Hwang, Methods of Optimization Volume 3: Search Techniques, Institute for Systems Design and Optimization, Kansas State University, Manhattan, Kansas, October 1, 1971.

22. Fan, L.T., R.S. Nadkarni, and L.E. Erickson, Management of optimum water quality in a stream, Water Research, Pergamon Press, London, Vol. 5, 1971, pp. 1005-1021.

23. Fan, L.T., J.T. Tseng, and C.L. Hwang, Optimization of power plant cooling water discharge in streams, Journal of the Power Division, ASCE, Vo. 97, No. PO4, Proc. Paper 8595, December 1971, pp. 841-860.

24. Fan, L.T. and C.S. Wang, The Discrete Maximum Principle - A Study of Mutistage Systems Optimization, Wiley, New York, 1964.

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

26. Fletcher, R., and C.M. Reeves, Function minimization by conjugate gradients, The Computer Journal, Vol. 7, July 1964, pp. 149-154.

27. Fletcher, R. and M.J.D. Powell, A rapidly convergent descent method for minimization, The Computer Journal, Vol. 6, July 1963, pp. 163-168.

28. FORTRAN IV (G and H) Programmer's Guide, GC28-6817-2, International Business Machines Corporation, 1970.

29. Guigou, J., Présentation et utilisation du code GRG, EDF* note HI 582/2 du 25 Mai 1971.

30. Guigou, J., Présentation et utilisation du code GREG, EDF* note HI 102/2, du 25 Mai 1971.

31. Hadley, G., Nonlinear and Dynamic Programming, Addison-Wesley, Reading, Mass., 1964.

32. Handbook of Chemistry and Physics, 47th college edition, The Chemical Rubber Company, Cleveland, Ohio, 1966.

33. Hooke, R. and T.A. Jeeves, Direct search solution of numerical and statistical problems, J. Assoc. Compt. Mach., Vol. 8, 1961, p. 212.

34. Hwang, C.L., L.T. Fan, and S. Kumar, Hooke and Jeeves Pattern Search Solution to Optimal Production Planning Problems, Report 18, Institute for Systems Design and Optimization, Manhattan, Kansas, 1969.

35. Hwang, C.L., K.C. Lai, F.A. Tillman, and L.T. Fan, Optimization of Systems Reliability by the Sequential Unconstrained Minimization Technique, Institute for Systems Design and Optimization, Kansas State University, Manhattan, Kansas.

36. James, Robert C., Advanced Calculus, Wadsworth Publishing Co., Belmont, Cal., 1966.

37. Mathematical Programming System/360, Version 2, Linear and Separable Programming - User's Manual, GH20-0476-2, International Business Machines Corp., 1968.

38. McCormick, G.P., W.C. Mylander III, and A.V. Fiacco, RAC computer program implementing the sequential unconstrained minimization technique for nonlinear programming, SHARE Number 3189.

39. McMillan, C., Jr., Mathematical Programming, Wiley, New York, 1970.

40. Passy, U., Generalization of Geometric Programming; Partial Control of Linear Inventory Systems, PHD Thesis, Standford University Press, 1966.

41. Streeter, H.W., and E. B. Phelps, A study of the pollution and natural purification of the Ohio River, Public Health Bulletin No. 146, U.S. Public Health Service, Washington, D.C., 1925.

42. The Prediction of Stream Reaeration Rates, TVA Report, Tennessee Valley Authority, Chattanooga, Tenn., 1962.

43. Tillman, F.A. and J. Liittschwager, Integer programming formulation of constrained reliability problems, Management Science, Vol. 13, 1967, pp. 887-889.

44. Tillman, F.A., C.L. Hwang, L.T. Fan, and K.C. Lai, Optimal reliability of a complex system, IEEE Transactions on Reliability, Vol. R-19, August 1970, pp. 95-100.

45. Tillman, F.A., C.L. Hwang, L.T. Fan, and S.A. Balbale, Systems reliability subject to multiple nonlinear constraints, IEEE Transactions on Reliability, Vol. R-17, 1968, pp. 153-157.

46. Wilde, D.J. and C.L. Beightler, Foundations in Optimization, Prentice-Hall, Englewood Cliffs, N.J., 1967.

47. Wolfe, P., Methods of nonlinear programming, in: Recent Advances in Mathematical Programming, eds. R. Graves and P. Wolfe, McGraw-Hill, New York, 1963.

48. Wolfe, P., Methods for linear constraints, in: Nonlinear Programming, ed. J. Abadie, North Holland Publishing Co., Amsterdam, 1967.

* - Electricité de France

APPENDIX 1. COMPUTER PROGRAM FOR GENERATING SEPARABLE PROGRAMMING DATA

This appendix contains a listing of the main program for generating the separable programming data for MPS/360. The user must supply the subroutine FUNC.

```
C........JEREL WILLIAMS.....................JULY,1971.....................
C
C....THIS ROUTINE IS DESIGNED TO GENERATE THE LINEAR APPROXIMATION VALUES
C     AND TO PUNCH DATA CARDS WITHIN THE SPECIFICATIONS OF MPS/360
C     LINEAR AND SEPARABLE PROGRAMMING.
C
C....A USER DESIGNED SUBROUTINE FUNC MUST ACCOMPANY THIS ROUTINE.
C
C....THE USER MUST SUPPLY DATA FOR SIX VARIABLES.  ALL DATA IS ECHO CHECKED.
C         NFUN=THE NUMBER OF VARIABLES
C         NROWS=THE NUMBER OF CONSTRAINTS+1
C         START=THE INITIAL STARTING POINT FOR EACH VARIABLE
C         INTER=THE NUMBER OF UNIFORM PARTITIONS DESIRED BETWEEN ALOW & UPPER
C         ALOW=THE LOWER BOUND OF THE UNIFORM REGION
C         UPPER=THE UPPER BOUND OF THE UNIFORM REGION
C
C....THE DATA CARD FORMAT IS AS FOLLOWS.
C         FIRST CARD
C              READ NFUN & NROWS, FORMAT 2I5
C         THERE NOW OCCURS ONE DATA CARD FOR EACH VARIABLE CONTAINING
C         START, INTER, ALOW, & UPPER, FORMAT F10.0, I10, 2F10.0
C
  900 FORMAT(5X,5HERROR/5X,30HALOW EQUALS UPPER FOR VARIABLE,I5)
  901 FORMAT(5X,5HERROR/5X,31HINVALID VALUE FOR NFUN OR NROWS)
  902 FORMAT(5X,5HERROR/5X,27HINTER EQUALS 0 FOR VARIABLE,I5)
 1000 FORMAT(2I5)
 2000 FORMAT(4X,1HP,I4,5X,3HROW,I3,4X,F12.5,3X,3HROW,I3,4X,F12.5)
 3000 FORMAT(4X,1HP,I4,5X,4HGRID,I3,3X,F12.5)
 3001 FORMAT(4X,1HP,I4,5X,3HROW,I3,4X,F12.5,3X,4HGRID,I3,3X,F12.5)
 4000 FORMAT(4X,3HSET,I3,4X,8H'MARKER',17X,8H'SEPORG')
 5000 FORMAT(F10.0,I10,2F10.0)
 6000 FORMAT(5X,23HECHO CHECK FOR VARIABLE,I5/5X,6HSTART=,F10.2/
     15X,6HINTER=,I10  /5X,6HALOW =,F10.2/5X,6HUPPER=,F10.2/)
 7000 FORMAT(4X,6HENDSET,4X,8H'MARKER',17X,8H'SEPEND'/80(1H*))
 8000 FORMAT(1X,14HUP SEPBOUND  P,I4, 9X,3H1.0)
 9000 FORMAT(5X,27HECHO CHECK FOR NFUN & NROWS/
     15X,6HNFUN =,I5/5X,6HNROWS=,I5/)
C
C.....UNIT NUMBERS
C
      NREAD=5
      NPRNT=6
      NPNCH=7
C
C.....NREAD=CARD INPUT,NPRNT=PRINT OUTPUT,NPNCH=PUNCH OUTPUT
C
      DIMENSION ROW(50),F(50,50)
      READ(NREAD,1000) NFUN,NROWS
      WRITE(NPRNT,9000) NFUN,NROWS
      IF(NFUN.LT.1.OR.NROWS.LT.1) GO TO 96
      NKK=0
      MKK=1000
C
C.....DO CALCULATIONS FOR EACH VARIABLE
C
      DO400KK=1,NFUN
      IAX=1
      READ(NREAD,5000) START,INTER,ALOW,UPPER
```

```
      WRITE(NPRNT,6000) KK,START,INTER,ALOW,UPPER
C
C.....CHECK DATA FOR ERRORS.  PROGRAM DOES NOT ALLOW INTER .GT. 50.
C
      IF(INTER.GT.50) INTER=50
      IF(INTER.EQ.0) GO TO 97
      IF(ALOW.EQ.UPPER) GO TO 98
C
C.....DETERMINE UNIFORM PARTITION STEP-SIZE
C
      AN=INTER
      DELTA=(UPPER-ALOW)/AN
      LL=KK+100
      WRITE(NPNCH,4000) LL
      AN=NROWS+KK
C
C.....CHECK ALOW.NE.START.  CREATE ANOTHER PARTITION IF TRUE
C
      IJ=0
      IF(ALOW-START)2,3,2
    2 X=START
      AX=ALOW
      IJ=1
      INTER=INTER+1
      IAX=2
      DO11IK=1,NROWS
   11 ROW(IK)=0.0
      GO TO 8
C
C.....CALCULATE PARTITION VALUES
C
    3 IJ=IJ+1
      DO10IK=1,NROWS
   10 ROW(IK)=0.0
      XK=IJ-IAX
      X=XK*DELTA+ALOW
      AX=X+DELTA
C
C.....CALCULATE FUNCTION DIFFERENCES
C
    8 DO100J=1,NROWS
      CALL FUNC(J,KK,AX,F,NFUN,NROWS)
      ROW(J)=F(J,KK)
      CALL FUNC(J,KK, X,F,NFUN,NROWS)
  100 ROW(J)=ROW(J)-F(J,KK)
C
C.....CREATE GRID VALUE
C
      GRID=X-AX
      II=IJ+KKK
C
C.....START PUNCH LOOP
C
      IR=0
   16 IR=IR+1
      IF(IR.GT.NROWS) GO TO 90
      IF(ROW(IR))17,200,17
   17 IRRR=IR+1
```

```
      GO TO 20
   19 IRRR=IRRR+1
   20 IF(IRRR.GT.NROWS) GO TO 89
      IF(ROW(IRRR))75,19,75
   89 IRR=IR+100
      WRITE(NPNCH,3001) II,IRR,ROW(IR),LL,GRID
      IR=NN
      GO TO 200
   90 IR=NN
      WRITE(NPNCH,3000) II,LL,GRID
      GO TO 200
   75 IRR=IR+100
      IMM=IRRR+100
      WRITE(NPNCH,2000) II,IRR,ROW(IR),IMM,ROW(IRRR)
  199 IR=IRRR
  200 IF(IR.LT.NN) GO TO 16
      IF(IJ.LT.INTER) GO TO 3
      KKK=KKK+INTER
      NKK=NKK+INTER
  400 CONTINUE
      WRITE(NPNCH,7000)
C
C.....CREATE BOUNDS SECTION
C
      DO1I=1,NKK
      II=I+1000
    1 WRITE(NPNCH,8000) II
      STOP
   96 WRITE(NPRNT,901)
      STOP
   97 WRITE(NPRNT,902) KK
      STOP
   98 WRITE(NPRNT,900) KK
      STOP
      END
```

APPENDIX 2. COMPUTER OUTPUT FROM THE RELIABILITY PROBLEM

This appendix contains a listing of the subroutine FUNC and the MPS/360 control language program used to solve the reliability problem. The MPS/360 solution output is also given.

```
      SUBROUTINE FUNC(J,KK,X,F,NFUN,NROWS)
      DIMENSION F(50,50)
      MM=(KK-1)*NROWS+J
      GO TO(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20),MM
    1 F(1,1)=ALOG(1.-.2**X)
      RETURN
    2 F(2,1)=X**2
      RETURN
    3 F(3,1)=7.*(X+EXP(X/4.))
      RETURN
    4 F(4,1)=7.*X*EXP(X/4.)
      RETURN
    5 F(1,2)=ALOG(1.-.15**X)
      RETURN
    6 F(2,2)=2.*X**2
      RETURN
    7 F(3,2)=7.*(X+EXP(X/4.))
      RETURN
    8 F(4,2)=8.*X*EXP(X/4.)
      RETURN
    9 F(1,3)=ALOG(1.-.1**X)
      RETURN
   10 F(2,3)=3.*X**2
      RETURN
   11 F(3,3)=5.*(X+EXP(X/4.))
      RETURN
   12 F(4,3)=8.*X*EXP(X/4.)
      RETURN
   13 F(1,4)=ALOG(1.-.35**X)
      RETURN
   14 F(2,4)=4.*X**2
      RETURN
   15 F(3,4)=9.* (X+EXP(X/4.))
      RETURN
   16 F(4,4)=6.*X*EXP(X/4.)
      RETURN
   17 F(1,5)=ALOG(1.-.25**X)
      RETURN
   18 F(2,5)=2.*X**2
      RETURN
   19 F(3,5)=4.*(X+EXP(X/4.))
      RETURN
   20 F(4,5)=9.*X*EXP(X/4.)
      RETURN
      END
```

```
CONTROL PROGRAM COMPILER - MPS/360 V2-M8

C001                    PROGRAM
C002                    INITIALZ
C064                    MOVE(XDATA,'DATA-SET')
C065                    MOVE(XPBNAME,'SEPARABLE')
C066                    CONVERT
C067                    BCDOUT
C068                    MOVE(XOBJ,'ROW101')
C069                    MOVE(XRHS,'LIMITS')
C070                    TITLE('PRIMAL-LOCAL OPTIMUM')
C071                    XSETLB=-1    $* BOUNDED VARIABLES AT LOWER LIMIT
C073                    SETUP('BOUND','SEPBOUND','MAX')
C074                    PRIMAL
C075                    SOLUTION     $* LOCAL OPTIMUM
C077                    TITLE('PRIMAL-GLOBAL OPTIMUM')
C078                    XSETLB=+1    $* BOUNDED VARIABLES AT UPPER LIMIT
C080                    SETUP('BOUND','SEPBOUND','MAX')
C081                    PRIMAL
C082                    SOLUTION     $* GLOBAL OPTIMUM
C084                    TITLE('DUAL-GLOBAL OPTIMUM')
C085                    SETUP('BOUND','SEPBOUND','MAX')
C086                    DUAL
C087                    SOLUTION     $* GLOBAL OPTIMUM-DUAL
C089                    EXIT
C090                    PEND
```

SOLUTION (OPTIMAL)

TIME = 0.31 MINS. ITERATION NUMBER = 39

...NAME...	...ACTIVITY...	DEFINED AS
OBJECTIVE	1.12976	ROW101
RESTRAINTS		LIMITS
BOUNDS....		SEMBOUND

SECTION 1 - ROWS

	NUMBER	...ROW...	AT	...ACTIVITY...	SLACK ACTIVITY	..LOWER LIMIT.	..UPPER LIMIT.	.DUAL ACTIVITY
	1	ROW101	BS	1.12976	1.12976-	NONE	NONE	1.00000
	2	ROW102	BS	84.05514	13.94486	NONE	98.00000	.
	3	ROW103	BS	79.69760	22.29840	NONE	101.99200	.
	4	ROW104	UL	151.21800	.	NONE	151.21800	.00098-
A	5	GRID101	EQ	1.00000	.	1.00000	1.00000	.
A	6	GRID102	EQ	1.00000	.	1.00000	1.00000	.
A	7	GRID103	EQ	1.00000	.	1.00000	1.00000	.
A	8	GRID104	EQ	1.00000	.	1.00000	1.00000	.
A	9	GRID105	EQ	1.00000	.	1.00000	1.00000	.

SECTION 2 - COLUMNS

NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
10	X1	BS	2.70000	.	.	NONE	.
11	X2	BS	2.32979	.	.	NONE	.
12	X3	BS	2.10000	.	.	NONE	.
13	X4	BS	3.50000	.	.	NONE	.
14	X5	BS	2.80000	.	.	NONE	.
15	P1C01	UL	1.00000	.18232	.	1.00000	.16957
16	P1C02	UL	1.00000	.00617	.	1.00000	.00445
17	P1C03	UL	1.00000	.00523	.	1.00000	.00343
18	P1C04	UL	1.00000	.00443	.	1.00000	.00256
19	P1C05	UL	1.00000	.00375	.	1.00000	.00180
20	P1C06	UL	1.00000	.00319	.	1.00000	.00126
21	P1C07	UL	1.00000	.00270	.	1.00000	.00058
22	P1C08	UL	1.00000	.00230	.	1.00000	.00010
23	P1C09	LL	.	.00195	.	1.00000	.00034-
24	P1010	LL	.	.00166	.	1.00000	.00073-
25	P1011	LL	.	.00141	.	1.00000	.00107-
26	P1C12	UL	1.00000	.13976	.	1.00000	.12404
27	P1C13	UL	1.00000	.00397	.	1.00000	.00200
28	P1C14	UL	1.00000	.00327	.	1.00000	.00122
29	P1C15	UL	1.00000	.00270	.	1.00000	.00056
30	P1C16	BS	.29288	.00223	.	1.00000	.
31	P1C17	LL	.	.00184	.	1.00000	.00048-
32	P1C18	LL	.	.00152	.	1.00000	.00090-
33	P1C19	LL	.	.00125	.	1.00000	.00127-
34	P1C20	LL	.	.00104	.	1.00000	.00158-
35	P1C21	LL	.	.00086	.	1.00000	.00187-
36	P1C22	LL	.	.00071	.	1.00000	.00213-
37	P1C23	UL	1.00000	.07323	.	1.00000	.06622
38	P1C24	UL	1.00000	.00669	.	1.00000	.00509
39	P1C25	UL	1.00000	.00529	.	1.00000	.00362
40	P1C26	UL	1.00000	.00418	.	1.00000	.00244
41	P1C27	LL	1.00000	.00331	.	1.00000	.00150
42	P1C28	LL	1.00000	.00262	.	1.00000	.00073
43	P1C29	UL	1.00000	.00208	.	1.00000	.00011
44	P1C30	LL	.	.00165	.	1.00000	.00040-
45	P1C31	LL	.	.00131	.	1.00000	.00083-
46	P1C32	LL	.	.00104	.	1.00000	.00119-
47	P1C33	LL	.	.00082	.	1.00000	.00150-
48	P1C34	LL	1.00000	.38696	.	1.00000	.35730
49	P1C35	UL	1.00000	.00445	.	1.00000	.00224
50	P1C36	UL	1.00000	.00399	.	1.00000	.00169
51	P1C37	LL	1.00000	.00358	.	1.00000	.00119
52	P1C38	LL	1.00000	.00321	.	1.00000	.00071
53	P1039	UL	1.00000	.00289	.	1.00000	.00033
54	P1C40	LL	.	.00259	.	1.00000	.00069-
55	P1041	LL	.	.00233	.	1.00000	.00046-
56	P1042	LL	.	.00209	.	1.00000	.00061-
57	P1043	LL	.	.00188	.	1.00000	.00113-
58	P1044	LL	.	.00169	.	1.00000	.00143-

NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
59	P1045	LL	1.00000	.25593	.	1.00000	.22619
60	P1046	UL	1.00000	.00417	.	1.00000	.00143
61	P1047	LL	1.00000	.00361	.	1.00000	.00078
62	P1048	LL	1.00000	.00314	.	1.00000	.00019
63	P1049	LL	.	.00272	.	1.00000	.00065-
64	P1050	LL	.	.00236	.	1.00000	.00103-
65	P1051	LL	.	.00205	.	1.00000	.00127-
66	P1052	LL	.	.00178	.	1.00000	.00167-
67	P1053	LL	.	.00155	.	1.00000	.00204-
68	P1054	LL	.	.00135	.	1.00000	.00249-
69	P1055	LL	.	.00117	.	1.00000	.00276-

SOLUTION (OPTIMAL)

TIME = 0.41 MINS. ITERATION NUMBER = 38

...NAME...	...ACTIVITY...	DEFINED AS
FUNCTIONAL	1.12976	ROW101
RESTRAINTS		LIMITS
BOUNDS....		SEPBOUND

SECTION 1 - ROWS

	NUMBER	...ROW..	AT	...ACTIVITY...	SLACK ACTIVITY	..LOWER LIMIT.	..UPPER LIMIT.	.DUAL ACTIVITY
	1	ROW101	BS	1.12976	1.12976-	NONE	NONE	1.00000
	2	ROW102	BS	84.05514	13.94486	NONE	98.00000	.
	3	ROW103	BS	79.69340	22.29840	NONE	101.99200	.
	4	ROW104	UL	151.21800	.	NONE	151.21800	.00048-
A	5	GRID101	EQ	1.00000	.	1.00000	1.00000	.
A	6	GRID102	EQ	1.00000	.	1.00000	1.00000	.
A	7	GRID103	EQ	1.00000	.	1.00000	1.00000	.
A	8	GRID104	EQ	1.00000	.	1.00000	1.00000	.
A	9	GRID105	EQ	1.00000	.	1.00000	1.00000	.

SECTION 2 - COLUMNS

NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
10	X1	BS	2.70000	.	.	NONE	.
11	X2	BS	2.32979	.	.	NONE	.
12	X3	BS	2.10000	.	.	NONE	.
13	X4	BS	3.50000	.	.	NONE	.
14	X5	BS	2.80000	.	.	NONE	.
15	P1C01	UL	1.00000	.18232	.	1.00000	.16857
16	P1C02	UL	1.00000	.06617	.	1.00000	.00445
17	P1C03	UL	1.00000	.00523	.	1.00000	.00343
18	P1C04	UL	1.00000	.00443	.	1.00000	.00256
19	P1C05	UL	1.00000	.00375	.	1.00000	.00180
20	P1C06	UL	1.00000	.00319	.	1.00000	.00116
21	P1C07	UL	1.00000	.00270	.	1.00000	.00058
22	P1C08	UL	1.00000	.00230	.	1.00000	.00010
23	P1C09	LL	.	.00195	.	1.00000	.00034-
24	P1C10	LL	.	.00166	.	1.00000	.00073-
25	P1C11	LL	.	.00141	.	1.00000	.00107-
26	P1012	UL	1.00000	.13976	.	1.00000	.12404
27	P1C13	LL	1.00000	.00397	.	1.00000	.00200
28	P1C14	LL	1.00000	.00327	.	1.00000	.00122
29	P1C15	UL	1.00000	.00270	.	1.00000	.00056
30	P1C16	BS	.29288	.00223	.	1.00000	.
31	P1C17	LL	.	.00184	.	1.00000	.00048-
32	P1C18	LL	.	.00152	.	1.00000	.00090-
33	P1C19	LL	.	.00125	.	1.00000	.00127-
34	P1C20	LL	.	.00104	.	1.00000	.00158-
35	P1C21	LL	.	.00086	.	1.00000	.00187-
36	P1C22	LL	.	.00071	.	1.00000	.00213-
37	P1C23	UL	1.00000	.07323	.	1.00000	.06622
38	P1C24	UL	1.00000	.00669	.	1.00000	.00509
39	P1C25	UL	1.00000	.00529	.	1.00000	.00362
40	P1C26	LL	1.00000	.00419	.	1.00000	.00244
41	P1C27	LL	1.00000	.00331	.	1.00000	.00150
42	P1C28	UL	1.00000	.00262	.	1.00000	.00073
43	P1C29	LL	1.00000	.00208	.	1.00000	.00011
44	P1C30	LL	.	.00165	.	1.00000	.00040-
45	P1C31	LL	.	.00131	.	1.00000	.00083-
46	P1C32	LL	.	.00104	.	1.00000	.00119-
47	P1C33	LL	.	.00082	.	1.00000	.00159-
48	P1C34	UL	1.00000	.38696	.	1.00000	.35730
49	P1C35	UL	1.00000	.00445	.	1.00000	.00224
50	P1C36	UL	1.00000	.00339	.	1.00000	.00169
51	P1C37	UL	1.00000	.00358	.	1.00000	.00119
52	P1C38	UL	1.00000	.00321	.	1.00000	.00072
53	P1C39	LL	1.00000	.00298	.	1.00000	.00030
54	P1C40	LL	.	.00259	.	1.00000	.00009-
55	P1C41	LL	.	.00233	.	1.00000	.00046-
56	P1C42	LL	.	.00209	.	1.00000	.00081-
57	P1C43	LL	.	.00193	.	1.00000	.00113-
58	P1C44	LL	.	.00169	.	1.00000	.00144-

NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
59	P1045	UL	1.00000	.25593	.	1.00000	.22619
60	P1046	UL	1.00000	.00417	.	1.00000	.00145
61	P1047	UL	1.00000	.00361	.	1.00000	.00078
62	P1048	UL	1.00000	.00314	.	1.00000	.00019
63	P1049	LL	.	.00272	.	1.00000	.00045-
64	P1050	LL	.	.00236	.	1.00000	.00083-
65	P1051	LL	.	.00205	.	1.00000	.00127-
66	P1052	LL	.	.00178	.	1.00000	.00167-
67	P1053	LL	.	.00155	.	1.00000	.00204-
68	P1054	LL	.	.00135	.	1.00000	.00238-
69	P1055	LL	.	.00117	.	1.00000	.00270-

SOLUTION (OPTIMAL)

TIME = 0.50 MINS. ITERATION NUMBER = 31

...NAME...	...ACTIVITY...	DEFINED AS
FUNCTIONAL	1.12976	ROW101
RESTRAINTS		LIMITS
BOUNDS....		SEPBOUND

SECTION 1 - ROWS

NUMBER	...ROW...	AT	...ACTIVITY...	SLACK ACTIVITY	..LOWER LIMIT.	..UPPER LIMIT.	.DUAL ACTIVITY
1	ROW101	BS	1.12976	1.12976-	NONE	NONE	1.00000
2	ROW102	BS	84.05514	13.94486	NONE	98.00000	.
3	ROW103	BS	79.69840	22.29840	NONE	101.99200	.
4	ROW104	LL	151.21800	.	NONE	151.21800	.00098-
A 5	GRID101	EQ	1.00000	.	1.00000	1.00000	.
A 6	GRID102	EQ	1.00000	.	1.00000	1.00000	.
A 7	GRID103	EQ	1.00000	.	1.00000	1.00000	.
A 8	GRID104	EQ	1.00000	.	1.00000	1.00000	.
A 9	GRID105	EQ	1.00000	.	1.00000	1.00000	.

SECTION 2 - COLUMNS

NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
10	X1	BS	2.70000	.	.	NONE	.
11	X2	BS	2.32979	.	.	NONE	.
12	X3	BS	2.10000	.	.	NONE	.
13	X4	BS	3.50000	.	.	NONE	.
14	X5	BS	2.80000	.	.	NONE	.
15	P1C01	UL	1.00000	.18232	.	1.00000	.16857
16	P1C02	LL	1.00000	.00417	.	1.00000	.00445
17	P1C03	LL	1.00000	.00523	.	1.00000	.00343
18	P1C04	UL	1.00000	.00443	.	1.00000	.00256
19	P1C05	LL	1.00000	.00375	.	1.00000	.00180
20	P1C06	UL	1.00000	.00319	.	1.00000	.00116
21	P1C07	UL	1.00000	.00270	.	1.00000	.00058
22	P1C08	UL	1.00000	.00230	.	1.00000	.00010
23	P1C09	LL	.	.00195	.	1.00000	.00034-
24	P1C10	LL	.	.00166	.	1.00000	.00073-
25	P1C11	LL	.	.00141	.	1.00000	.00107-
26	P1C12	UL	1.00000	.13976	.	1.00000	.12404
27	P1C13	UL	1.00000	.00397	.	1.00000	.00200
28	P1C14	UL	1.00000	.00327	.	1.00000	.00122
29	P1C15	LL	1.00000	.00270	.	1.00000	.00056
30	P1C16	BS	.29288	.00223	.	1.00000	.
31	P1C17	LL	.	.00184	.	1.00000	.00048-
32	P1C18	LL	.	.00152	.	1.00000	.00090-
33	P1C19	LL	.	.00125	.	1.00000	.00127-
34	P1C20	LL	.	.00104	.	1.00000	.00156-
35	P1C21	LL	.	.00086	.	1.00000	.00187-
36	P1C22	LL	.	.00071	.	1.00000	.00213-
37	P1C23	UL	1.00000	.07323	.	1.00000	.06622
38	P1C24	UL	1.00000	.00669	.	1.00000	.00509
39	P1C25	LL	1.00000	.00529	.	1.00000	.00362
40	P1C26	UL	1.00000	.00418	.	1.00000	.00244
41	P1C27	UL	1.00000	.00331	.	1.00000	.00150
42	P1C28	LL	1.00000	.00262	.	1.00000	.00073
43	P1C29	UL	1.00000	.00208	.	1.00000	.00011
44	P1C30	LL	.	.00165	.	1.00000	.00040-
45	P1C31	LL	.	.00131	.	1.00000	.00083-
46	P1C32	LL	.	.00104	.	1.00000	.00119-
47	P1C33	LL	.	.00082	.	1.00000	.00150-
48	P1C34	LL	1.00000	.38696	.	1.00000	.35730
49	P1C35	UL	1.00000	.00445	.	1.00000	.00284
50	P1C36	UL	1.00000	.00399	.	1.00000	.00169
51	P1C37	LL	1.00000	.00358	.	1.00000	.00119
52	P1C38	LL	1.00000	.00321	.	1.00000	.00072
53	P1C39	UL	1.00000	.00288	.	1.00000	.00030
54	P1C40	LL	.	.00259	.	1.00000	.00059-
55	P1C41	LL	.	.00233	.	1.00000	.00046-
56	P1C42	LL	.	.00209	.	1.00000	.00081-
57	P1C43	LL	.	.00188	.	1.00000	.00113-
58	P1C44	LL	.	.00169	.	1.00000	.00143-

NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
59	P1045	LL	1.00000	.25593	.	1.00000	.22619
60	P1046	UL	1.00000	.00417	.	1.00000	.00145
61	P1047	UL	1.00000	.00361	.	1.00000	.00078
62	P1048	LL	1.00000	.00314	.	1.00000	.00019
63	P1049	LL	.	.00272	.	1.00000	.00035-
64	P1050	LL	.	.00236	.	1.00000	.00083-
65	P1051	LL	.	.00205	.	1.00000	.00127-
66	P1052	LL	.	.00178	.	1.00000	.00167-
67	P1053	LL	.	.00155	.	1.00000	.00204-
68	P1054	LL	.	.00135	.	1.00000	.00238-
69	P1055	LL	.	.00117	.	1.00000	.00270-

APPENDIX 3. GEOMETRIC PROGRAMMING USING MPS/360

This appendix contains the necessary programming material to solve the example geometric programming dual problem. The contents are

1. the main FORTRAN program and its supporting subroutine
2. the job control language
3. sample input data to the FORTRAN program
4. an echo check output from the FORTRAN program
5. the MPS/360 control language program with the solution output

The separable data is read with the same formats as the program of Appendix one.

NFUN = the number of dual function variables including the transformed variables.

NROWS is replaced by NCON.

NCON = the number of primal constraints.

ALOW can never equal zero (because $\ell n(0)$ does not exist) and UPPER ≤ 1.

The data for the variables are ordered as

$w_{01}, w_{02}, \ldots, w_{04}, w_{11}, w_{12}, w_{21}, A_1$, and A_2.

The variable START is set equal to the coefficients

$c_{01}, c_{02}, \ldots, c_{04}, c_{11}, c_{12}, c_{21}$

in the same order as the dual variables. For the transformed variables, A_1 and A_2, it equals 1.0. This procedure is generalized and the FORTRAN program can be used to solve any posynomial geometric programming problem.

The separable data is not punched but is stored in a data-set where it is accessed by the MPS/360 control language program. This eliminates the cumbersome process of punching cards.

```
1000 FORMAT(2I5)
1001 FORMAT(20A4)
2000 FORMAT(4X,1HP,I4,5X,6HOBJECT,4X,F12.8,3X,4HGRID,I3,3X,F12.5)
4000 FORMAT(4X,3HSET,I3,4X,8H'MARKER',17X,8H'SEPORG')
5000 FORMAT(F10.0,I10,2F10.0)
6000 FORMAT(5X,23HECHO CHECK FOR VARIABLE,I5/5X,6HSTART=,F10.5/
    15X,6HINTER=,I10/5X,6HALOW =,F10.5/5X,6HUPPER=,F10.5/)
7000 FORMAT(4X,6HENDSET,4X,8H'MARKER',17X,8H'SEPEND')
7001 FORMAT(6HBOUNDS)
8000 FORMAT(1X,14HUP SEPBOUND  P,I4,9X,3H1.0)
9000 FORMAT(5X,35HTHE NUMBER OF SEPARATED VARIABLES =,I5/
    15X,35HTHE NUMBER OF DEPENDENT VARIABLES =,I5//)
     NREAD=5
     NPRNT=6
     NDISK=9
     DIMENSION A(20)
     DOUBLE PRECISION OBJ,F
     COMMON KK,START,NFC
     DATA SEPA,ENDA/4HSEPA,4HENDA/
     REWIND NDISK
  20 READ(NREAD,1001) A
     IF(A(1).EQ.SEPA) GO TO 21
     WRITE(NDISK,1001) A
     GO TO 20
  21 READ(NREAD,1000) NFUN,NCON
     WRITE(NPRNT,9000) NFUN,NCON
     IF(NCON.GT.NFUN) STOP9999
     NFC=NFUN-NCON
     NKK=0
     KKK=1000
     DO400KK=1,NFUN
     IAX=1
     READ(NREAD,5000) START,INTER,ALOW,UPPER
     WRITE(NPRNT,6000) KK,START,INTER,ALOW,UPPER
     IF(INTER.GT.50)INTER=50
     IF(ALOW.EQ.UPPER)STOP8888
     IF(INTER.EQ.0)STOP7777
     AN=INTER
     DELTA=(UPPER-ALOW)/AN
     LL=KK+100
     WRITE(NDISK,4000) LL
     IJ=0
     IF(ALOW-START)2,3,2
   2 X=START
     AX=ALOW
     IJ=1
     INTER=INTER+1
     IAX=2
     GO TO 8
   3 IJ=IJ+1
     XK=IJ-IAX
     X=XK*DELTA+ALOW
     AX=X+DELTA
   8 CALL FUNC(AX,OBJ)
     CALL FUNC(X,F)
     OBJ=OBJ-F
     GRID=X-AX
     II=IJ+KKK
```

```
      WRITE(NDISK,2000) II,OBJ,LL,GRID
      IF(IJ-INTER)3,301,301
  301 KKK=KKK+INTER
      NKK=NKK+INTER
  400 CONTINUE
      WRITE(NDISK,7000)
   22 READ(NREAD,1001) A
      IF(A(1).EQ.ENDA) GO TO 23
      WRITE(NDISK,1001) A
      GO TO 22
   23 WRITE(NDISK,7001)
      DO1I=1,NKK
      II=I+1000
    1 WRITE(NDISK,8000) II
      WRITE(NDISK,1001) A
      END FILE NDISK
      STOP
      END


      SUBROUTINE FUNC(X,F)
      COMMON KK,START,NFC
      DOUBLE PRECISION SS,XX,F
      XX=X
      IF(KK.GT.NFC) GO TO 1
      SS=START
      F=XX*DLOG(SS/XX)
      RETURN
    1 F=XX*DLOG(XX)
      RETURN
      END
```

```
//          JOB ( standard K.S.U. job card )
//JOBLIB DD DSNAME=SYS1.USERLIB,DISP=SHR
//STEP1  EXEC FORTGCLG
//FORT.SYSIN DD *

            FORTRAN source deck

/*
//GO.FT09F001 DD DSN=&INPUT,UNIT=SYSDA,DISP=(NEW,PASS),
//       SPACE=(TRK,(5,5)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=7280)
//GO.SYSIN DD *

            data

/*
//STEP2  EXEC KSLP
//CPC.SYSIN DD *

            MPS control language program

/*
//EXEC.SYSIN DD DSN=&INPUT,UNIT=SYSDA,DISP=(OLD,PASS)
//
```

```
NAME          DATA-SET
ROWS
  N OBJECT
  E GRID101
  E GRID102
  E GRID103
  E GRID104
  E GRID105
  E GRID106
  E GRID107
  E GRID108
  E GRID109
  E NORM
  E ORTH1
  E ORTH2
  E ORTH3
  E ORTH4
  E CONST1
  E CONST2
COLUMNS
    W1        GRID101      1.0        NORM        1.0
    W1        ORTH1        1.0
    W2        GRID102      1.0        NORM        1.0
    W2        ORTH2        1.0
    W3        GRID103      1.0        NORM        1.0
    W3        ORTH3        1.0
    W4        GRID104      1.0        NORM        1.0
    W4        ORTH4        1.0
    W5        GRID105      1.0        ORTH1       2.0
    W5        ORTH4        -2.0       CONST1      -1.0
    W6        GRID106      1.0        ORTH2       2.0
    W6        ORTH4        -2.0       CONST1      -1.0
    W7        GRID107      1.0        ORTH1       -1.0
    W7        ORTH2        -1.0       ORTH3       -1.0
    W7        CONST2       -1.0
    A1        GRID108      1.0        CONST1      1.0
    A2        GRID109      1.0        CONST2      1.0
SEPARABLE DATA
    9    2
    4.0          20      .228       .238
    10.0         20      .298       .308
    4.0          20      .328       .338
    2.0          20      .125       .135
    1.0          20      .045       .055
    1.0          20      .010       .020
    100.0        20      .328       .338
    1.0          20      .060       .070
    1.0          20      .328       .338
RHS
    LIMITS    GRID101      4.0        GRID102     10.0
    LIMITS    GRID103      4.00       GRID104     2.0
    LIMITS    GRID105      1.0        GRID106     1.0
    LIMITS    GRID107      100.0      GRID108     1.0
    LIMITS    GRID109      1.0        NORM        1.0
ENDATA
```

```
THE NUMBER OF SEPARATED VARIABLES =     9
THE NUMBER OF DEPENDENT VARIABLES =     2


ECHO CHECK FOR VARIABLE     1
START=    4.00000
INTER=          20
ALCW =    0.22800
UPPER=    0.23800

ECHO CHECK FOR VARIABLE     2
START=   10.00000
INTER=          20
ALCW =    0.29800
UPPER=    0.30800

ECHO CHECK FOR VARIABLE     3
START=    4.00000
INTER=          20
ALOW =    0.32800
UPPER=    0.33800

ECHO CHECK FOR VARIABLE     4
START=    2.00000
INTER=          20
ALCW =    0.12500
UPPER=    0.13500

ECHO CHECK FOR VARIABLE     5
START=    1.00000
INTER=          20
ALOW =    0.04500
UPPER=    0.05500

ECHO CHECK FOR VARIABLE     6
START=    1.00000
INTER=          20
ALOW =    0.01000
UPPER=    0.02000

ECHO CHECK FOR VARIABLE     7
START=  100.00000
INTER=          20
ALCW =    0.32800
UPPER=    0.33800

ECHO CHECK FOR VARIABLE     8
START=    1.00000
INTER=          20
ALCW =    0.06000
UPPER=    0.07000

ECHO CHECK FOR VARIABLE     9
START=    1.00000
INTER=          20
ALOW =    0.32800
UPPER=    0.33800
```

CONTROL PROGRAM COMPILER - MPS/360 V2-M8

```
C001                    PROGRAM
C002                    INITIALZ
C064                    MOVE(XDATA,'DATA-SET')
C065                    MOVE(XPBNAME,'EXAMPLE')
C066                    CONVERT
C067                    BCDOUT
C068                    MOVE(XOBJ,'OBJECT')
0069                    MOVE(XRHS,'LIMITS')
C070                    TITLE('GEOMETRIC PROGRAMMING')
C071                    SETUP('BOUND','SEPBOUND','MAX','SCALE')
C072                    PRIMAL
C073                    SOLUTION
C074                    EXIT
C075                    PEND
```

SOLUTION (OPTIMAL)

TIME = 0.84 MINS. ITERATION NUMBER = 125

...NAME...	...ACTIVITY...	DEFINED AS
FUNCTIONAL	4.47719	OBJECT
RESTRAINTS		LIMITS
BOUNDS....		SEPBOUND

SECTION 1 - ROWS

NUMBER	...ROW..	AT	...ACTIVITY...	SLACK ACTIVITY	..LOWER LIMIT.	..UPPER LIMIT.	.DUAL ACTIVITY
1	OBJECT	BS	4.47719	4.47719-	NONE	NONE	1.00000
2	GRID101	EQ	4.00000	.	4.00000	4.00000	1.85042
3	GRID102	EQ	10.00000	.	10.00000	10.00000	2.49042
4	GRID103	EQ	4.00000	.	4.00000	4.00000	1.48504
5	GRID104	EQ	2.00000	.	2.00000	2.00000	1.72960
6	GRID105	EQ	1.00000	.	1.00000	1.00000	1.97104
7	GRID106	EQ	1.00000	.	1.00000	1.00000	3.25104
8	GRID107	EQ	100.00000	.	100.00000	100.00000	4.70370
9	GRID108	EQ	1.00000	.	1.00000	1.00000	1.72940-
10	GRID109	EQ	1.00000	.	1.00000	1.00000	.09886-
11	NORM	EQ	1.00000	.	1.00000	1.00000	3.47691-
12	CRTH1	EQ	.	.	.	.	1.62649
13	CRTH2	EQ	.	.	.	.	.98649
14	CRTH3	EQ	.	.	.	.	1.99187
15	CRTH4	EQ	.	.	.	.	1.74731
16	CONST1	EQ	.	.	.	.	1.72940
17	CONST2	EQ	.	.	.	.	.09886

SECTION 2 - COLUMNS

	NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
	18	W1	BS	.23117	.	.	NONE	.
	19	W2	BS	.30500	.	.	NONE	.
	20	W3	BS	.33333	.	.	NONE	.
	21	W4	BS	.13050	.	.	NONE	.
	22	W5	BS	.05108	.	.	NONE	.
	23	W6	BS	.01417	.	.	NONE	.
	24	W7	BS	.33333	.	.	NONE	.
	25	A1	BS	.06575	.	.	NONE	.
	26	A2	BS	.33333	.	.	NONE	.
	27	P1C01	UL	1.00000	.65315	.	1.00000	7.63294
	28	P1C02	UL	1.00000	.00093	.	1.00000	.00001
	29	P1C03	UL	1.00000	.00093	.	1.00000	.00001
	30	P1C04	UL	1.00000	.00093	.	1.00000	.00000
	31	P1C05	UL	1.00000	.00093	.	1.00000	.00000
	32	P1C06	UL	1.00000	.00093	.	1.00000	.00000
	33	P1C07	UL	1.00000	.00093	.	1.00000	.00000
	34	P1C08	BS	.33333	.00093	.	1.00000	.
	35	P1C09	LL	.	.00092	.	1.00000	.00000-
	36	P1C10	LL	.	.00092	.	1.00000	.00000-
	37	P1011	LL	.	.00092	.	1.00000	.00000-
	38	P1012	LL	.	.00092	.	1.00000	.00000-
	39	P1013	LL	.	.00092	.	1.00000	.00001-
	40	P1014	LL	.	.00092	.	1.00000	.00001-
	41	P1015	LL	.	.00092	.	1.00000	.00001-
	42	P1016	LL	.	.00092	.	1.00000	.00001-
	43	P1017	LL	.	.00092	.	1.00000	.00001-
	44	P1018	LL	.	.00091	.	1.00000	.00001-
	45	P1019	LL	.	.00091	.	1.00000	.00001-
	46	P1020	LL	.	.00091	.	1.00000	.00001-
	47	P1021	LL	.	.00091	.	1.00000	.00001-
	48	P1022	LL	1.00000	1.04695	.	1.00000	25.20900
	49	P1023	LL	1.00000	.00126	.	1.00000	.00001
	50	P1024	UL	1.00000	.00126	.	1.00000	.00001
	51	P1025	LL	1.00000	.00125	.	1.00000	.00001
	52	P1026	LL	1.00000	.00125	.	1.00000	.00001
	53	P1027	UL	1.00000	.00125	.	1.00000	.00001
	54	P1028	LL	1.00000	.00125	.	1.00000	.00001
	55	P1029	LL	1.00000	.00125	.	1.00000	.00001
	56	P1030	UL	1.00000	.00125	.	1.00000	.00001
	57	P1031	LL	1.00000	.00125	.	1.00000	.00000
	58	P1032	LL	1.00000	.00125	.	1.00000	.00000
	59	P1033	UL	1.00000	.00125	.	1.00000	.00000
	60	P1034	LL	1.00000	.00125	.	1.00000	.00000
	61	P1035	LL	1.00000	.00125	.	1.00000	.00000
A	62	P1036	LL	1.00000	.00125	.	1.00000	.
	63	P1037	LL	.	.00124	.	1.00000	.00000-
	64	P1038	LL	.	.00124	.	1.00000	.00000-
	65	P1039	LL	.	.00124	.	1.00000	.00000-
	66	P1040	LL	.	.00124	.	1.00000	.00000-

NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
67	P1C41	LL	.	.00124	.	1.00000	.00000-
68	P1C42	LL	.	.00124	.	1.00000	.00000-
69	P1C43	LL	1.00000	.82034	.	1.00000	6.27341
70	P1C44	UL	1.00000	.00075	.	1.00000	.00001
71	P1C45	LL	1.00000	.00075	.	1.00000	.00001
72	P1C46	LL	1.00000	.00075	.	1.00000	.00001
73	P1C47	UL	1.00000	.00075	.	1.00000	.00001
74	P1C48	UL	1.00000	.00075	.	1.00000	.00001
75	P1C49	LL	1.00000	.00075	.	1.00000	.00001
76	P1C50	LL	1.00000	.00075	.	1.00000	.00001
77	P1C51	LL	1.00000	.00074	.	1.00000	.00000
78	P1C52	UL	1.00000	.00074	.	1.00000	.00000
79	P1C53	UL	1.00000	.00074	.	1.00000	.00000
80	P1C54	BS	.66667	.00074	.	1.00000	.
81	P1C55	LL	.	.00074	.	1.00000	.00000-
82	P1C56	LL	.	.00074	.	1.00000	.00000-
83	P1C57	LL	.	.00074	.	1.00000	.00000-
84	P1C58	LL	.	.00074	.	1.00000	.00000-
85	P1C59	LL	.	.00074	.	1.00000	.00000-
86	P1C60	LL	.	.00074	.	1.00000	.00000-
87	P1C61	LL	.	.00074	.	1.00000	.00001-
88	P1C62	LL	.	.00074	.	1.00000	.00001-
89	P1C63	LL	.	.00074	.	1.00000	.00001-
90	P1C64	LL	1.00000	.34657	.	1.00000	3.58957
91	P1C65	LL	1.00000	.00089	.	1.00000	.00002
92	P1C66	LL	1.00000	.00088	.	1.00000	.00002
93	P1C67	LL	1.00000	.00088	.	1.00000	.00002
94	P1C68	UL	1.00000	.00088	.	1.00000	.00001
95	P1C69	UL	1.00000	.00088	.	1.00000	.00001
96	P1C70	LL	1.00000	.00088	.	1.00000	.00001
97	P1C71	LL	1.00000	.00087	.	1.00000	.00001
98	P1C72	UL	1.00000	.00087	.	1.00000	.00001
99	P1C73	LL	1.00000	.00087	.	1.00000	.00000
100	P1C74	UL	1.00000	.00087	.	1.00000	.00000
101	P1C75	UL	1.00000	.00087	.	1.00000	.00000
102	P1C76	LL	.	.00086	.	1.00000	.00000-
103	P1C77	LL	.	.00086	.	1.00000	.00000-
104	P1C78	LL	.	.00086	.	1.00000	.00000-
105	P1C79	LL	.	.00086	.	1.00000	.00001-
106	P1C80	LL	.	.00086	.	1.00000	.00001-
107	P1C81	LL	.	.00085	.	1.00000	.00001-
108	P1C82	LL	.	.00085	.	1.00000	.00001-
109	P1C83	LL	.	.00085	.	1.00000	.00001-
110	P1C84	LL	.	.00085	.	1.00000	.00001-
111	P1C85	LL	1.00000	.13955	.	1.00000	2.02189
112	P1C86	LL	1.00000	.00105	.	1.00000	.00006
113	P1C87	LL	1.00000	.00104	.	1.00000	.00006
114	P1C88	UL	1.00000	.00104	.	1.00000	.00005
115	P1C89	LL	1.00000	.00103	.	1.00000	.00005
116	P1C90	LL	1.00000	.00103	.	1.00000	.00004
117	P1C91	LL	1.00000	.00102	.	1.00000	.00004

NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
118	P1C92	UL	1.00000	.01C2	.	1.00000	.00003
119	P1C93	LL	1.00000	.00101	.	1.00000	.00003
120	P1C94	LL	1.00000	.00101	.	1.00000	.00002
121	P1C95	UL	1.00000	.01C0	.	1.00000	.00001
122	P1C96	LL	1.00000	.01C0	.	1.00000	.00001
123	P1C97	LL	1.00000	.00099	.	1.00000	.00000
124	P1C98	BS	.16667	.00099	.	1.00000	.
125	P1C99	LL	.	.00098	.	1.00000	.00000-
126	P11C0	LL	.	.00098	.	1.00000	.00001-
127	P1101	LL	.	.00097	.	1.00000	.00001-
128	P1102	LL	.	.00097	.	1.00000	.00002-
129	P1103	LL	.	.00096	.	1.00000	.00002-
130	P1104	LL	.	.00096	.	1.00000	.00003-
131	P1105	LL	.	.00095	.	1.00000	.00003-
132	P1106	UL	1.00000	.04605	.	1.00000	3.26459
133	P1107	LL	1.00000	.00179	.	1.00000	.00016
134	P1108	UL	1.00000	.00177	.	1.00000	.00014
135	P1109	UL	1.00000	.00174	.	1.00000	.00012
136	P1110	LL	1.00000	.00172	.	1.00000	.00010
137	P1111	UL	1.00000	.00170	.	1.00000	.00008
138	P1112	UL	1.00000	.00168	.	1.00000	.00006
139	P1113	LL	1.00000	.00166	.	1.00000	.00004
140	P1114	UL	1.00000	.00164	.	1.00000	.00002
141	P1115	BS	.33333	.00163	.	1.00000	.
142	P1116	LL	.	.00161	.	1.00000	.00002-
143	P1117	LL	.	.00159	.	1.00000	.00003-
144	P1118	LL	.	.00158	.	1.00000	.00005-
145	P1119	LL	.	.00156	.	1.00000	.00007-
146	P1120	LL	.	.00154	.	1.00000	.00008-
147	P1121	LL	.	.00153	.	1.00000	.00010-
148	P1122	LL	.	.00152	.	1.00000	.00011-
149	P1123	LL	.	.00150	.	1.00000	.00012-
150	P1124	LL	.	.00149	.	1.00000	.00014-
151	P1125	LL	.	.00148	.	1.00000	.00015-
152	P1126	LL	.	.00146	.	1.00000	.00016-
153	P1127	LL	1.00000	1.87613	.	1.00000	470.70332
154	P1128	UL	1.00000	.00236	.	1.00000	.00001
155	P1129	LL	1.00000	.00236	.	1.00000	.00001
156	P1130	LL	1.00000	.00236	.	1.00000	.00001
157	P1131	UL	1.00000	.00236	.	1.00000	.00001
158	P1132	UL	1.00000	.00236	.	1.00000	.00000
159	P1133	LL	1.00000	.00236	.	1.00000	.00000
160	P1134	UL	1.00000	.00235	.	1.00000	.00000
161	P1135	UL	1.00000	.00235	.	1.00000	.00000
162	P1136	LL	1.00000	.00235	.	1.00000	.00000
163	P1137	UL	1.00000	.00235	.	1.00000	.00000
164	P1138	BS	.66667	.00235	.	1.00000	.
165	P1139	LL	.	.00235	.	1.00000	.00000-
166	P1140	LL	.	.00235	.	1.00000	.00000-
167	P1141	LL	.	.00235	.	1.00000	.00000-
168	P1142	LL	.	.00235	.	1.00000	.00000-

NUMBER	.COLUMN.	AT	...ACTIVITY...	..INPUT COST..	..LOWER LIMIT.	..UPPER LIMIT.	.REDUCED COST.
169	P1143	LL	.	.00235	.	1.00000	.00000-
170	P1144	LL	.	.00235	.	1.00000	.00000-
171	P1145	LL	.	.00235	.	1.00000	.00001-
172	P1146	LL	.	.00235	.	1.00000	.00001-
173	P1147	LL	.	.00235	.	1.00000	.00001-
174	P1148	UL	1.00000	.16880-	.	1.00000	1.79444-
175	P1149	UL	1.00000	.00090-	.	1.00000	.00004-
176	P1150	UL	1.00000	.00090-	.	1.00000	.00004-
177	P1151	UL	1.00000	.00090-	.	1.00000	.00003-
178	P1152	UL	1.00000	.00089-	.	1.00000	.00003-
179	P1153	LL	1.00000	.00089-	.	1.00000	.00002-
180	P1154	UL	1.00000	.00088-	.	1.00000	.00002-
181	P1155	UL	1.00000	.00088-	.	1.00000	.00002-
182	P1156	LL	1.00000	.00088-	.	1.00000	.00001-
183	P1157	UL	1.00000	.00087-	.	1.00000	.00001-
184	P1158	UL	1.00000	.00087-	.	1.00000	.00000-
185	P1159	BS	.50000	.00086-	.	1.00000	.
186	P1160	LL	.	.00086-	.	1.00000	.00000
187	P1161	LL	.	.00086-	.	1.00000	.00001
188	P1162	LL	.	.00085-	.	1.00000	.00001
189	P1163	LL	.	.00085-	.	1.00000	.00002
190	P1164	LL	.	.00085-	.	1.00000	.00002
191	P1165	LL	.	.00084-	.	1.00000	.00002
192	P1166	LL	.	.00084-	.	1.00000	.00003
193	P1167	LL	.	.00083-	.	1.00000	.00003
194	P1168	LL	.	.00083-	.	1.00000	.00003
195	P1169	UL	1.00000	.36564-	.	1.00000	.43207-
196	P1170	UL	1.00000	.00006-	.	1.00000	.00001-
197	P1171	UL	1.00000	.00006-	.	1.00000	.00001-
198	P1172	UL	1.00000	.00006-	.	1.00000	.00001-
199	P1173	LL	1.00000	.00005-	.	1.00000	.00001-
200	P1174	UL	1.00000	.00005-	.	1.00000	.00000-
201	P1175	UL	1.00000	.00005-	.	1.00000	.00000-
202	P1176	UL	1.00000	.00005-	.	1.00000	.00000-
203	P1177	UL	1.00000	.00005-	.	1.00000	.00000-
204	P1178	UL	1.00000	.00005-	.	1.00000	.00000-
205	P1179	LL	1.00000	.00005-	.	1.00000	.00000-
206	P1180	BS	.66667	.00005-	.	1.00000	.
207	P1181	LL	.	.00005-	.	1.00000	.00000
208	P1182	LL	.	.00005-	.	1.00000	.00000
209	P1183	LL	.	.00005-	.	1.00000	.00000
210	P1184	LL	.	.00005-	.	1.00000	.00000
211	P1185	LL	.	.00005-	.	1.00000	.00000
212	P1186	LL	.	.00004-	.	1.00000	.00000
213	P1187	LL	.	.00004-	.	1.00000	.00001
214	P1188	LL	.	.00004-	.	1.00000	.00001
215	P1189	LL	.	.00004-	.	1.00000	.00001

APPENDIX 4. GREG PROGRAMMING MATERIAL FOR CHAPTER FIVE

This appendix contains listings of the external subroutines and the resulting output for the example problems of Chapter five.

```
C001              SUBROUTINE PHIX
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)   GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)          GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                              GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                         GEGA 100
C007              PHI=2.0*XC(1)-0.5*XC(1)**2+3.0*XC(2)-0.5*XC(2)**2
0008              RETURN
0009              END
```

```
C001              SUBROUTINE CPHI
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)   GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)          GEGA  30
0003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                              GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                         GEGA 100
C007              VC(1)=XC(1)**2+XC(2)**2-1.0
C008              RETURN
C009              END
```

```
C001              SUBROUTINE JACOB
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
C007              A(1,1)=2.0*XC(1)
C008              A(1,2)=2.0*XC(2)
C009              RETURN
C010              END
```

```
C001              SUBROUTINE GRADFI
0002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
0007              C(1)=2.0-XC(1)
C008              C(2)=3.0-XC(2)
C009              RETURN
C010              END
```

THE PAGE NUMBER ON THE FOLLOWING PAGE IS ILLEGIBLE DUE TO BEING CUT OFF.

THIS IS AS RECEIVED FROM CUSTOMER.

PARAMETRES

NV	2
NIN	1
NEG	0
NEVL	20
NTD	6
ITET	20
IDONJ	0
IDIAG	0
ITMAX	20
KFIL	0
KLIN	0
NCO	10
ITSOR	1
ISOLSR	20
EPSIL	0.1E 00
EPSILO	0.1E-06
EPSIL1	0.1E-04
EPSIL2	0.1E-04
PC	0.1E 01

GRADIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES	2
NOMBRE TOTAL DE VARIABLES	3
NOMBRE DE CONTRAINTES	1

EPSILON DE NEWTON	0.1000E-06
EPSILON TEST GRADIENT	0.1000E-04

FONCTION ECONOMIQUE 0.225000000000000E 01

BORNE INFERIEURE		VARIABLE NATURELLE		BORNE SUPERIEURE	
XI(1)	0.0	X(1)	0.500000000000000E 00	XS(1)	0.100000000000000E 02
XI(2)	0.0	X(2)	0.500000000000000E 00	XS(2)	0.100000000000000E 02

VALEUR DES CONTRAINTES

C(1) -0.500000000000000E 00

```
IT  1   PHI 0.276464843750000E 01   DIR.GRAD.  NO  0   YN 0.29E 01   DELTAFI 0.38E 02   ETA 0.24E 02   NCDB  0   NCN  2   NITN  2
IT  2   PHI 0.293035984039316E 01   DIR.GRAD.  NO  0   YN 0.27E 01   DELTAFI 0.35E 02   ETA 0.22E 02   NCDB  0   NCN  2   NITN  2
IT  3   PHI 0.300653076171886E 01   DIR.GRAD.  NO  0   YN 0.27E 01   DELTAFI 0.34E 02   ETA 0.21E 02   NCDB  0   NCN  2   NITN  2
IT  4   PHI 0.307740783691416E 01   DIR.GRAD.  NO  0   YN 0.26E 01   DELTAFI 0.34E 02   ETA 0.21E 02   NCDB  1   NCN  3   NITN 11
IT  5   PHI 0.310554504394536E 01   DIR.GRAD.  NO  1   YN 0.59E 00   DELTAFI 0.39E 00   ETA 0.39E 00   NCDB  0   NCN  4   NITN 23
IT  6   PHI 0.310554981231696E 01   DIR.GRAD.  NO  1   YN 0.63E-02   DELTAFI 0.59E-01   ETA 0.59E-01   NCDB  0   NCN  3   NITN  3
IT  7   PHI 0.310555076599126E 01   DIR.GRAD.  NO  1   YN 0.19E-02   DELTAFI 0.10E-02   ETA 0.10E-02   NCDB  0   NCN  3   NITN  1
        PHI 0.310555076599126E 01                      YN 0.57E-03   DELTAFI 0.10E-02   ETA 0.54E-02


DUREE  DU  CALCUL     50  CENTISECONDES

          VARIABLE  NATURELLE                          VARIABLE  DUALE  ASSOCIEE

X(  1) =  0.554591953754436 00                  V(  1) =  0.566482543945316-03
X(  2) =  0.832122802734386 00                  V(  2) =  0.


               CONTRAINTE                          VARIABLE  DUALE  ASSOCIEE

CONTRAINTE  1        0.0                        U(  1) =  0.130261802673346 01


NIVEAU  DE  SORTIE      8

NOMBRE  D ITERATIONS         7

NOMBRE  D APPELS  DU  S/P  CALCULANT  LES  CONTRAINTES    61

NOMBRE  D APPELS  DU  S/P  CALCULANT  LA  FONCTION  ECONOMIQUE    85

NOMBRE  D APPELS  DU  S/P  CALCULANT  LE  GRADIENT  DE  LA  FONCTION  ECONOMIQUE    14

NOMBRE  D APPELS  DU  S/P  CALCULANT  L INVERSE  AVEC  BASE  DONNEE      5
NOMBRE  D APPELS  DU  S/P  CALCULANT  L INVERSE  AVEC  RECHERCHE  DE  BASE      1

NOMBRE  D APPELS  DU  S/P  EFFECTUANT  LES  CHANGEMENTS  DE  BASE      1

NOMBRE  D APPELS  DU  S/P  CALCULANT  LE  JACOBIEN  DES  CONTRAINTES      9

NOMBRE  D APPELS  DU  S/P  EFFECTUANT  UNE  RECHERCHE  DE  MAX  SANS  DICHOTOMIE      7
NOMBRE  D APPELS  DU  S/P  EFFECTUANT  UNE  RECHERCHE  DE  MAX  AVEC  DICHOTOMIE      3

NOMBRE  D APPELS  DU  S/P  EFFECTUANT  LA  RENTREE  DANS  LE  DOMAINE    19          NOMBRE  TOTAL  D ITERATIONS    44
```

```
C001              SUBROUTINE PHIX
C002              DIMENSION  A(50,100),ALFA(50,30),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                               GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
C007              XX1=1.0-XC(1)
C008              XX3=1.0-XC(3)
C009              XX4=1.0-XC(4)
C010              PHI=1.0-XC(3)*(XX1*XX4)**2-XX3*(1.0-XC(2)*(1.0-XX1*XX4))**2
C011              RETURN
0012              END
```

```
C001              SUBROUTINE CPHI
C002              DIMENSION  A(50,100),ALFA(50,30),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
0003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                               GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
C007              VC(1)=200.0*XC(1)**0.6+200.0*XC(2)**0.6+200.0*XC(3)**0.6
                 1+300.0*XC(4)**0.6-800.0
C008              RETURN
C009              END
```

```
C001              SUBROUTINE JACOB
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
C005              COMMON     KFONC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                               GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
C007              A(1,1)=120.0*XC(1)**(-0.4)
C008              A(1,2)=120.0*XC(2)**(-0.4)
C009              A(1,3)=120.0*XC(3)**(-0.4)
C010              A(1,4)=180.0*XC(4)**(-0.4)
C011              RETURN
C012              END
```

```
C001              SUBROUTINE GRADFI
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
C005              COMMON     KFONC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                               GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
C007              XX1=1.0-XC(1)
C008              XX3=1.0-XC(3)
C009              XX4=1.0-XC(4)
C010              C(1)=2.0*XC(3)*XX1*XX4*XX4+2.0*XX3*(1.0-XC(2)*(1.0-XX1*XX4))*XC(2)
                 1*XX4
C011              C(2)=2.0*XX3*(1.0-XC(2)*(1.0-XX1*XX4))*(1.0-XX1*XX4)
C012              C(3)=-(XX1*XX4)**2+(1.0-XC(2)*(1.0-XX1*XX4))**2
C013              C(4)=2.0*XC(3)*XX1*XX4*XX1+2.0*XX3*(1.0-XC(2)*(1.0-XX1*XX4))*XC(2)
                 1*XX1
C014              RETURN
C015              END
```

PARAMETRES

NV	4
NIN	1
NES	0
NEVL	30
NTS	6
ITET	20
ICONJ	1
ICIAG	0
ITMAX	50
KFIL	0
KLIN	0
NCC	10
ITSCP	1
ISOLSP	50
EPSIL	0.1E 00
EPSILO	0.1E-05
EPSIL1	0.1E-05
EPSIL2	0.1E-05
PC	0.1E 01

GRADIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES	4
NOMBRE TOTAL DE VARIABLES	5
NOMBRE DE CONTRAINTES	1

EPSILON DE VENTOU	0.1000E-05
EPSILON TEST GRADIENT	0.1000E-05

FONCTION ECONOMIQUE 0.686233627796176 00

BORNE INFERIEURE		VARIABLE NATURELLE		BORNE SUPERIEURE	
XI(1)	0.500000000000000E 00	X(1)	0.599999964237216 00	XS(1)	0.100000000000000E 01
XI(2)	0.500000000000000E 00	X(2)	0.599999964237216 00	XS(2)	0.100000000000000E 01
XI(3)	0.500000000000000E 00	X(3)	0.599999964237216 00	XS(3)	0.100000000000000E 01
XI(4)	0.500000000000000E 00	X(4)	0.599999964237216 00	XS(4)	0.100000000000000E 01

VALEUR DES CONTRAINTES

C(1) -0.137580564402[illegible]E 03

```
IT  1   PHI 0.994009577448085 00   DIR.GRAD.  NO  0   YN 0.47E 00   DELTAFI 0.36E 00   ETA 0.13E 00   NCOB  0   NCN  2   NITN  2
IT  2   PHI 0.995929419994355 00   DIR.CONJ.  LC  3                                                    NCOB  0   NCN  2   NITN  2
IT  3   PHI 0.995436095714576 00   DIR.CONJ.  LC  4                                                    NCOB  0   NCN  2   NITN  2
IT  4   PHI 0.993671328547760E 00   ITERATION  SPECIALE  BASE  DEGENEREE                              NCOB  1   NCN  1   NITN  3
IT  5   PHI 0.100000CCC2000C0E 01   DIR.GRAD.  NO  2   YN 0.21E-01   DELTAFI 0.81E-02   ETA 0.52E-02   NCOB  0   NCN  1   NITN  3
        PHI 0.100000000000000E 01                      YN 0.0        DELTAFI 0.0        ETA 0.0

DUREE DU CALCUL      55 CENTISECONDES

          VARIABLE NATURELLE                                    VARIABLE DUALE ASSOCIEE

X(  1) =  0.100000000000000E 01 = XS(  1)                  V(  1) =  0.0
X(  2) =  0.100000000000000E 01 = XS(  2)                  V(  2) =  0.0
X(  3) =  0.533447307566675E 00                            V(  3) =  0.0
X(  4) =  0.797416865F2565E 00                             V(  4) =  0.

               CONTRAINTE                                   VARIABLE DUALE ASSOCIEE

CONTRAINTE  1       0.0                                    U(  1) =  0.0

NIVEAU DE SORTIE      1
NOMBRE D ITERATIONS          5
NOMBRE D APPELS DU S/P  CALCULANT  LES CONTRAINTES    17
NOMBRE D APPELS DU S/P  CALCULANT  LA  FONCTION  ECONOMIQUE     19
NOMBRE D APPELS DU S/P  CALCULANT  LE  GRADIENT  DE  LA  FONCTION  ECONOMIQUE      8
NOMBRE D APPELS DU S/P  CALCULANT  L INVERSE AVEC BASE DONNEE      2
NOMBRE D APPELS DU S/P  CALCULANT  L INVERSE AVEC RECHERCHE DE BASE      0
NOMBRE D APPELS DU S/P  EFFECTUANT  LES CHANGEMENTS DE BASE      1
NOMBRE D APPELS DU S/P  CALCULANT  LA  JACOBIENNE DES CONTRAINTES      7
NOMBRE D APPELS DU S/P  EFFECTUANT  UNE RECHERCHE DE MAX SANS DISCONTINUITE      5
NOMBRE D APPELS DU S/P  EFFECTUANT  UNE RECHERCHE DE MAX AVEC DISCONTINUITE      0
NOMBRE D APPELS DU S/P  EFFECTUANT  LA  PROJECTION DANS LE DOMAINE      4                    NOMBRE TOTAL D ITERATIONS     5
```

```
C001              SUBROUTINE PHIX
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
0003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDTREC,DELTFI,ETA,JKO,LC,YSORT                        GEGA 100
C007              PHI=200.0*XC(1)**0.6+200.0*XC(2)**0.6+200.0*XC(3)**0.6
                 1+300.0*XC(4)**0.6
C008              PHI=-PHI
C009              RETURN
C010              END
```

```
C001              SUBROUTINE CPHI
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDTREC,DELTFI,ETA,JKO,LC,YSORT                        GEGA 100
C007              XX1=1.0-XC(1)
C008              XX3=1.0-XC(3)
C009              XX4=1.0-XC(4)
C010              VC(1)=1.0-XC(3)*(XX1*XX4)**2-XX3*(1.0-XC(2)*(1.0-XX1*XX4))**2
C011              VC(1)=-VC(1)+0.9
C012              RETURN
C013              END
```

```
C001              SUBROUTINE JACOB
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDTREC,DELTFI,ETA,JK0,LC,YSORT                        GEGA 100
C007              XX1=1.0-XC(1)
C008              XX3=1.0-XC(3)
C009              XX4=1.0-XC(4)
C010              A(1,1)=2.0*XC(3)*XX1*XX4*XX4+2.0*XX3*(1.0-XC(2)*(1.0-XX1*XX4))*XC(
                 12)*XX4
0011              A(1,1)=-A(1,1)
C012              A(1,2)=2.0*XX3*(1.0-XC(2)*(1.0-XX1*XX4))*(1.0-XX1*XX4)
C013              A(1,2)=-A(1,2)
C014              A(1,3)=-(XX1*XX4)**2+(1.0-XC(2)*(1.0-XX1*XX4))**2
C015              A(1,3)=-A(1,3)
C016              A(1,4)=2.0*XC(3)*XX1*XX4*XX1+2.0*XX3*(1.0-XC(2)*(1.0-XX1*XX4))*XC(
                 12)*XX1
0017              A(1,4)=-A(1,4)
0018              RETURN
C019              END


C001              SUBROUTINE GRADFI
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDTREC,DELTFI,ETA,JK0,LC,YSORT                        GEGA 100
C007              C(1)=120.0*XC(1)**(-0.4)
C008              C(1)=-C(1)
C009              C(2)=120.0*XC(2)**(-0.4)
C010              C(2)=-C(2)
C011              C(3)=120.0*XC(3)**(-0.4)
C012              C(3)=-C(3)
C013              C(4)=180.0*XC(4)**(-0.4)
C014              C(4)=-C(4)
C015              RETURN
C016              END
```

PARAMETRES

IV	4
IIV	1
NFS	0
IEVL	29
ITO	4
ITFT	20
ICONJ	1
IPIAS	0
ITMAX	50
KFIL	0
KLIS	0
ICO	10
ITSOP	1
ISOLSP	50
EPSIL	0.1E-06
EPSIL0	0.1E-04
EPSIL1	0.1E-06
EPSIL2	0.1E-06
PC	0.0

GRADIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES 4
NOMBRE TOTAL DE VARIABLES 5
NOMBRE DE CONTRAINTES 1

EPSILON DE VERIFS 0.100E-06
EPSILON TEST GRADIENT 0.100E-06

FONCTION ECONOMIQUE -0.776609619140632 03

	BORNE INFERIEURE	VARIABLE NATURELLE	BORNE SUPERIEURE
XI(1)	0.500000000E-06	X(1)	0.699999998807907E 00
XI(2)	0.500000000E-06	X(2)	0.699999998807907E 00
XI(3)	0.500000000E-06	X(3)	0.699999998807907E 00
XI(4)	0.500000000E-06	X(4)	0.699999998807907E 00

VALEURS DES CONTRAINTES

C(1) -0.547999818313E-01

```
IT  1   PHI=0.[illegible]E 03   DIR.CONJ.  LG  2   YA 0.32E 06   DELTAFI 0.12E 03   ETA 0.44E 02   NCOR  2   NCA  1   NITN  2
IT  2   PHI=0.[illegible]E 03   DIR.CONJ.  LG  4                                                    NCOR  3   NCA  1   NITN  2
IT  3   PHI=0.[illegible]E 03   DIR.CONJ.  LG  4                                                    NCOR  3   NCA  1   NITN  2
IT  4   PHI=0.[illegible]E 03   DIR.CONJ.  LG  4                                                    NCOR  1   NCA  6   NITN 30
        PHI=0.[illegible]E 03                    YA 0.0        DELTAFI 0.0        ETA 0.0

DUREE  DU  CALCUL     41  CENTISECONDES

          VARIABLE  PRIMELLE                          VARIABLE  DUALE  ASSOCIEE

X(  1) =  0.5000000000000000E 00 = XI(  1)          V(  1) = -0.28481002807617E 02
X(  2) =  0.8399202952384XE 00                      V(  2) =  0.
X(  3) =  0.5000000000000000E 00 = XI(  3)          V(  3) = -0.123624023437500E 03
X(  4) =  0.5000000000000000E 00 = XI(  4)          V(  4) = -0.10765145874023E 03

             CONTRAINTE                             VARIABLE  DUALE  ASSOCIEE

CONTRAINTE  1      -0.11920929955078E-06            U(  1) =  0.462892922265630E 03

NIVEAU  DE  SORTIE     1

NOMBRE  D ITERATIONS       4

NOMBRE  D APPELS  OU  S/P  CALCULANT  LES  CONTRAINTES    48

NOMBRE  D APPELS  OU  S/P  CALCULANT  LA  FONCTION  ECONOMIQUE    30

NOMBRE  D APPELS  OL  S/P  CALCULANT  LE  GRADIENT  DE  LA  FONCTION  ECONOMIQUE    12

NOMBRE  D APPELS  OU  S/P  CALCULANT  L INVERSE  AVEC  BASE  DONNEE     2
NOMBRE  D APPELS  OU  S/P  CALCULANT  L INVERSE  AVEC  RECHERCHE  DE  BASE     0

NOMBRE  D APPELS  OL  S/P  EFFECTUANT  LES  CHANGEMENTS  DE  BASE     1

NOMBRE  D APPELS  OL  S/P  CALCULANT  LE  JACOBIEN  DES  CONTRAINTES     7

NOMBRE  D APPELS  OU  S/P  EFFECTUANT  UNE  RECHERCHE  DE  MAX  SANS  DICHOTOMIE     4
NOMBRE  D APPELS  OL  S/P  EFFECTUANT  UNE  RECHERCHE  DE  MAX  AVEC  DICHOTOMIE     0

NOMBRE  D APPELS  OU  S/P  EFFECTUANT  LA  PENTREE  DANS  LE  DOMAINE    14          NOMBRE  TOTAL  D ITERATIONS    45
```

```
C001              SUBROUTINE PHIX
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)   GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)          GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2CB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
C005              COMMON     KFONC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                              GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
C007              COMMON B(5),CC(3,5)
C008              IF(IT)100,101,101
C009          100 B(1)=ALOG(0.2)
C010              B(2)=ALOG(0.15)
C011              B(3)=ALOG(0.1)
C012              B(4)=ALOG(0.35)
C013              B(5)=ALOG(0.25)
C014          101 PHI=0.0
C015              DO102I=1,5
C016          102 PHI=PHI+ALOG(1.0-EXP(B(I)*XC(I)))
C017              RETURN
C018              END
```

```
C001              SUBROUTINE CPHI
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFCNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                        GEGA 100
C007              COMMON B(5),CC(3,5)
C008              IF(IT)100,101,101
C009         100  CC(1,1)=1.0
C010              CC(2,1)=7.0
C011              CC(3,1)=7.0
C012              CC(1,2)=2.0
C013              CC(2,2)=7.0
C014              CC(3,2)=8.0
C015              CC(1,3)=3.0
C016              CC(2,3)=5.0
C017              CC(3,3)=8.0
C018              CC(1,4)=4.0
C019              CC(2,4)=9.0
C020              CC(3,4)=6.0
C021              CC(1,5)=2.0
C022              CC(2,5)=4.0
C023              CC(3,5)=9.0
C024         101  VC(1)=0.0
C025              DO102J=1,5
C026         102  VC(1)=VC(1)+CC(1,J)*XC(J)**2
C027              VC(1)=VC(1)-110.0
C028              VC(2)=0.0
C029              DO103J=1,5
C030         103  VC(2)=VC(2)+CC(2,J)*(XC(J)+EXP(XC(J)/4.0))
C031              VC(2)=VC(2)-175.0
C032              VC(3)=0.0
C033              DO104J=1,5
C034         104  VC(3)=VC(3)+CC(3,J)*XC(J)*EXP(XC(J)/4.0)
C035              VC(3)=VC(3)-200.0
C036              RETURN
C037              END
```

```
C001              SUBROUTINE JACOB
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                        GEGA 100
C007              COMMON B(5),CC(3,5)
C008              DO101J=1,5
C009          101 A(1,J)=2.0*CC(1,J)*XC(J)
C010              DO102J=1,5
C011          102 A(2,J)=CC(2,J)*(1.0+0.25*EXP(XC(J)/4.0))
C012              DO103J=1,5
C013          103 A(3,J)=CC(3,J)*EXP(XC(J)/4.0)*(1.0+0.25*XC(J))
C014              RETURN
C015              END
```

```
C001              SUBROUTINE GRADFI
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                        GEGA 100
C007              COMMON B(5),CC(3,5)
C008              DO101J=1,5
C009              AA=EXP(B(J)*XC(J))
C010          101 C(J)=B(J)*AA/(AA-1.0)
C011              RETURN
C012              END
```

PARAMETRES

NV	5
NIN	3
NEG	0
NEVL	20
NTO	6
ITET	20
ICONJ	1
ICIAG	0
ITMAX	50
KFIL	0
KLIN	0
NCO	10
ITSCR	1
ISOLSR	50
EPSIL	0.1E 00
EPSIL0	0.1E-04
EPSIL1	0.1E-02
EPSIL2	0.1E-02
PC	0.0

GRADIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES	5
NOMBRE TOTAL DE VARIABLES	8
NOMBRE DE CONTRAINTES	3

EPSILON DE NEWTON	0.1000E-04
EPSILON TEST GRADIENT	0.1000E-02

FONCTION ECONOMIQUE -0.268846094660831E 00

BORNE INFERIEURE		VARIABLE NATURELLE		BORNE SUPERIEURE	
XI(1)	0.100000000000000E 01	X(1)	0.200000000000000E 01	XS(1)	0.500000000000000E 01
XI(2)	0.100000000000000E 01	X(2)	0.200000000000000E 01	XS(2)	0.500000000000000E 01
XI(3)	0.100000000000000E 01	X(3)	0.200000000000000E 01	XS(3)	0.500000000000000E 01
XI(4)	0.100000000000000E 01	X(4)	0.200000000000000E 01	XS(4)	0.500000000000000E 01
XI(5)	0.100000000000000E 01	X(5)	0.200000000000000E 01	XS(5)	0.500000000000000E 01

VALEUR DES CONTRAINTES

C(1)	-0.620000000000000E 02
C(2)	-0.582409362792978E 02
C(3)	-0.746971740722666E 02

```
IT  1   PHI=0.12854999C3C3818E 00   DIR.GRAC.  NO  0   YN 0.19E 00   DELTAFI 0.11E 01   ETA 0.44E 00   NCOB  0   NCN  2   NITN  2
IT  2   PHI=0.10524564981461E 00   DIR.CONJ.  LC  5                                                      NCOB  0   NCN  2   NITN  2
IT  3   PHI=0.9119C5169487C0E-01   DIR.CONJ.  LC  5                                                      NCOB  0   NCN  2   NITN  2
IT  4   PHI=0.818144C8302307E-01   DIR.CONJ.  LC  5                                                      NCOB  1   NCN  3   NITN 13
IT  5   PHI=0.809409C2233124E-01   DIR.GRAC.  NO  1   YN 0.15E-01   DELTAFI 0.38E-01   ETA 0.13E-01   NCOB  0   NCN  4   NITN 24
IT  6   PHI=0.7961887121200C6E-01   DIR.GRAC.  NO  1   YN C.11E-01   DELTAFI 0.27E-01   ETA 0.12E-01   NCOB  0   NCN  2   NITN  5
IT  7   PHI=0.79601645469666E-01   DIR.CONJ.  LC  4                                                      NCOB  0   NCN  2   NITN 12
IT  8   PHI=0.79601526260376E-01   DIR.GRAC.  NO  1   YN 0.82E-03   DELTAFI 0.27E-02   ETA 0.10E-02   NCOB  0   NCN  5   NITN 25
IT  9   PHI=0.7959945S702454E-01   DIR.GRAC.  NO  1   YN 0.82E-03   DELTAFI 0.27E-02   ETA 0.10E-02   NCOB  0   NCN  2   NITN  2
        PHI=0.7959945S702454E-01                      YN C.30E-03   DELTAFI 0.12E-02   ETA 0.55E-03
```

DUREE DU CALCUL 286 CENTISECONDES

VARIABLE NATURELLE		VARIABLE DUALE ASSOCIEE	
X(1) =	0.267739568176276 01	V(1) =	-0.22783875465393E-04
X(2) =	C.235254383C8716E 01	V(2) =	0.13865530490875E-03
X(3) =	0.2070096C159302E 01	V(3) =	0.18708407878876E-03
X(4) =	0.353063106536876 01	V(4) =	0.195067375898362-03
X(5) =	C.279202461242686 01	V(5) =	0.

CONTRAINTE		VARIABLE DUALE ASSOCIEE	
CONTRAINTE 1	-C.134513397216P0E 02	U(1) =	0.0
CONTRAINTE 2	-C.2203C639648438E 02	U(2) =	0.0
CONTRAINTE 3	-C.1525878906250E-04	U(3) =	0.960985664278276-03

```
NIVEAU DE SORTIE     7

NOMBRE D ITERATIONS        9

NOMBRE D APPELS DU S/P CALCULANT LES CONTRAINTES   147

NOMBRE D APPELS DU S/P CALCULANT LA FONCTION ECONOMIQUE   110

NOMBRE D APPELS DU S/P CALCULANT LE GRADIENT DE LA FONCTION ECONOMIQUE    15

NOMBRE D APPELS DU S/P CALCULANT L INVERSE AVEC BASE DONNEE     6
NOMBRE D APPELS DU S/P CALCULANT L INVERSE AVEC RECHERCHE DE BASE     1

NOMBRE D APPELS DU S/P EFFECTUANT LES CHANGEMENTS DE BASE     1

NOMBRE D APPELS DU S/P CALCULANT LE JACOBIEN DES CONTRAINTES    11

NOMBRE D APPELS DU S/P EFFECTUANT UNE RECHERCHE DE MAX SANS DICHOTOMIE    12
NOMBRE D APPELS DU S/P EFFECTUANT UNE RECHERCHE DE MAX AVEC DICHOTOMIE     3

NOMBRE D APPELS DU S/P EFFECTUANT LA RENTREE DANS LE DOMAINE    31                NOMBRE TOTAL D ITERATIONS   130
```

```
      SUBROUTINE  PHIX                                                      GEGK  10
      DIMENSION   A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)      GEGK  20
     1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)              GEGK  30
      COMMON      A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB             GEGK  40
      COMMON      NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGK  50
     1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGK  60
     2DB,KFIL,KLIN,KREN,KO,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGK  70
      COMMON      KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGK  80
     1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                                  GEGK  90
      COMMON      IDIREC,DELTFI,ETA,JKO,LC,YSORT                            GEGK 100
      DIMENSION   B(10)                                                     GEGK 110
      COMMON      B                                                         GEGK 120
      IF (IT.GE.0) GO TO 100                                                GEGK 130
      READ (5,1001) (B(I),I=1,10)                                           GEGK 140
 1001 FORMAT (10F6.0)                                                       GEGK 150
  100 CONTINUE                                                              GEGK 160
      PHI = -8.*XC(1)**3-16.*XC(2)**3-20.*XC(3)**3-12.*XC(4)**3-4.*XC(5)GEGK 170
     1**3-30.*XC(1)**2-39.*XC(2)**2-10.*XC(3)**2-39.*XC(4)**2-30.*XC(5)*GEGK 180
     2*2+40.*XC(1)*XC(2)+20.*XC(1)*XC(3)-64.*XC(1)*XC(4)+20.*XC(1)*XC(5)GEGK 190
     3+12.*XC(2)*XC(3)+62.*XC(2)*XC(4)-64.*XC(2)*XC(5)+12.*XC(3)*XC(4)+2GEGK 200
     40.*XC(3)*XC(5)+40.*XC(4)*XC(5)                                        GEGK 210
      DO 20 I=6,15                                                          GEGK 220
      PHI = PHI+B(I-5)*XC(I)                                                GEGK 230
   20 CONTINUE                                                              GEGK 240
      RETURN                                                                GEGK 250
      END                                                                   GEGK 260
```

```
      SUBROUTINE  CPHI                                                      GEGL  10
      DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)       GEGL  20
     1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)              GEGL  30
      COMMON      A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB             GEGL  40
      COMMON      NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGL  50
     1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGL  60
     2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGL  70
      COMMON      KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGL  80
     1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                                  GEGL  90
      COMMON      IDIREC,DELTFI,ETA,JKO,LC,YSORT                            GEGL 100
      IF (IT.GE.0) GO TO 103                                                GEGL 110
      READ (5,1001) (I,J,A(I,J),L=1,37)                                     GEGL 120
 1001 FORMAT      (6(2I3,F6.0)/6(2I3,F6.0)/6(2I3,F6.0)/6(2I3,F6.0)/6(2I3,GEGL 130
     1F6.0)/6(2I3,F6.0)/6(2I3,F6.0))                                        GEGL 140
      A(4,16) = 1.                                                          GEGL 150
      A(5,17) = 1.                                                          GEGL 160
      A(1,2) = 40.                                                          GEGL 170
      A(1,3) = 20.                                                          GEGL 180
      A(1,4) = -64.                                                         GEGL 190
      A(1,5) = 20.                                                          GEGL 200
      A(2,1) = 40.                                                          GEGL 210
      A(2,3) = 12.                                                          GEGL 220
      A(2,4) = 62.                                                          GEGL 230
      A(2,5) = -64.                                                         GEGL 240
      A(3,1) = 20.                                                          GEGL 250
      A(3,2) = 12.                                                          GEGL 260
      A(3,4) = 12.                                                          GEGL 270
      A(3,5) = 20.                                                          GEGL 280
      A(4,1) = -64.                                                         GEGL 290
      A(4,2) = 62.                                                          GEGL 300
      A(4,3) = 12.                                                          GEGL 310
      A(4,5) = 40.                                                          GEGL 320
      A(5,1) = 20.                                                          GEGL 330
      A(5,2) = -64.                                                         GEGL 340
      A(5,3) = 20.                                                          GEGL 350
      A(5,4) = 40.                                                          GEGL 360
  103 CONTINUE                                                              GEGL 370
      VC(1) = 15.-60.*XC(1)+40.*XC(2)+20.*XC(3)-64.*XC(4)+20.*XC(5)-12.*GEGL 380
     1XC(1)**2                                                              GEGL 390
      VC(2) = 27.+40.*XC(1)-78.*XC(2)+12.*XC(3)+62.*XC(4)-64.*XC(5)-24.*GEGL 400
     1XC(2)**2                                                              GEGL 410
      VC(3) = 36.+20.*XC(1)+12.*XC(2)-20.*XC(3)+12.*XC(4)+20.*XC(5)-30.*GEGL 420
     1XC(3)**2                                                              GEGL 430
      VC(4) = 18.-64.*XC(1)+62.*XC(2)+12.*XC(3)-78.*XC(4)+40.*XC(5)-18.*GEGL 440
     1XC(4)**2                                                              GEGL 450
      VC(5) = 12.+20.*XC(1)-64.*XC(2)+20.*XC(3)+40.*XC(4)-60.*XC(5)-6.*XGEGL 460
     1C(5)**2                                                               GEGL 470
      DO 30 I=1,5                                                           GEGL 480
      VC1 = 0.                                                              GEGL 490
      DO 25 J=6,15                                                          GEGL 500
      VC1 = VC1+A(I,J)*XC(J)                                                GEGL 510
   25 CONTINUE                                                              GEGL 520
      VC(I) = VC(I)+VC1                                                     GEGL 530
   30 CONTINUE                                                              GEGL 540
      VC(4) = VC(4)+XC(16)                                                  GEGL 550
      VC(5) = VC(5)+XC(17)                                                  GEGL 560
      RETURN                                                                GEGL 570
      END                                                                   GEGL 580
```

```
      SUBROUTINE  JACOB                                                 GEGM  10
      DIMENSION   A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)  GEGM  20
     1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)          GEGM  30
      COMMON      A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB         GEGM  40
      COMMON      NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGM  50
     1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGM  60
     2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGM  70
      COMMON      KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGM  80
     1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                              GEGM  90
      COMMON      IDIREC,DELTFI,ETA,JKO,LC,YSORT                        GEGM 100
      DIMENSION   B(10)                                                 GEGM 110
      COMMON      B                                                     GEGM 120
      A(1,1) = -24.*XC(1)-60.                                           GEGM 130
      A(2,2) = -48.*XC(2)-78.                                           GEGM 140
      A(3,3) = -60.*XC(3)-20.                                           GEGM 150
      A(4,4) = -36.*XC(4)-78.                                           GEGM 160
      A(5,5) = -12.*XC(5)-60.                                           GEGM 170
      RETURN                                                            GEGM 180
      END                                                               GEGM 190
```

```
      SUBROUTINE  GRADFI                                                    GEGN  10
      DIMENSION   A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)      GEGN  20
     1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)              GEGN  30
      COMMON      A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB             GEGN  40
      COMMON      NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGN  50
     1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGN  60
     2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGN  70
      COMMON      KFONC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGN  80
     1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                                  GEGN  90
      COMMON      IDIREC,DELTFI,ETA,JKO,LC,YSORT                            GEGN 100
      DIMENSION   B(10)                                                     GEGN 110
      COMMON      B                                                         GEGN 120
      IF (IT.GE.0) GO TO 100                                                GEGN 130
      DO 10 I=6,15                                                          GEGN 140
      C(I) = B(I-5)                                                         GEGN 150
   10 CONTINUE                                                              GEGN 160
  100 CONTINUE                                                              GEGN 170
      C(1) = -24.*XC(1)**2-60.*XC(1)+40.*XC(2)+20.*XC(3)-64.*XC(4)+20.*XGEGN 180
     1C(5)                                                                  GEGN 190
      C(2) = -48.*XC(2)**2+40.*XC(1)-78.*XC(2)+12.*XC(3)+62.*XC(4)-64.*XGEGN 200
     1C(5)                                                                  GEGN 210
      C(3) = -60.*XC(3)**2+20.*XC(1)+12.*XC(2)-20.*XC(3)+12.*XC(4)+20.*XGEGN 220
     1C(5)                                                                  GEGN 230
      C(4) = -36.*XC(4)**2-64.*XC(1)+62.*XC(2)+12.*XC(3)-78.*XC(4)+40.*XGEGN 240
     1C(5)                                                                  GEGN 250
      C(5) = -12.*XC(5)**2+20.*XC(1)-64.*XC(2)+20.*XC(3)+40.*XC(4)-60.*XGEGN 260
     1C(5)                                                                  GEGN 270
      RETURN                                                                GEGN 280
      END                                                                   GEGN 290
```

PARAMETRES

NV	17
NIN	3
NEG	2
NEVL	20
NTO	6
ITET	20
ICONJ	1
IDIAG	0
ITMAX	50
KFIL	0
KLIN	0
NCO	10
ITSCR	1
ISOLSR	50
EPSIL	0.1E 00
EPSILO	0.1E-03
EPSIL1	0.1E-03
EPSIL2	0.1E-03
PC	0.0

GRADIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES 17
NOMBRE TOTAL DE VARIABLES 23
NOMBRE DE CONTRAINTES 5

EPSILON DE NEWTON 0.1000E-03
EPSILON TEST GRADIENT 0.1000E-03

FONCTION ECONOMIQUE -0.800000000000000E 01

	BORNE INFERIEURE	VARIABLE NATURELLE		BORNE SUPERIEURE
XI(1)	0.0	X(1)	0.0	XS(1) 0.100000000000000E 03
XI(2)	0.0	X(2)	0.0	XS(2) 0.100000000000000E 03
XI(3)	0.0	X(3)	0.0	XS(3) 0.100000000000000E 03
XI(4)	0.0	X(4)	0.0	XS(4) 0.100000000000000E 03
XI(5)	0.0	X(5)	0.100000000000000E 01	XS(5) 0.100000000000000E 03
XI(6)	0.0	X(6)	0.0	XS(6) 0.100000000000000E 03
XI(7)	0.0	X(7)	0.0	XS(7) 0.100000000000000E 03
XI(8)	0.0	X(8)	0.0	XS(8) 0.100000000000000E 03
XI(9)	0.0	X(9)	0.0	XS(9) 0.100000000000000E 03
XI(10)	0.0	X(10)	0.0	XS(10) 0.100000000000000E 03
XI(11)	0.0	X(11)	0.0	XS(11) 0.100000000000000E 03
XI(12)	0.0	X(12)	0.0	XS(12) 0.100000000000000E 03
XI(13)	0.0	X(13)	0.0	XS(13) 0.100000000000000E 03
XI(14)	0.0	X(14)	0.500000000000000E 01	XS(14) 0.100000000000000E 03
XI(15)	0.0	X(15)	0.100000000000000E 01	XS(15) 0.100000000000000E 03
XI(16)	0.0	X(16)	0.0	XS(16) 0.100000000000000E 03
XI(17)	0.0	X(17)	0.280000000000000E 02	XS(17) 0.100000000000000E 03

VALEUR DES CONTRAINTES

C(1) 0.410000000000000E 02
C(2) -0.260000000000000E 02
C(3) 0.720000000000000E 02
C(4) 0.790000000000000E 02
C(5) 0.0

```
IT  1   PHI-0.200753356800000E 11   DIR.GRAD.  NO  6    YN 0.22E 11   DELTAFI 0.32E 13   ETA 0.16E 13   NCDB  0   NCN  2   NITN  2
IT  2   PHI-0.168571658240000E 11   DIR.CONJ.  LC 12                                                        NCDB  0   NCN  2   NITN  2
IT  3   PHI-0.168504975360000E 11   DIR.CONJ.  LC 12                                                        NCDB  0   NCN  2   NITN  1
IT  4   PHI-0.149107015680000E 11   ITERATION  SPECIALE  BASE  DEGENEREE                                    NCDB  1   NCN  1   NITN  1
IT  5   PHI-0.141003960960000E 11   DIR.GRAD.  NO  6    YN 0.11E 11   DELTAFI 0.21E 13   ETA 0.83E 12   NCDB  1   NCN  3   NITN  7
IT  6   PHI-0.685215744000000E 10   DIR.GRAD.  NO  5    YN 0.41E 10   DELTAFI 0.21E 13   ETA 0.70E 18   NCDB  1   NCN  5   NITN 18
IT  7   PHI-0.520674918400000E 10   DIR.GRAD.  NO  4    YN 0.12E 11   DELTAFI 0.19E 13   ETA 0.85E 12   NCDB  1   NCN  2   NITN 10
IT  8   PHI-0.312404684900000E 10   DIR.GRAD.  NO  4    YN 0.14E 11   DELTAFI 0.19E 13   ETA 0.11E 13   NCDB  2   NCN  5   NITN 67
IT  9   PHI-0.113037200927736E 03   DIR.GRAD.  NO 17    YN 0.12E 09   DELTAFI 0.31E 10   ETA 0.31E 10   NCDB  0   NCN  3   NITN 34
LES  VARIABLES  ARTIFICIELLES  SONT  TOUTES  ANNULEES
IT 10   PHI-0.652564392089846E 02   DIR.GRAD.  NO  8    YN 0.92E 02   DELTAFI 0.12E 05   ETA 0.56E 04   NCDB  1   NCN 23   NITN 25
IT 11   PHI-0.591575317382816E 02   DIR.GRAD.  NO 12    YN 0.42E 02   DELTAFI 0.58E 04   ETA 0.39E 04   NCDB  1   NCN  4   NITN 14
IT 12   PHI-0.424032440185556E 02   DIR.GRAD.  NO 12    YN 0.93E 01   DELTAFI 0.97E 03   ETA 0.60E 03   NCDB  1   NCN  5   NITN 41
IT 13   PHI-0.399825286865236E 02   DIR.GRAD.  NO 15    YN 0.40E 01   DELTAFI 0.25E 03   ETA 0.18E 03   NCDB  0   NCN  3   NITN 23
IT 14   PHI-0.393623962402346E 02   DIR.GRAD.  NO 16    YN 0.68E 01   DELTAFI 0.18E 03   ETA 0.18E 03   NCDB  0   NCN  2   NITN  6
IT 15   PHI-0.377737121582036E 02   DIR.CONJ.  LC  2                                                        NCDB  0   NCN  3   NITN 14
IT 16   PHI-0.374343719482426E 02   DIR.GRAD.  NO 16    YN 0.29E 01   DELTAFI 0.18E 03   ETA 0.13E 03   NCDB  0   NCN  1   NITN  3
IT 17   PHI-0.370439300537116E 02   DIR.GRAD.  NO 16    YN 0.28E 01   DELTAFI 0.12E 03   ETA 0.12E 03   NCDB  0   NCN  2   NITN  7
IT 18   PHI-0.367792053222666E 02   DIR.GRAD.  NO 15    YN 0.25E 01   DELTAFI 0.12E 03   ETA 0.11E 03   NCDB  0   NCN  1   NITN  3
IT 19   PHI-0.349787139892586E 02   DIR.GRAD.  NO 16    YN 0.23E 01   DELTAFI 0.12E 03   ETA 0.10E 03   NCDB  0   NCN  2   NITN 15
IT 20   PHI-0.328193664550786E 02   DIR.CONJ.  LC  2                                                        NCDB  0   NCN  2   NITN 19
IT 21   PHI-0.327176513671886E 02   DIR.CONJ.  LC  2                                                        NCDB  0   NCN  2   NITN  7
IT 22   PHI-0.324473266601566E 02   DIR.GRAD.  NO 14    YN 0.31E 01   DELTAFI 0.12E 03   ETA 0.30E 03   NCDB  0   NCN  1   NITN  3
IT 23   PHI-0.323720092773446E 02   DIR.GRAD.  NO 14    YN 0.19E 01   DELTAFI 0.12E 03   ETA 0.18E 03   NCDB  0   NCN  2   NITN  5
IT 24   PHI-0.323695220947276E 02   DIR.CONJ.  LC  4                                                        NCDB  0   NCN  2   NITN  2
IT 25   PHI-0.323682250976566E 02   DIR.GRAD.  NO 15    YN 0.55E 00   DELTAFI 0.18E 02   ETA 0.11E 02   NCDB  0   NCN  2   NITN  2
IT 26   PHI-0.323602142333986E 02   DIR.CONJ.  LC  3                                                        NCDB  0   NCN  3   NITN  4
IT 27   PHI-0.323518676757816E 02   DIR.GRAD.  NO 12    YN 0.53E 00   DELTAFI 0.10E 02   ETA 0.99E 01   NCDB  0   NCN  8   NITN 10
IT 28   PHI-0.323513641357426E 02   DIR.GRAD.  NO 14    YN 0.16E 00   DELTAFI 0.41E 01   ETA 0.38E 01   NCDB  0   NCN  2   NITN  2
IT 29   PHI-0.323506927490236E 02   DIR.CONJ.  LC  4                                                        NCDB  0   NCN  2   NITN  2
IT 30   PHI-0.323503875732426E 02   DIR.GRAD.  NO 14    YN 0.12E 00   DELTAFI 0.41E 01   ETA 0.11E 02   NCDB  0   NCN  3   NITN  3
IT 31   PHI-0.323502349853526E 02   DIR.GRAD.  NO 14    YN 0.47E-01   DELTAFI 0.34E 01   ETA 0.18E 01   NCDB  0   NCN  4   NITN  1
IT 32   PHI-0.323501586914066E 02   DIR.GRAD.  NO 14    YN 0.59E-01   DELTAFI 0.34E 01   ETA 0.43E 01   NCDB  0   NCN  2   NITN  0
IT 33   PHI-0.323493804931646E 02   DIR.GRAD.  NO 14    YN 0.37E-01   DELTAFI 0.33E 01   ETA 0.18E 01   NCDB  0   NCN  2   NITN  2
IT 34   PHI-0.323487396240236E 02   DIR.CONJ.  LC  4                                                        NCDB  0   NCN  2   NITN  2
IT 35   PHI-0.323487091064456E 02   DIR.CONJ.  LC  4                                                        NCDB  0   NCN  2   NITN  0
IT 36   PHI-0.323486785888676E 02   DIR.CONJ.  LC  4                                                        NCDB  0   NCN  3   NITN  0
        PHI-0.323486785888676E 02                      YN 0.67E-02   DELTAFI 0.33E 00   ETA 0.31E 00

DUREE  DU  CALCUL   1288  CENTISECONDES
```

VARIABLE NATURELLE		VARIABLE DUALE ASSOCIEE	
X(1) =	C.29945605993271E 00	V(1) =	0.
X(2) =	C.33329182863735E 00	V(2) =	-0.69427490234375E-03
X(3) =	C.40060669183731E 00	V(3) =	0.
X(4) =	C.42876362800598E 00	V(4) =	0.
X(5) =	C.22434699930725E 00	V(5) =	0.
X(6) =	C.0 = XI(6)	V(6) =	-0.36304092407227E 02
X(7) =	C.0 = XI(7)	V(7) =	-0.19537429809570E 01
X(8) =	C.51755962371826E 01	V(8) =	-0.31472444534302E-02
X(9) =	C.0 = XI(9)	V(9) =	-0.13940868377686E 01
X(10) =	C.30615062713623E 01	V(10) =	0.
X(11) =	C.11632645416260E 02	V(11) =	0.35247802734375E-02
X(12) =	C.0 = XI(12)	V(12) =	-0.38313537597656E 02
X(13) =	C.0 = XI(13)	V(13) =	-0.56750320434570E 02
X(14) =	C.10277658700943E 00	V(14) =	-0.46758651733398E-02
X(15) =	C.0 = XI(15)	V(15) =	-0.68643802404404E 00
X(16) =	C.0 = XI(16)	V(16) =	-0.42875647544861E 00
X(17) =	C.0 = XI(17)	V(17) =	-0.22437608242035E 00

CONTRAINTE		VARIABLE DUALE ASSOCIEE	
CONTRAINTE 1	-0.11444091796875E-03	U(1) =	0.29944628477097E 00
CONTRAINTE 2	-0.61035156250000E-04	U(2) =	0.33325487375259E 00
CONTRAINTE 3	-0.91552734375000E-04	U(3) =	0.40060430765152E 00
CONTRAINTE 4	0.57220458984375E-05	U(4) =	0.42875647544861E 00
CONTRAINTE 5	0.17166137695313E-04	U(5) =	0.22437608242035E 00

```
NIVEAU  DE  SORTIE      1

NOMBRE  D ITERATIONS        36

NOMBRE  D APPELS  DU  S/P  CALCULANT  LES  CONTRAINTES   466

NOMBRE  D APPELS  DU  S/P  CALCULANT  LA  FONCTION  ECONOMIQUE   298

NOMBRE  D APPELS  DU  S/P  CALCULANT  LE  GRADIENT  DE  LA  FONCTION  ECONOMIQUE   104

NOMBRE  D APPELS  DU  S/P  CALCULANT  L INVERSE  AVEC  BASE  DONNEE    39
NOMBRE  D APPELS  DU  S/P  CALCULANT  L INVERSE  AVEC  RECHERCHE  DE  BASE     0

NOMBRE  D APPELS  DU  S/P  EFFECTUANT  LES  CHANGEMENTS  DE  BASE     9

NOMBRE  D APPELS  DU  S/P  CALCULANT  LE  JACOBIEN  DES  CONTRAINTES    43

NOMBRE  D APPELS  DU  S/P  EFFECTUANT  UNE  RECHERCHE  DE  MAX  SANS  DICHOTOMIE    33
NOMBRE  D APPELS  DU  S/P  EFFECTUANT  UNE  RECHERCHE  DE  MAX  AVEC  DICHOTOMIE     3

NOMBRE  D APPELS  DU  S/P  EFFECTUANT  LA  RENTREE  DANS  LE  DOMAINE   115                NOMBRE  TOTAL  D ITERATIONS   357
```

```
C001              SUBROUTINE PHIX
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)   GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)          GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                              GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                         GEGA 100
C007              COMMON B(5),CC(3,4)
C008              IF(IT)100,110,110
C009          100 B(1)=-40792.14
C010              B(2)=37.29324
C011              B(3)=0.835689
C012              B(4)=5.357855
C013          110 PHI=B(1)+B(2)*XC(1)+B(3)*XC(1)*XC(5)+B(4)*XC(3)**2
C014              PHI=PHI+30373.94
C015              PHI=-PHI
C016              RETURN
C017              END
```

```
C001              SUBROUTINE CPHI
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
C007              COMMON B(5),CC(3,4)
C008              IF(IT)100,110,110
C009          100 CC(1,1)=85.33441
C010              CC(1,2)=0.005686
C011              CC(1,3)=0.000626
C012              CC(1,4)=-0.002205
C013              CC(2,1)=80.51249
C014              CC(2,2)=0.007132
C015              CC(2,3)=0.002996
C016              CC(2,4)=0.002181
C017              CC(3,1)=9.300961
C018              CC(3,2)=0.004703
C019              CC(3,3)=0.001755
C020              CC(3,4)=0.001908
C021          110 VC(1)=CC(1,1)+CC(1,2)*XC(2)*XC(5)+CC(1,3)*XC(1)*XC(4)+CC(1,4)
                 1*XC(3)*XC(5)
C022              VC(2)=-VC(1)
C023              VC(1)=VC(1)-92.0
C024              VC(3)=CC(2,1)+CC(2,2)*XC(2)*XC(5)+CC(2,3)*XC(1)*XC(2)+CC(2,4)
                 1*XC(3)**2
C025              VC(4)=-VC(3)+90.0
C026              VC(3)=VC(3)-110.0
C027              VC(5)=CC(3,1)+CC(3,2)*XC(3)*XC(5)+CC(3,3)*XC(1)*XC(3)+CC(3,4)
                 1*XC(3)*XC(4)
C028              VC(6)=-VC(5)+20.0
C029              VC(5)=VC(5)-25.0
C030              RETURN
C031              END
```

```
C001              SUBROUTINE JACOB
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
C007              COMMON B(5),CC(3,4)
C008              A(1,1)=CC(1,3)*XC(4)
C009              A(1,2)=CC(1,2)*XC(5)
C010              A(1,3)=CC(1,4)*XC(5)
C011              A(1,4)=CC(1,3)*XC(1)
C012              A(1,5)=CC(1,2)*XC(2)+CC(1,4)*XC(3)
C013              A(3,1)=CC(2,3)*XC(2)
C014              A(3,2)=CC(2,2)*XC(5)+CC(2,3)*XC(1)
C015              A(3,3)=2.0*CC(2,4)*XC(3)
C016              A(3,5)=CC(2,2)*XC(2)
C017              A(5,1)=CC(3,3)*XC(3)
C018              A(5,3)=CC(3,2)*XC(5)+CC(3,4)*XC(4)+CC(3,3)*XC(1)
C019              A(5,4)=CC(3,4)*XC(3)
C020              A(5,5)=CC(3,2)*XC(3)
C021              DO1I=2,6,2
C022              DO1J=1,5
C023            1 A(I,J)=-A(I-1,J)
C024              RETURN
C025              END
```

```
C001              SUBROUTINE GRADFI
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                             GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                         GEGA 100
C007              COMMON B(5),CC(3,4)
C008              C(1)=-B(2)-B(3)*XC(5)
C009              C(3)=-2.0*B(4)*XC(3)
C010              C(5)=-B(3)*XC(1)
C011              RETURN
C012              END
```

PARAMETRES

NV	5
NIN	6
NEG	0
NEVL	20
NTO	6
ITET	20
ICONJ	1
ICIAG	0
ITMAX	50
KFIL	0
KLIN	0
NCO	10
ITSCR	1
ISOLSR	10
EPSIL	0.1E 00
EPSILO	0.1E-04
EPSIL1	0.1E-02
EPSIL2	0.1E-02
PC	0.0

GRADIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES 5
NOMBRE TOTAL DE VARIABLES 11
NOMBRE DE CONTRAINTES 6

EPSILON DE NEWTON 0.1000E-04
EPSILON TEST GRADIENT 0.1000E-02

FONCTION ECONOMIQUE 0.11718750000000E-01

BORNE INFERIEURE		VARIABLE NATURELLE		BORNE SUPERIEURE	
XI(1)	0.78000000000000E 02	X(1)	0.78619995117188E 02	XS(1)	0.10200000000000E 03
XI(2)	0.33000000000000E 02	X(2)	0.33439987182617E 02	XS(2)	0.45000000000000E 02
XI(3)	0.27000000000000E 02	X(3)	0.31069992065430E 02	XS(3)	0.45000000000000E 02
XI(4)	0.27000000000000E 02	X(4)	0.44099990844727E 02	XS(4)	0.45000000000000E 02
XI(5)	0.27000000000000E 02	X(5)	0.35219985961914E 02	XS(5)	0.45000000000000E 02

VALEUR DES CONTRAINTES

C(1)	-0.21136474609375E 00
C(2)	-0.91788635253906E 02
C(3)	-0.11105728149414E 02
C(4)	-0.88942718505859E 01
C(5)	-0.48727111816406E 01
C(6)	-0.12728881835938E 00

```
IT  1   PHI 0.118175781250000E 03   DIR.GRAD.  NO  2   YN 0.35E 03   DELTAFI 0.19E 04   ETA 0.14E 04   NCDB  3   NCN  2   NIT.  2
IT  2   PHI 0.118234375000000E 03   DIR.CONJ.  LC  3                                                      NCDB  0   NCN  2   NIT.  2
IT  3   PHI 0.118289062500000E 03   DIR.CONJ.  LC  3                                                      NCDB  0   NCN  2   NIT.  2
IT  4   PHI 0.254003906250000E 03   DIR.GRAD.  NO  2   YN 0.95E 03   DELTAFI 0.57E 03   ETA 0.70E 03   NCDB  2   NCN  8   NIT. 20
IT  5   PHI 0.256320312500000E 03   DIR.GRAD.  NO  3   YN 0.83E 03   DELTAFI 0.38E 02   ETA 0.38E 02   NCDB  0   NCN  2   NIT.  7
IT  6   PHI 0.292011718750000E 03   DIR.GRAD.  NO  4   YN 0.83E 03   DELTAFI 0.36E 02   ETA 0.36E 02   NCDB  0   NCN  3   NIT. 12
        PHI 0.292011718750000E 03                      YN 0.81E 03   DELTAFI 0.43E-02   ETA 0.43E-02
```

DUREE DU CALCUL 165 CENTISECONDES

VARIABLE NATURELLE		VARIABLE DUALE ASSOCIEE	
X(1) = 0.780000000000000E 02	= XI(1)	V(1) = -0.489176025390630E 02	
X(2) = 0.330000000000000E 02	= XI(2)	V(2) = -0.843099212646484E 02	
X(3) = 0.299938964843750E 02		V(3) = 0.	
X(4) = 0.450000000000000E 02	= XS(4)	V(4) = 0.266324462890630E 02	
X(5) = 0.367761230468750E 02		V(5) = 0.	

CONTRAINTE		VARIABLE DUALE ASSOCIEE	
CONTRAINTE 1	-0.152587890625000E-04	U(1) = 0.403186523437500E 03	
CONTRAINTE 2	-0.919999847412110E 02	U(2) = 0.0	
CONTRAINTE 3	-0.111582489013672E 02	U(3) = 0.0	
CONTRAINTE 4	-0.884175109863280E 01	U(4) = 0.0	
CONTRAINTE 5	-0.499998474121090E 01	U(5) = 0.0	
CONTRAINTE 6	-0.152587890625000E-04	U(6) = 0.809375732421880E 03	

```
NIVEAU DE SORTIE     7

NOMBRE D ITERATIONS        6

NOMBRE D APPELS DU S/P CALCULANT LES CONTRAINTES    93

NOMBRE D APPELS DU S/P CALCULANT LA FONCTION ECONOMIQUE    69

NOMBRE D APPELS DU S/P CALCULANT LE GRADIENT DE LA FONCTION ECONOMIQUE    14

NOMBRE D APPELS DU S/P CALCULANT L INVERSE AVEC BASE DONNEE     4
NOMBRE D APPELS DU S/P CALCULANT L INVERSE AVEC RECHERCHE DE BASE     1

NOMBRE D APPELS DU S/P EFFECTUANT LES CHANGEMENTS DE BASE     3

NOMBRE D APPELS DU S/P CALCULANT LE JACOBIEN DES CONTRAINTES     8

NOMBRE D APPELS DU S/P EFFECTUANT UNE RECHERCHE DE MAX SANS DICHOTOMIE    14
NOMBRE D APPELS DU S/P EFFECTUANT UNE RECHERCHE DE MAX AVEC DICHOTOMIE     0

NOMBRE D APPELS DU S/P EFFECTUANT LA RENTREE DANS LE DOMAINE    26                NOMBRE TOTAL D ITERATIONS    74
```

```
C001              SUBROUTINE PHIX
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFCNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                               GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
C007              PHI=-4.0*XC(1)-10.0*XC(2)-4.0*XC(3)-2.0*SQRT(XC(1)**2+XC(2)**2)
C008              RETURN
C009              END
```

```
C001              SUBROUTINE CPHI
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
C005              COMMON     KFCNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                               GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
C007              VC(1)=-XC(1)*XC(2)*XC(3)+100.0
C008              RETURN
C009              END
```

```
C001              SUBROUTINE JACOB
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                               GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
C007              A(1,1)=-XC(2)*XC(3)
C008              A(1,2)=-XC(1)*XC(3)
C009              A(1,3)=-XC(1)*XC(2)
C010              RETURN
C011              END
```

```
C001              SUBROUTINE GRADFI
C002              DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
                 1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
C003              COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
C004              COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
                 1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
                 2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
C005              COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
                 1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                               GEGA  90
C006              COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSORT                          GEGA 100
C007              SS=SQRT(XC(1)**2+XC(2)**2)
C008              C(1)=-4.0-2.0*XC(1)/SS
C009              C(2)=-10.0-2.0*XC(2)/SS
C010              C(3)=-4.0
C011              RETURN
C012              END
```

PARAMETRES

NV	3
NIN	1
NEG	0
NEVL	20
NTO	6
ITET	20
ICONJ	1
IDIAG	0
ITMAX	50
KFIL	0
KLIN	0
NCO	10
ITSCR	1
ISOLSR	50
EPSIL	0.1E 00
EPSIL0	0.1E-04
EPSIL1	0.1E-02
EPSIL2	0.1E-02
PC	0.1E 01

GRADIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES	3
NOMBRE TOTAL DE VARIABLES	4
NOMBRE DE CONTRAINTES	1

EPSILON DE NEWTON	0.1000E-04
EPSILON TEST GRADIENT	0.1000E-02

FONCTION ECONOMIQUE -0.2F8284271240236 03

BORNE INFERIEURE		VARIABLE NATURELLE		BORNE SUPERIEURE	
XI(1)	0.0	X(1)	0.100000000000000E 02	XS(1)	0.100000000000000E 03
XI(2)	0.0	X(2)	0.100000000000000E 02	XS(2)	0.100000000000000E 03
XI(3)	0.0	X(3)	0.100000000000000E 02	XS(3)	0.100000000000000E 03

VALEUR DES CONTRAINTES

C(1) -0.900000000000000E 03

```
IT  1   PHI-0.1327776489257BE 03   DIR.GRAD.  NO  C    YN 0.13E 02   DELTAFI 0.21E 03   ETA 0.11E 03   NCDB  3   NCN  2   NITN  2
IT  2   PHI-0.9837782287597 7E 02   DIR.CONJ.  LC  3                                                       NCDB  3   NCN  2   NITN  2
IT  3   PHI-0.8984851074218 8E 02   DIR.CONJ.  LC  3                                                       NCDB  0   NCN  2   NITN  2
IT  4   PHI-0.8931365966796 9E 02   DIR.CONJ.  LC  3                                                       NCDB  1   NCN  3   NITN 10
IT  5   PHI-0.8879714965820 3E 02   DIR.GRAD.  NO  1    YN 0.28E 01   DELTAFI 0.21E 03   ETA 0.24E 03   NCDB  0   NCN  3   NITN 33
IT  6   PHI-0.8799572753906 3E 02   DIR.CONJ.  LC  2                                                       NCDB  0   NCN  5   NITN 49
IT  7   PHI-0.8799032592773 4E 02   DIR.CONJ.  LC  2                                                       NCDB  0   NCN  3   NITN 30
IT  8   PHI-0.8798777770996 1E 02   DIR.GRAD.  NO  0    YN 0.29E 00   DELTAFI 0.39E 00   ETA 0.24E 00   NCDB  0   NCN  7   NITN 10
IT  9   PHI-0.8798774719238 3E 02   DIR.GRAD.  NO  1    YN 0.18E-01   DELTAFI 0.39E 00   ETA 0.17E 01   NCDB  0   NCN  4   NITN  6
        PHI-0.8798774719238 3E 02                       YN 0.68E-02   DELTAFI 0.42E-01   ETA 0.34E-01


DUREE  DU  CALCUL     83  CENTISECONDES

          VARIABLE  NATURELLE                          VARIABLE  DUALE  ASSOCIEE


X(  1) =  0.50872011164692E 01                    V(  1) = -0.66871643066406E-02
X(  2) =  0.26814708709717E 01                    V(  2) =  0.
X(  3) =  0.73307437896729E 01                    V(  3) = -0.10337829589844E-02


              CONTRAINTE                          VARIABLE  DUALE  ASSOCIEE


CONTRAINTE  1        0.0                          U(  1) =  0.29315400123596E 00


NIVEAU  DE  SORTIE      1

NOMBRE  D ITERATIONS          9

NOMBRE  D APPELS  DU  S/P  CALCULANT  LES  CONTRAINTES   167

NOMBRE  D APPELS  DU  S/P  CALCULANT  LA  FONCTION  ECONOMIQUE    95

NOMBRE  D APPELS  DU  S/P  CALCULANT  LE  GRADIENT  DE  LA  FONCTION  ECONOMIQUE    19

NOMBRE  D APPELS  DU  S/P  CALCULANT  L INVERSE  AVEC  BASE  DONNEE      7
NOMBRE  D APPELS  DU  S/P  CALCULANT  L INVERSE  AVEC  RECHERCHE  DE  BASE      1

NOMBRE  D APPELS  DU  S/P  EFFECTUANT  LES  CHANGEMENTS  DE  BASE      1

NOMBRE  D APPELS  DU  S/P  CALCULANT  LE  JACOBIEN  DES  CONTRAINTES    12

NOMBRE  D APPELS  DU  S/P  EFFECTUANT  UNE  RECHERCHE  DE  MAX  SANS  DICHOTOMIE     8
NOMBRE  D APPELS  DU  S/P  EFFECTUANT  UNE  RECHERCHE  DE  MAX  AVEC  DICHOTOMIE     2

NOMBRE  D APPELS  DU  S/P  EFFECTUANT  LA  RENTREE  DANS  LE  DOMAINE    31          NOMBRE  TOTAL  D ITERATIONS   144
```

APPENDIX 5. GREG PROGRAMMING MATERIAL FOR CHAPTER SIX

This appendix contains the GREG external subroutines and four supporting subroutines used to optimize the water quality control model of Chapter six. The four supporting subroutines are

1. MINDO - minimizes the dissolved oxygen function in a given stage by a Fibonacci search.

2. INPUT - reads in the supplementary parameter values, initializes variables, and determines the number of iterations the Fibonacci search must perform in each stage to guarantee a final search interval length = 0.0001.

3. FUNCT - a function that calculates the DO profile in each stage.

4. CHANGE- calculates the boundary values.

Definition of the Variables*

XT(I)	thermal input in BTU/hr
XQ(I)	flow in ft^3/sec
XB(I)	organic waste in lbs/day
XE(I)	equilibrium temperature in ^oF
AK(I)	temperature dissipation rate coefficient in $(days)^{-1}$
AK1(I)	deoxygenation rate coefficient in $(days)^{-1}$ at 20 ^oC
AK2(I)	reaeration rate coefficient in $(days)^{-1}$ at 20 ^oC
ST(I)	incoming water temperature in ^oF
SB(I)	incoming BOD in mg/ℓ
SO(I)	incoming DO in mg/ℓ
TF(I)	flow in days
TRS(I)	maximum allowable temperature rise in ^oF
TMAX(I)	maximum allowable temperature in ^oF
MDO(I)	minimum allowable DO in mg/ℓ
TMIN	the flow time when DO is minimum, $0 \leq TMIN \leq TF(I)$
FMIN	minimum DO
NSTAGE	the number of stages
CC1	the conversion factor C_1
CC2	the conversion factor C_2

* - each subscripted variable ranges from I = 1,..., NSTAGE.

```
      SUBROUTINE PHIX
      DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)    GEGA  20
     1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
      COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB           GEGA  40
      COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
     1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
     2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TC,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
      COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA  80
     1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                               GEGA  90
      COMMON     ICTREC,DELTFI,ETA,JKO,LC,YSCRT                          GEGA 100
      COMMON/AREA1/XT(4),XQ(4),XB(4),XE(4),AK(4),AK1(4),AK2(4),ST(4)
     1,SB(4),SO(4),TF(4),TRS(4),TMAX(4),MDC(4),NSTAGE
     2,CC1,CC2,DELTA
      REAL MDO
      IF(IT.LT.0) CALL INPUT
      PHI=0.0
      PHI=PHI+(0.817/0.9)*(1.0-EXP(-0.023*XC(2)))
      PHI=PHI+(0.575/0.9)*(1.0-EXP(-0.023*XC(4)))
      IF(XC(5).LT.20.0) GO TO 10
      IF(XC(5).LT.30.0) GO TO 11
      IF(XC(5).LT.40.0) GO TO 12
      IF(XC(5).LT.70.0) GO TO 13
      IF(XC(5).LT.90.0) GO TO 14
      IF(XC(5).LT.95.0) GO TO 15
      PHI=PHI+0.98*XC(5)-91.2
      GO TO 30
   10 PHI=PHI+0.04*XC(5)
      GO TO 30
   11 PHI=PHI+0.03*XC(5)+0.2
      GO TO 30
   12 PHI=PHI+0.01*XC(5)+0.8
      GO TO 30
   13 PHI=PHI+(0.01/3.0)*XC(5)-(0.4/3.0)+1.2
      GO TO 30
   14 PHI=PHI+0.015*XC(5)+0.25
      GO TO 30
   15 PHI=PHI+0.06*XC(5)-3.8
   30 PHI=-PHI
      RETURN
      END
```

```
      SUBROUTINE GRADFI
      DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)   GEGA  20
     1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)          GEGA  30
      COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
      COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
     1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
     2DB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TD,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
      COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACC,KMAX1,KMAX2,KGEGA  80
     1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                              GEGA  90
      COMMON     IDTREC,DELTFI,ETA,JKO,LC,YSCRT                         GEGA 100
      COMMON/AREA1/XT(4),XQ(4),XB(4),XE(4),AK(4),AK1(4),AK2(4),ST(4)
     1,SB(4),SO(4),TF(4),TRS(4),TMAX(4),MDC(4),NSTAGE
     2,CC1,CC2,DELTA
      REAL MDC
      C(2)=-(0.817/0.9)*EXP(-0.023*XC(2))*0.023
      C(4)=-(0.575/0.9)*EXP(-0.023*XC(4))*0.023
      IF(XC(5).LT.20.0) GO TO 10
      IF(XC(5).LT.30.0) GO TO 11
      IF(XC(5).LT.40.0) GO TO 12
      IF(XC(5).LT.70.0) GO TO 13
      IF(XC(5).LT.90.0) GO TO 14
      IF(XC(5).LT.95.0) GO TO 15
      C(5)=-0.98
      RETURN
   10 C(5)=-0.04
      RETURN
   11 C(5)=-0.03
      RETURN
   12 C(5)=-0.01
      RETURN
   13 C(5)=-(0.01/3.0)
      RETURN
   14 C(5)=-0.015
      RETURN
   15 C(5)=-0.06
      RETURN
      END
```

```
      SUBROUTINE CPHI
      DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)   GEGA  20
     1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)          GEGA  30
      COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
      COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTC,NIN1,NIN2,NIN3,NIGEGA  50
     1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
     2CB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
      COMMON     KFUNC,KGRAD,KCONT,KINV1,KINV2,KCCBA,KJACO,KMAX1,KMAX2,KGEGA  80
     1REN1,KREN2,KINV,KCCBA1,KREN11,KREN21                              GEGA  90
      COMMON     IDIREC,DELTFI,ETA,JKO,LC,YSCRT                         GEGA 100
      COMMON/AREA1/XT(4),XO(4),XB(4),XE(4),AK(4),AK1(4),AK2(4),ST(4)
     1,SB(4),SO(4),TF(4),TRS(4),TMAX(4),MCC(4),NSTAGE
     2,CC1,CC2,DELTA
      REAL MCO
C.....INITIALIZE
      IF(IT.GE.0) GO TO 100
      DO5I=1,NSTAGE
      XB(I)=(CC2*XE(I))/(100.0*XC(I))
      XT(I)=(CC1*XT(I))/(100.0*XC(I))
    5 CONTINUE
C.....END OF INITIALIZATION
C.....SET UP BOUNDARY CONDITIONS
  100 CALL CHANGE(1,NSTAGE-1)
C.....END OF BOUNDARY CONDITIONS
C.....INEQUALITY CONSTRAINTS
C.....I SUBSCRIPTS...MAX TEMP RISE
C.....NSTAGE+I SUBSCRIPTS...MAX TEMP
C.....2*NSTAGE+I SUBSCRIPTS...MIN CC
      DO10I=1,NSTAGE
      VC(I)=XT(I)*(100.0-XC(I))-TRS(I)
      VC(NSTAGE+I)=ST(I)+XT(I)*(100.0-XC(I))-TMAX(I)
      CALL MINDO(I,TMIN,FMIN)
      VC(2*NSTAGE+I)=-FMIN+MCO(I)
   10 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE JACOB
      DIMENSION  A(150,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)   GEGA  20
     1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)           GEGA  30
      COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB           GEGA  40
      COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NICEGA  50
     1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IP1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
     2DB,KFIL,KLIN,KREN,KC,FI1,PHI,PSI,PSI3,TB,TC,TC,EPSIL,EPSILO,EPSIL2GEGA  70
      COMMON     KECNC,KGRAC,KCONT,KINV1,KINV2,KCDBA,KJACC,KMAX1,KMAX2,KGEGA  80
     1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                               GEGA  90
      COMMON     ICIREC,DELTFI,ETA,JKO,LC,YSCRT                          GEGA 100
      COMMON/AREA1/XT(4),XC(4),XB(4),XE(4),AK(4),AK1(4),AK2(4),ST(4)
     1,SB(4),SO(4),TF(4),TRS(4),TMAX(4),MDC(4),NSTAGE
     2,CC1,CC2,DELTA
      REAL MDO
C.....INITIALIZE CONSTANTS
      IF(IT.GE.0) GO TO 100
      DO60J=1,NSTAGE
      AXXX=0.0
      K=J
   50 A(NSTAGE+J,K)=-XT(K)*EXP(AXXX)
      K=K-1
      IF(K.LE.0) GO TO 55
      AXXX=AXXX-AK(K)*TF(K)
      GO TO 50
   55 A(J,J)=-XT(J)
   60 CONTINUE
C.....END OF INITIALIZATION
  100 CALL CHANGE(1,NSTAGE-1)
C.....NUMERICAL PARTIAL DERIVATIVES FOR THE MIN DO CONSTRAINT
      DO30I=1,NSTAGE
      CALL MINDO(I,TMIN,FMIN)
      IF(XT(I).EQ.0.0) GO TO 5
      XXCC=XC(I)
      XC(I)=XC(I)+DELTA
      CALL MINDO(I,TMIN1,FMIN1)
      A(2*NSTAGE+I,I)=(FMIN-FMIN1)/DELTA
      XC(I)=XXCC
    5 IF(XB(I).EQ.0.0) GO TO 6
      XXCC=XC(NSTAGE+I)
      XC(NSTAGE+I)=XC(NSTAGE+I)+DELTA
      CALL MINDO(I,TMIN1,FMIN1)
      A(2*NSTAGE+I,NSTAGE+I)=(FMIN-FMIN1)/DELTA
      XC(NSTAGE+I)=XXCC
    6 IF(I.EQ.1) GO TO 30
      III=I-1
      DO20J=1,III
      IF(XT(J).EQ.0.0) GO TO 7
      XXCC=XC(J)
      XC(J)=XC(J)+DELTA
      CALL CHANGE(J,III)
      CALL MINDO(I,TMIN1,FMIN1)
      A(2*NSTAGE+I,J)=(FMIN-FMIN1)/DELTA
      XC(J)=XXCC
    7 IF(XB(J).EQ.0.0) GO TO 8
      XXCC=XC(NSTAGE+J)
      XC(NSTAGE+J)=XC(NSTAGE+J)+DELTA
      CALL CHANGE(J,III)
      CALL MINDO(I,TMIN1,FMIN1)
```

```
      A(2*NSTAGE+I,NSTAGE+J)=(FMIN-FMIN1)/DELTA
      XC(NSTAGE+J)=XXCC
    8 IF(XT(J).EQ.C.0.AND.XB(J).EQ.0.0) GC TC 20
      CALL CHANGE(J,III)
   2C CONTINUE
   3C CONTINUE
      RETURN
      END
```

```
      SUBROUTINE CHANGE(J,N)
      DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)   GEGA  20
     1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)          GEGA  30
      COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
      COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA  50
     1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
     2DB,KFIL,KLIN,KREN,KD,FII,PHI,PSI,PSIB,TB,TD,TC,EPSIL,EPSILO,EPSIL2GEGA  70
      COMMON     KFONC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACC,KMAX1,KMAX2,KGEGA  80
     1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                              GEGA  90
      COMMON     ICIREC,DELTFI,ETA,JKO,LC,YSORT                         GEGA 100
      COMMON/AREA1/XT(4),XO(4),XB(4),XE(4),AK(4),AK1(4),AK2(4),ST(4)
     1,SB(4),SO(4),TF(4),TRS(4),TMAX(4),MDC(4),NSTAGE
     2,CC1,CC2,DELTA
      REAL MDO
C.....FUNCTION DEFINITIONS
      TEMPC(T)=(5.0/9.0)*(T-32.0)
      CS(T)=14.652-0.41022*TEMPC(T)+0.79910E-2*TEMPC(T)**2-0.77774E-4
     1*TEMPC(T)**3
C.....END OF FUNCTION DEFINITIONS
C.....SET UP BOUNDS FOR STAGES J+1 TO N+1
      DO20I=J,N
      TMIX=ST(I)+XT(I)*(100.0-XC(I))
      ST(I+1)=XE(I)+(TMIX-XE(I))*EXP(-AK(I)*TF(I))
      BOD=SB(I)+XB(I)*(100.0-XC(NSTAGE+I))
      AAK1=AK1(I)*(1.047)**(TEMPC(ST(I+1))-20.0)
      SB(I+1)=BOD*EXP(-AAK1*TF(I))
      AAK2=AK2(I)*(1.024)**(TEMPC(ST(I+1))-20.0)
      DODEF=CS(TMIX)-SO(I)
      IF(DODEF.LE.0.0) DODEF=0.0
      SO(I+1)=CS(ST(I+1))-(AAK1*BOD/(AAK2-AAK1))*(EXP(-AAK1*TF(I))
     1-EXP(-AAK2*TF(I)))-DODEF*EXP(-AAK2*TF(I))
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MINDC(I,TMIN,FMIN)                                          GEGA  20
      DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)        GEGA  30
     1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)               GEGA  40
      COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB               GEGA  50
      COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NICEGA  60
     1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  70
     2DB,KFIL,KLIN,KREN,KC,FI1,PHI,PSI,PSI3,TB,TC,TC,EPSIL,EPSILO,EPSIL2GEGA  80
      COMMON     KFONC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACC,KMAX1,KMAX2,KGEGA  90
     1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21                                   GEGA  95
      COMMON     ICTREC,DELTFI,ETA,JKO,LC,YSORT                              GEGA 100
      COMMON/AREA1/XT(4),XQ(4),XB(4),XE(4),AK(4),AK1(4),AK2(4),ST(4)
     1,SB(4),SO(4),TF(4),TRS(4),TMAX(4),MDC(4),NSTAGE
     2,CC1,CC2,DELTA
      COMMON/AREA2/NTIMES(4),DEL(4),NF(30)
      REAL MDC
C.....FIBONNACI SEARCH FOR THE MIN DC IN STAGE I
      XA=0.0
      XBB=TF(I)
      X1=XA+DEL(I)*(XBB-XA)
      X2=XA+XBB-X1
      Y1=FUNCT(I,X1)
      Y2=FUNCT(I,X2)
      YA=FUNCT(I,XA)
      YB=FUNCT(I,XBB)
C.....CHECK FOR MINIMUM AT BOUNDS
      FMIN=AMIN1(Y1,Y2,YA,YB)
      IF(FMIN.EQ.YA) GO TO 100
      IF(FMIN.EQ.YB) GO TO 200
      N=NTIMES(I)
      DO300J=1,N
      IF(Y1.LT.Y2) GO TO 10
      XBB=X1
      X1=X2
      Y1=Y2
      X2=XA+XBB-X1
      Y2=FUNCT(I,X2)
      GO TO 300
   10 XA=X2
      X2=X1
      Y2=Y1
      X1=XA+XBB-X2
      Y1=FUNCT(I,X1)
  300 CONTINUE
      IF(Y1.LT.Y2) GO TO 20
      TMIN=X2
      FMIN=Y2
      RETURN
   20 TMIN=X1
      FMIN=Y1
      RETURN
  100 TMIN=0.0
      RETURN
  200 TMIN=TF(I)
      RETURN
      END
```

```
      FUNCTION FUNCT(I,TTT)
      DIMENSION  A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)   GEGA  20
     1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100)          GEGA  30
      COMMON     A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB          GEGA  40
      COMMON     NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTC,NIN1,NIN2,NIN3,NIGEGA  50
     1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KCGEGA  60
     2DB,KFIL,KLIN,KREN,KC,FII,PHI,PSI,PSI3,TB,TC,TC,EPSIL,EPSIL0,EPSIL2GEGA  70
      COMMON     KFONC,KGRAD,KCONT,KINV1,KINV2,KCCBA,KJACC,KMAX1,KMAX2,KGEGA  80
     1REN1,KREN2,KINV,KCCBA1,KREN11,KREN21                              GEGA  90
      COMMON     ICTREC,DELTFI,ETA,JKO,LC,YSCRT                         GEGA 100
      COMMON/AREA1/XT(4),XQ(4),XB(4),XE(4),AK(4),AK1(4),AK2(4),ST(4)
     1,SB(4),SO(4),TF(4),TRS(4),TMAX(4),MDO(4),NSTAGE
     2,CC1,CC2,DELTA
      REAL MDO
C.....FUNCTION DEFINITIONS
      TEMPC(T)=(5.0/9.0)*(T-32.0)
      CS(T)=14.652-0.41022*TEMPC(T)+0.79910E-2*TEMPC(T)**2-0.77774E-4
     1*TEMPC(T)**3
C.....END OF FUNCTION DEFINITIONS
      TMIX=ST(I)+XT(I)*(100.C-XC(I))
      STI1=XE(I)+(TMIX-XE(I))*EXP(-AK(I)*TTT)
      BOD=SB(I)+XB(I)*(100.C-XC(NSTAGE+I))
      AAK1=AK1(I)*(1.047)**(TEMPC(STI1)-20.C)
      AAK2=AK2(I)*(1.024)**(TEMPC(STI1)-20.C)
      DODEF=CS(TMIX)-SO(I)
      IF(DODEF.LE.0.0) DODEF=0.0
      FUNCT=CS(STI1)-(AAK1*BOD/(AAK2-AAK1))*(EXP(-AAK1*TTT)
     1-EXP(-AAK2*TTT))-DODEF*EXP(-AAK2*TTT)
      RETURN
      END
```

```
      SUBROUTINE INPUT
      COMMON/AREA1/XT(4),XQ(4),XB(4),XE(4),AK(4),AK1(4),AK2(4),ST(4)
     1,SB(4),SO(4),TF(4),TRS(4),TMAX(4),MDO(4),NSTAGE
     2,CC1,CC2,DELTA
      COMMON/AREA2/NTIMES(4),DEL(4),NF(30)
      REAL MDO
C.....NUMBER OF STAGES
      NSTAGE=4
      READ(5,1000) XT,XQ,XB,XE,AK,AK1,AK2,ST,SB,SC,TF,TRS,TMAX,MDO
      READ(5,2000) DELTA
C.....CONVERSION FACTORS
      CC1=1.0/(3.6E3*62.4262)
      CC2=0.453592E6/(8.64E4*28.31605)
C.....DETERMINE NUMBER OF ITERATIONS FOR FIBONNACI SEARCH
      NF(1)=1
      NF(2)=1
      DO70I=1,NSTAGE
      DO50J=3,30
      NTIMES(I)=J-1
      NF(J)=NF(J-1)+NF(J-2)
      AF1=NF(J-1)
      AF=NF(J)
      AF=1.0/AF
      IF((AF*TF(I)).LE.0.0001) GO TO 70
   50 CONTINUE
   70 DEL(I)=AF1*AF
      WRITE(6,3000) XT,XQ,XB,XE,AK,AK1,AK2,ST,SB,SO,TF,TRS,TMAX,MDO
      WRITE(6,4000) NSTAGE,DELTA,CC1,CC2
      WRITE(6,5000) NTIMES,DEL
      RETURN
 1000 FORMAT(4G10.6)
 2000 FORMAT(F10.0)
 3000 FORMAT(4(5X,E13.6))
 4000 FORMAT(5X,I5,3(5X,E13.6))
 5000 FORMAT(4(5X,I5)/4(5X,E13.6))
      END
```

```
PARAMETRES

NV          8
NIN        12
NEG         0
NEVL       20
NTO         6
ITET       20
ICONJ       1
IDIAG       0
ITMAX      50
KFIL        0
KLIN        0
NCO        20
ITSCR       1
ISOLSR     50
EPSIL      0.1E 00
EPSILO     0.1E-02
EPSIL1     0.1E-02
EPSIL2     0.1E-02
PC         0.0
      0.0                0.240000E 10       0.0                0.160000E 10
      0.820000E 03       0.870000E 03       0.885000E 03       0.920000E 03
      0.110000E 06       0.0                0.0                0.0
      0.807000E 02       0.807000E 02       0.807000E 02       0.807000E 02
      0.124000E 01       0.129000E 01       0.110000E 01       0.134000E 01
      0.100000E 01       0.100000E 01       0.100000E 01       0.100000E 01
      0.900000E 00       0.900000E 00       0.900000E 00       0.900000E 00
      0.807000E 02       0.0                0.0                0.0
      0.200000E 01       0.0                0.0                0.0
      0.670000E 01       0.0                0.0                0.0
      0.500000E-01       0.187000E 01       0.170000E 00       0.240000E 01
      0.100000E 02       0.100000E 02       0.100000E 02       0.100000E 02
      0.930000E 02       0.930000E 02       0.930000E 02       0.930000E 02
      0.300000E 01       0.300000E 01       0.400000E 01       0.400000E 01
         4       0.100000E-03       0.444970E-05       0.185404E 00
        14         22         17         22
      0.618033E 00       0.618034E 00       0.618034E 00       0.618034E 00
```

GRADIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES 8
NOMBRE TOTAL DE VARIABLES 24
NOMBRE DE CONTRAINTES 12

EPSILON DE NEWTON 0.1000E-02
EPSILON TEST GRADIENT 0.1000E-02

FONCTION ECONOMIQUE 0.0

	BORNE INFERIEURE	VARIABLE NATURELLE	BORNE SUPERIEURE
XI(1)	0.0	X(1) 0.0	XS(1) 0.100000000000000E 03
XI(2)	0.0	X(2) 0.0	XS(2) 0.100000000000000E 03
XI(3)	0.0	X(3) 0.0	XS(3) 0.100000000000000E 03
XI(4)	0.0	X(4) 0.0	XS(4) 0.100000000000000E 03
XI(5)	0.0	X(5) 0.0	XS(5) 0.100000000000000E 03
XI(6)	0.0	X(6) 0.0	XS(6) 0.100000000000000E 03
XI(7)	0.0	X(7) 0.0	XS(7) 0.100000000000000E 03
XI(8)	0.0	X(8) 0.0	XS(8) 0.100000000000000E 03

VALEUR DES CONTRAINTES

C(1) -0.100000000000000E 02
C(2) 0.227502822875 98E 01
C(3) -0.100000000000000E 02
C(4) -0.226139831547 97E 01
C(5) -0.123000030517 58E 02
C(6) -0.249786376953 13E-01
C(7) -0.112000579833 98E 02
C(8) -0.364906311035 16E 01
C(9) -0.201266002655 03E 01
C(10) 0.714672088623 05E 01
C(11) 0.301983165740 97E 01
C(12) 0.221012592315 67E 01

```
IT  1   PHI-0.7754716750000E 04   DIR.GRAD.   NC 10    YN 0.34E 03   DELTAFI 0.37E 05   ETA 0.33E 05   NCDB  0   NCN  2   NITN  2
IT  2   PHI-0.5899492187500E 04   DIR.CONJ.   LC  2                                                      NCDB  0   NCN  2   NITN  2
IT  3   PHI-0.4676492187500E 04   DIR.CONJ.   LC  2                                                      NCDB  0   NCN  2   NITN  2
IT  4   PHI-0.4325335937500E 04   DIR.GRAD.   NC 10    YN 0.29E 03   DELTAFI 0.26E 05   ETA 0.16E 05   NCDB  1   NCN  2   NITN  5
IT  5   PHI-0.2940133056640E 04   DIR.GRAD.   NC 11    YN 0.27E 04   DELTAFI 0.26E 05   ETA 0.27E 14   NCDB  1   NCN  4   NITN 25
IT  6   PHI-0.9695302734375E 03   DIR.GRAD.   NC 11    YN 0.16E 04   DELTAFI 0.26E 05   ETA 0.16E 14   NCDB  1   NCN  3   NITN 18
IT  7   PHI-0.1604821205139 2E 01   DIR.GRAD.   NC 11    YN 0.12E 03   DELTAFI 0.11E 05   ETA 0.11E 05   NCDB  1   NCN  2   NITN  6
LES  VARIABLES  ARTIFICIELLES  SONT  TOUTES  ANNULEES
       PHI-0.1604821205139 2E 01                        YN 0.0        DELTAFI 0.0        ETA 0.0


DUREE  DU  CALCUL   3035  CENTISECONDES
```

VARIABLE NATURELLE

X(1)	=	0.0	= XI(1)
X(2)	=	0.185419311523445 02	
X(3)	=	0.0	= XI(3)
X(4)	=	0.0	= XI(4)
X(5)	=	0.668959655761725 02	
X(6)	=	0.0	= XI(6)
X(7)	=	0.0	= XI(7)
X(8)	=	0.0	= XI(8)

VARIABLE DUALE ASSOCIEE

V(1)	=	0.0
V(2)	=	0.
V(3)	=	0.0
V(4)	=	-0.146944411C9856E-01
V(5)	=	0.
V(6)	=	0.0
V(7)	=	0.0
V(8)	=	0.0

CONTRAINTE

CONTRAINTE 1	-0.100000000000C0E 02
CONTRAINTE 2	-0.99945C68359375E-03
CONTRAINTE 3	-0.100000C0000C00E 02
CONTRAINTE 4	-0.226139831542976 01
CONTRAINTE 5	-0.123CCC0305175BE 02
CONTRAINTE 6	-0.230101013183595 01
CONTRAINTE 7	-0.114040069580C8E 02
CONTRAINTE 8	-0.381823730466756 01
CONTRAINTE 9	-0.309469604492196 01
CONTRAINTE 10	-0.995635986328136-03
CONTRAINTE 11	-0.115696048736576 01
CONTRAINTE 12	-0.147732925415046 01

VARIABLE DUALE ASSCCIEE

U(1)	=	0.0
U(2)	=	0.111C3844642639E C0
U(3)	=	0.0
U(4)	=	0.0
U(5)	=	0.0
U(6)	=	0.0
U(7)	=	0.0
U(8)	=	0.0
U(9)	=	0.0
U(10)	=	0.34952521324158E-01
U(11)	=	0.0
U(12)	=	0.0

```
NIVEAU  DE  SORTIE     1

NOMBRE  D ITERATIONS        7

NOMBRE  D APPELS  DU  S/P  CALCULANT  LES  CONTRAINTES    75

NOMBRE  D APPELS  DU  S/P  CALCULANT  LA  FONCTION  ECONOMIQUE    39

NOMBRE  D APPELS  DU  S/P  CALCULANT  LE  GRADIENT  DE  LA  FONCTION  ECONOMIQUE    15

NOMBRE  D APPELS  DU  S/P  CALCULANT  L INVERSE  AVEC  BASE  DONNEE     4
NOMBRE  D APPELS  DU  S/P  CALCULANT  L INVERSE  AVEC  RECHERCHE  DE  BASE     0

NOMBRE  D APPELS  DU  S/P  EFFECTUANT  LES  CHANGEMENTS  DE  BASE     4

NOMBRE  D APPELS  DU  S/P  CALCULANT  LE  JACOBIEN  DES  CONTRAINTES     9

NOMBRE  D APPELS  DU  S/P  EFFECTUANT  UNE  RECHERCHE  DE  MAX  SANS  DICHOTOMIE     7
NOMBRE  D APPELS  DU  S/P  EFFECTUANT  UNE  RECHERCHE  DE  MAX  AVEC  DICHOTOMIE     0

NOMBRE  D APPELS  DU  S/P  EFFECTUANT  LA  RENTREE  DANS  LE  DOMAINE    17                    NOMBRE  TOTAL  D ITERATIONS    60
```

PARAMETRES

```
NV          8
NIN        12
NEG         0
NEVL       20
NTO         6
ITET       20
ICONJ       1
IDIAG       0
ITMAX      50
KFIL        0
KLIA        0
NCO        20
ITSCR       1
ISOLSR     50
EPSIL     0.1E 00
EPSILO    0.1E-02
EPSIL1    0.1E-02
EPSILZ    0.1E-02
PC          0.0
      0.0                 0.240000E 10        0.0                 0.160000E 10
      0.820000E 03        0.870000E 03        0.885000E 03        0.920000E 03
      0.110000E 06        0.0                 0.0                 0.0
      0.807000E 02        0.807000E 02        0.807000E 02        0.807000E 02
      0.124000E 01        0.129000E 01        0.110000E 01        0.134000E 01
      0.100000E 01        0.100000E 01        0.100000E 01        0.100000E 01
      0.900000E 00        0.900000E 00        0.900000E 00        0.900000E 00
      0.807000E 02        0.0                 0.0                 0.0
      0.200000E 01        0.0                 0.0                 0.0
      0.670000E 01        0.0                 0.0                 0.0
      0.500000E-01        0.187000E 01        0.170000E 00        0.240000E 01
      0.100000E 02        0.100000E 02        0.100000E 02        0.100000E 02
      0.930000E 02        0.930000E 02        0.930000E 02        0.930000E 02
      0.300000E 01        0.300000E 01        0.400000E 01        0.400000E 01
         4        0.100000E-03        0.444970E-05        0.1654C4E C0
        14          22          17          22
      0.618033E 00        0.618034E 00        0.618034E 00        0.618034E 00
```

GRADIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES 8
NOMBRE TOTAL DE VARIABLES 20
NOMBRE DE CONTRAINTES 12

EPSILON DE NEWTON 0.1000E-02
EPSILON TEST GRADIENT 0.1000E-02

FONCTION ECONOMIQUE -0.819157409667976E 01

	BORNE INFERIEURE	VARIABLE NATURELLE	BORNE SUPERIEURE
XI(1)	0.0	X(1) 0.0	XS(1) 0.1000000000000E 03
XI(2)	0.0	X(2) 0.1000000000000E 03	XS(2) 0.1000000000000E 03
XI(3)	0.0	X(3) 0.0	XS(3) 0.1000000000000E 03
XI(4)	0.0	X(4) 0.1000000000000E 03	XS(4) 0.1000000000000E 03
XI(5)	0.0	X(5) 0.1000000000000E 03	XS(5) 0.1000000000000E 03
XI(6)	0.0	X(6) 0.0	XS(6) 0.1000000000000E 03
XI(7)	0.0	X(7) 0.0	XS(7) 0.1000000000000E 03
XI(8)	0.0	X(8) 0.0	XS(8) 0.1000000000000E 03

VALEUR DES CONTRAINTES

C(1)	-0.1000000000000E 02
C(2)	-0.1000000000000E 02
C(3)	-0.1000000000000E 02
C(4)	-0.1000000000000E 02
C(5)	-0.1230000030517578E 02
C(6)	-0.1230000030517578E 02
C(7)	-0.1230000030517578E 02
C(8)	-0.1230000030517578E 02
C(9)	-0.3630150794982910E 01
C(10)	-0.3423480987548828E 01
C(11)	-0.3196732521057129E 01
C(12)	-0.3280018806457520E 01

```
IT  1   PHI-0.2804905E9143185E 01   DIR.GRAD.  NC  5   YN 0.98E 00   DELTAFI 0.98E 02   ETA 0.98E 02   NCDB  0   NCN  2   NITN  2
IT  2   PHI-0.26766376499361E 01   DIR.CONJ.  LC  3                                                    NCDB  0   NCN  6   NITN  6
IT  3   PHI-0.26742133167725E 01   DIR.CONJ.  LC  3                                                    NCDB  0   NCN  2   NITN  2
IT  4   PHI-0.22251167297363E 01   DIR.CONJ.  LC  3                                                    NCDB  2   NCN 10   NITN 45
IT  5   PHI-0.16048078536987E 01   DIR.GRAD.  NC  6   YN 0.11E-01   DELTAFI 0.55E 00   ETA 0.32E 00   NCDB  1   NCN  7   NITN 26
        PHI-0.16048078536987E 01                      YN 0.0       DELTAFI 0.0        ETA 0.0

DUREE DU CALCUL   3541 CENTISECONDES
```

VARIABLE NATURELLE			VARIABLE DUALE ASSOCIEE	
X(1) =	0.0	= XI(1)	V(1) =	0.0
X(2) =	0.18541931152344E 02		V(2) =	0.
X(3) =	0.0	= XI(3)	V(3) =	0.0
X(4) =	0.0	= XI(4)	V(4) =	-0.14694441109896E-01
X(5) =	0.668919372558593 02		V(5) =	0.
X(6) =	0.0	= XI(6)	V(6) =	0.0
X(7) =	0.0	= XI(7)	V(7) =	0.0
X(8) =	0.0	= XI(8)	V(8) =	0.0

CONTRAINTE		VARIABLE DUALE ASSOCIEE	
CONTRAINTE 1	-0.10000000000000E 02	U(1) =	0.0
CONTRAINTE 2	-0.99945068359375E-03	U(2) =	0.11103844642639E 00
CONTRAINTE 3	-0.10000000000000E 02	U(3) =	0.0
CONTRAINTE 4	-0.22613983154297E 01	U(4) =	0.0
CONTRAINTE 5	-0.12300303051758E 02	U(5) =	0.0
CONTRAINTE 6	-0.23010101318359E 01	U(6) =	0.0
CONTRAINTE 7	-0.11404006958008E 02	U(7) =	0.0
CONTRAINTE 8	-0.38182373046875E 01	U(8) =	0.0
CONTRAINTE 9	-0.30946311950684E 01	U(9) =	0.0
CONTRAINTE 10	-0.57029724121094E-03	U(10) =	0.34952521324158E-01
CONTRAINTE 11	-0.11567068099976E 01	U(11) =	0.0
CONTRAINTE 12	-0.14771051406860E 01	U(12) =	0.0

```
NIVEAU DE SORTIE     1

NOMBRE D ITERATIONS        5

NOMBRE D APPELS DU S/P CALCULANT LES CONTRAINTES  105

NOMBRE D APPELS DU S/P CALCULANT LA FONCTION ECONOMIQUE    57

NOMBRE D APPELS DU S/P CALCULANT LE GRADIENT DE LA FONCTION ECONOMIQUE    21

NOMBRE D APPELS DU S/P CALCULANT L INVERSE AVEC BASE CONNEE     4
NOMBRE D APPELS DU S/P CALCULANT L INVERSE AVEC RECHERCHE DE BASE     0

NOMBRE D APPELS DU S/P EFFECTUANT LES CHANGEMENTS DE BASE     3

NOMBRE D APPELS DU S/P CALCULANT LE JACOBIEN DES CONTRAINTES     9

NOMBRE D APPELS DU S/P EFFECTUANT UNE RECHERCHE DE MAX SANS DICHOTOMIE     5
NOMBRE D APPELS DU S/P EFFECTUANT UNE RECHERCHE DE MAX AVEC DICHOTOMIE     0

NOMBRE D APPELS DU S/P EFFECTUANT LA RENTREE DANS LE DOMAINE    27                    NOMBRE TOTAL D ITERATIONS    81
```

OPTIMIZATION OF INDUSTRIAL SYSTEMS WITH THE SEPARABLE PROGRAMMING AND THE GENERALIZED REDUCED GRADIENT METHODS

by

JEREL L. WILLIAMS

B.A., KANSAS STATE TEACHERS COLLEGE, EMPORIA, 1970

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1972

This thesis reviews two nonlinear programming methods and their application to industrial systems. Separable programming is described and exemplified. The same procedure is presented that is used in the separable programming subroutine of the Mathematical Programming System/360 (MPS/360). A supplemental FORTRAN program for assisting the usage of MPS/360 when solving separable programming problems is presented. An interesting application of separable programming to solve the geometric programming dual problem with N-degrees of difficulty is also exhibited. The second technique considered is the generalized reduced gradient method. The mathematical theory is presented and numerical examples are used to exemplify it. Its application via the GREG program is evaluated, and numerous computer examples are worked. The study is concluded with the application of the technique to optimize a water quality control model.