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:

[Signature]
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 THROUGHOUT ITS 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

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACKNOWLEDGEMENT</td>
<td>ii</td>
</tr>
<tr>
<td>1</td>
<td>CHAPTER 1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>CHAPTER 2 SEPARABLE PROGRAMMING</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2.1 INTRODUCTION</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>2.2 AN APPROXIMATION OF SEPARABLE FUNCTIONS</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>2.3 THE SEPARABLE PROGRAMMING PROBLEM</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>2.4 A REVIEW OF IBM MPS/360 VERSION 2, LINEAR AND SEPARABLE PROGRAMMING</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>2.5 A FORTRAN SUBROUTINE FOR GENERATING SEPARABLE PROGRAMMING DATA TO BE USED WITH MPS/360</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>2.6 SOLUTION TO A RELIABILITY PROBLEM</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>2.7 THE APPLICATION OF THE SEPARABLE PROGRAMMING ALGORITHM TO THE GEOMETRIC PROGRAMMING PROBLEM</td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>2.8 CONCLUDING REMARKS</td>
<td>57</td>
</tr>
<tr>
<td>3</td>
<td>CHAPTER 3 INTRODUCTION TO THE GENERALIZED REDUCED GRADIENT METHOD</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td>3.1 INTRODUCTION</td>
<td>61</td>
</tr>
<tr>
<td></td>
<td>3.2 DEFINITION OF NOTATION</td>
<td>62</td>
</tr>
<tr>
<td></td>
<td>3.3 STATEMENT OF THE PROBLEM</td>
<td>63</td>
</tr>
<tr>
<td></td>
<td>3.4 THE PRINCIPLE OF THE REDUCED GRADIENT</td>
<td>64</td>
</tr>
<tr>
<td></td>
<td>3.5 THE GENERALIZED REDUCED GRADIENT ALGORITHM</td>
<td>72</td>
</tr>
<tr>
<td></td>
<td>3.6 A NUMERICAL EXAMPLE</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>3.7 THE PSUEDO-NEWTON METHOD AND CHANGING THE BASIS</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>3.8 CONCLUDING REMARKS</td>
<td>96</td>
</tr>
<tr>
<td>4</td>
<td>CHAPTER 4 INTRODUCTION TO THE GREG PROGRAM</td>
<td>97</td>
</tr>
<tr>
<td></td>
<td>4.1 INTRODUCTION</td>
<td>98</td>
</tr>
<tr>
<td></td>
<td>4.2 THE GREG USER SUPPLIED SUBROUTINES</td>
<td>99</td>
</tr>
<tr>
<td></td>
<td>4.3 INPUT DATA FOR THE GREG PROGRAM</td>
<td>108</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.4</td>
<td>JOB CONTROL LANGUAGE FOR UTILIZING THE GREG PROGRAM</td>
<td>112</td>
</tr>
<tr>
<td>4.5</td>
<td>INTERPRETING THE GREG OUTPUT</td>
<td>112</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>NUMERICAL TESTING OF THE GREG PROGRAM</td>
<td>129</td>
</tr>
<tr>
<td>5.1</td>
<td>INTRODUCTION</td>
<td>130</td>
</tr>
<tr>
<td>5.2</td>
<td>THE EXAMPLE PROBLEM OF CHAPTER THREE</td>
<td>130</td>
</tr>
<tr>
<td>5.3</td>
<td>RELIABILITY OF A COMPLEX SYSTEM</td>
<td>131</td>
</tr>
<tr>
<td>5.4</td>
<td>OPTIMUM RELIABILITY OF A MULTISTAGE PARALLEL SYSTEM</td>
<td>135</td>
</tr>
<tr>
<td>5.5</td>
<td>TWO PROBLEMS FROM THE COLVILLE STUDY</td>
<td>138</td>
</tr>
<tr>
<td>5.6</td>
<td>SOLUTION TO THE GEOMETRIC PROGRAMMING EXAMPLE OF CHAPTER TWO</td>
<td>143</td>
</tr>
<tr>
<td>5.7</td>
<td>CONCLUDING REMARKS</td>
<td>144</td>
</tr>
<tr>
<td>Chapter 6</td>
<td>APPLICATION OF THE GREG METHOD TO A COMPLEX WATER QUALITY</td>
<td>145</td>
</tr>
<tr>
<td>6.1</td>
<td>CONTROL SYSTEM</td>
<td>146</td>
</tr>
<tr>
<td>6.2</td>
<td>INTRODUCTION TO THE WATER QUALITY MANAGEMENT PROBLEM</td>
<td>147</td>
</tr>
<tr>
<td>6.3</td>
<td>DO, BOD, AND TEMPERATURE RELATIONSHIPS IN FLOWING</td>
<td>147</td>
</tr>
<tr>
<td>6.4</td>
<td>STREAMS</td>
<td>147</td>
</tr>
<tr>
<td>6.5</td>
<td>DEFINITION OF THE N-STAGE RIVER BASIN MODEL</td>
<td>152</td>
</tr>
<tr>
<td>6.6</td>
<td>FORMULATION OF THE MATHEMATICAL OPTIMIZATION PROBLEM FOR THE GREG</td>
<td>157</td>
</tr>
<tr>
<td></td>
<td>PROGRAM</td>
<td>157</td>
</tr>
<tr>
<td></td>
<td>APPLICATION TO THE CHATTAHOOCHEE RIVER BASIN</td>
<td>162</td>
</tr>
<tr>
<td></td>
<td>CONCLUDING REMARKS</td>
<td>170</td>
</tr>
<tr>
<td>References</td>
<td></td>
<td>174</td>
</tr>
<tr>
<td>Appendix 1</td>
<td>COMPUTER PROGRAM FOR GENERATING SEPARABLE PROGRAMMING DATA</td>
<td>178</td>
</tr>
<tr>
<td>Appendix 2</td>
<td>COMPUTER OUTPUT FROM THE RELIABILITY PROBLEM</td>
<td>182</td>
</tr>
<tr>
<td>Appendix 3</td>
<td>GEOMETRIC PROGRAMMING USING MPS/360</td>
<td>194</td>
</tr>
<tr>
<td>Appendix 4</td>
<td>GREG PROGRAMMING MATERIAL FOR CHAPTER FIVE</td>
<td>206</td>
</tr>
<tr>
<td>Appendix 5</td>
<td>GREG PROGRAMMING MATERIAL FOR CHAPTER SIX</td>
<td>245</td>
</tr>
</tbody>
</table>
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 analyst, 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

\[ \phi(\mathbf{x}) = \sum_{i=1}^{m} h_i(x_i) \]  

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

\[ c(\mathbf{x}) = \sum_{i=1}^{m} f_i(x_i) \]  

subject to the constraints

\[ \sum_{i=1}^{m} g_{k,i}(x_i) \leq b_k, \quad k=1,\ldots,p \]  

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 by 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

$$\Delta x_i^j = x_i^j - x_i^{j-1}, \quad j = 1, 2, \ldots, n_i$$

$$\Delta f_i^j = f_i(x_i^j) - f_i(x_i^{j-1}), \quad i = 1, 2, \ldots, m$$

where the subscript refers to a function and/or variable such as $x_i$, $f_i(x_i)$, and $g_{k1}(x_i)$, 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, $D_i^j$, $j=1, 2, \ldots, n_i$, 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.
Fig. 2.1. Linear approximation of $f_i(x_i)$. 

\[
\begin{align*}
\Delta x_i &= x_i - x_i^0 \\
\Delta x_i^2 &= x_i^2 - x_i^1 \\
\Delta x_i^3 &= x_i^3 - x_i^2 \\
\Delta x_i^4 &= x_i^4 - x_i^3
\end{align*}
\]

\[
\begin{align*}
\Delta f_i^1 &= f_i(x_i^1) - f_i(x_i^0) \\
\Delta f_i^2 &= f_i(x_i^2) - f_i(x_i^1) \\
\Delta f_i^3 &= f_i(x_i^3) - f_i(x_i^2) \\
\Delta f_i^4 &= f_i(x_i^4) - f_i(x_i^3)
\end{align*}
\]
(i) \(0 \leq D_i^j \leq 1\) iff \(D_i^j = 0, j = 2, 3, \ldots, n_i\)

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

and

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

(iii) \(0 \leq D_i^{n_i}\) 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 \(D_i^k\) variables \((k = 1, \ldots, j-1)\) must have a value of one and all following values \((k = j+1, \ldots, n_i)\) must be zero.

**Example 1**

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

<table>
<thead>
<tr>
<th>(D_i^1)</th>
<th>(D_i^2)</th>
<th>(D_i^3)</th>
<th>(D_i^4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>1/4</td>
<td>0</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
<td>1</td>
<td>7/8</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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
\[ x_i = \Delta x_i^1 D_i^1 + \Delta x_i^2 D_i^2 + \Delta x_i^3 D_i^3 + \Delta x_i^4 D_i^4 \]  \hspace{1cm} (5) 

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

\[ f_i \approx \Delta f_i^1 D_i^1 + \Delta f_i^2 D_i^2 + \Delta f_i^3 D_i^3 + \Delta f_i^4 D_i^4 \]  \hspace{1cm} (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 \( S = \{0, 1/2, 1, 2\} \).

\[
\begin{align*}
  x_i^0 &= 0 & f_i(x_i^0) &= 0 \\
  x_i^1 &= 1/2 & f_i(x_i^1) &= 1/8 \\
  x_i^2 &= 1 & f_i(x_i^2) &= 1 \\
  x_i^3 &= 2 & f_i(x_i^3) &= 8 \\
\end{align*}
\]

The differences can now be calculated.

\[
\begin{array}{c|ccc}
  i & 1 & 2 & 3 \\
  \hline
  \Delta x_i^1 & 1/2 & 1/2 & 1 \\
  \Delta f_i^1 & 1/8 & 7/8 & 7 \\
\end{array}
\]

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

\[
f_i = 1/8 D_i^1 + 7/8 D_i^2 + 7 \cdot D_i^3
\]
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

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

and

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

(7)

(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

$$x_i^0 = 0 \quad \quad f_i(x_i^0) = 1$$

$$x_i^1 = 1 \quad \quad f_i(x_i^1) = 2.7183$$

$$x_i^2 = 2 \quad \quad f_i(x_i^2) = 7.3891$$

The differences are now calculated.

<table>
<thead>
<tr>
<th>$j$</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta x_i^j$</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>$\Delta f_i^j$</td>
<td>1.7183</td>
<td>4.6708</td>
</tr>
<tr>
<td>(x_i) (given)</td>
<td>(D_i^1)</td>
<td>(D_i^2)</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(\frac{3}{4})</td>
<td>1</td>
<td>(\frac{1}{2})</td>
</tr>
<tr>
<td>(\frac{3}{2})</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

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
\[ e_i \leq 1.0 + 1.7183 \, D_i^1 + 4.6708 \, D_i^2 \]
Now consider a constraint of the form
\[ e_i \leq b_k \]
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)
\[ c(\bar{x}) = \sum_{i=1}^{m} f_i(x_i) \] (2)
subject to the constraints
\[ \sum_{i=1}^{m} g_{ki} (x_i) \leq b_k, \quad k = 1, 2, \ldots, p \] (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
\[ x_i = x_i^0 + \sum_{j=1}^{n_i} \Delta x_i^j d_i^j \]  
\[ f_i(x_i) = f_i(x_i^0) + \sum_{j=1}^{n_i} \Delta f_i^j d_i^j \]  
\[ g_{ki}(x_i) = g_{ki}(x_i^0) + \sum_{j=1}^{n_i} \Delta g_{ki}^j d_i^j \]  
\[ k = 1, 2, \ldots, p, \text{ and } i = 1, 2, \ldots, m \]

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

\[ c = \sum_{i=1}^{m} \left( \sum_{j=1}^{n_i} \Delta f_i^j d_i^j \right) + \sum_{i=1}^{m} f_i(x_i^0) \]  

subject to the constraints

\[ \sum_{i=1}^{m} \sum_{j=1}^{n_i} \Delta g_{ki}^j d_i^j \leq b_k - \sum_{i=1}^{m} g_{ki}(x_i^0), \quad k = 1, 2, \ldots, p \]  
\[ x_i - \sum_{j=1}^{n_i} \Delta x_i^j d_i^j = x_i^0, \quad i = 1, 2, \ldots, m \]  
\[ 0 \leq d_i^j \leq 1, \quad j = 1, 2, \ldots, n_i \]  
\[ x_i \geq 0 \]

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 functions, \( d_i^j \leq 1 \), 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

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

$$x_1 \geq 0, \quad x_2 \geq 0$$

The functions of this problem are separable.

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

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

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

<table>
<thead>
<tr>
<th>j</th>
<th>( x_j^1 )</th>
<th>( f_j^1 )</th>
<th>( g_{11}^j )</th>
<th>( x_j^2 )</th>
<th>( f_2^j )</th>
<th>( g_{12}^j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{7}{16} )</td>
<td>( \frac{1}{16} )</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{3}{16} )</td>
<td>( \frac{1}{8} )</td>
</tr>
<tr>
<td>2</td>
<td>( \frac{1}{2} )</td>
<td>( \frac{3}{4} )</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{1}{2} )</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{1}{2} )</td>
</tr>
<tr>
<td>3</td>
<td>( \frac{3}{4} )</td>
<td>( \frac{15}{16} )</td>
<td>( \frac{9}{16} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Function differences

<table>
<thead>
<tr>
<th>j</th>
<th>( \Delta x_j^1 )</th>
<th>( \Delta f_j^1 )</th>
<th>( \Delta g_{11}^j )</th>
<th>( \Delta x_j^2 )</th>
<th>( \Delta f_2^j )</th>
<th>( \Delta g_{12}^j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{7}{16} )</td>
<td>( \frac{1}{16} )</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{3}{16} )</td>
<td>( \frac{1}{8} )</td>
</tr>
<tr>
<td>2</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{5}{16} )</td>
<td>( \frac{3}{16} )</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{1}{16} )</td>
<td>( \frac{3}{8} )</td>
</tr>
<tr>
<td>3</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{3}{16} )</td>
<td>( \frac{5}{16} )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( \frac{1}{4} )</td>
<td>( \frac{1}{16} )</td>
<td>( \frac{7}{16} )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2. The function values and differences for example 4.
\[ f_1(x_1) = \frac{7}{16} D_1^1 + \frac{5}{16} D_1^2 + \frac{3}{16} D_1^3 + \frac{1}{16} D_1^4 \]

\[ f_2(x_2) = \frac{3}{16} D_2^1 + \frac{1}{16} D_2^2 \]

\[ g_{11}(x_1) = \frac{1}{16} D_1^1 + \frac{3}{16} D_1^2 + \frac{5}{16} D_1^3 + \frac{7}{16} D_1^4 \]

\[ g_{12}(x_2) = \frac{1}{8} D_2^1 + \frac{3}{8} D_2^2 \]

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 \( D_1^j \leq 1 \). This series of constraints must be incorporated in the final L.P. problem. The final problem set-up is maximize

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

subject to the constraints

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

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

\[ x_2 - \frac{1}{4} D_2^1 - \frac{1}{4} D_2^2 = 0 \]
\[ 0 \leq D^j_i \leq 1, \quad i = 1, 2, \quad j = 1, \ldots, n_1 \]

\[ x_i \geq 0 \]

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, \( D^j_i \leq 1 \). Also notice that it is imperative that the first iteration involves \( D^1_1 \) or \( D^1_2 \). If the first iteration involves \( D^1_1 \) then the second iteration can involve \( D^2_2 \), or \( D^2_1 \) 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)
<table>
<thead>
<tr>
<th>$c_i$</th>
<th>$b_i$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$D_1$</th>
<th>$D_2$</th>
<th>$D_3$</th>
<th>$D_4$</th>
<th>$D_5$</th>
<th>$D_6$</th>
<th>$D_7$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
<th>$S_6$</th>
<th>$S_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$S_1$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\frac{1}{16}$</td>
<td>$\frac{3}{16}$</td>
<td>$\frac{5}{16}$</td>
<td>$\frac{7}{16}$</td>
<td>$\frac{1}{8}$</td>
<td>$\frac{3}{8}$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>$x_1$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$-\frac{1}{4}$</td>
<td>$-\frac{1}{4}$</td>
<td>$-\frac{1}{4}$</td>
<td>$-\frac{1}{4}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>$x_2$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$-\frac{1}{4}$</td>
<td>$-\frac{1}{4}$</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>$S_2$</td>
<td>1</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>$S_3$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>$S_4$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>$S_5$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>$S_6$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>$S_7$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

$c = 0$

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    0    0

Table 2.3. Starting simplex tableau. (All blank spaces are zero.)
<table>
<thead>
<tr>
<th>Iteration Number</th>
<th>Vector into Basis</th>
<th>Vector out of Basis</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$D_1^1$</td>
<td>$S_2$</td>
</tr>
<tr>
<td>2</td>
<td>$D_1^2$</td>
<td>$S_3$</td>
</tr>
<tr>
<td>3</td>
<td>$D_2^1$</td>
<td>$S_6$</td>
</tr>
<tr>
<td>4</td>
<td>$D_1^3$</td>
<td>$S_4$</td>
</tr>
<tr>
<td>5</td>
<td>$D_2^2$</td>
<td>$S_1$</td>
</tr>
</tbody>
</table>

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 arithmetic 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 "INITIAL2" 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 CCOMPILER - MPS/360 V2-M9

CC01       PROGRAM
CC02       INITIALZ
CC64       MOVE(XDATA,'DATA-SET')
CC65       MOVE(XPNAMEx,'EXAMPLE')
CC66       CCNVERT
CC67       ECDOUT
CC68       MOVE(XCBIx,'OBJECT')
CC69       MOVE(XRHS,'LIMITS')
CC70       TITLE('EXAMPLE')
CC71       SETUP('BOUNCE','SEPBND','MAX')
CC72       PRIMAL
CC73       SOLUTION
CC74       EXIT
CC75       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 SEPBBOUND 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.(101,754),(994,976)

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 \( y = 3x_1 + x_2 \).
<table>
<thead>
<tr>
<th>Columns</th>
<th>Field 1</th>
<th>Field 2</th>
<th>Field 3</th>
<th>Field 4</th>
<th>Field 5</th>
<th>Field 6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2-3</td>
<td>5-12</td>
<td>15-22</td>
<td>25-36</td>
<td>40-47</td>
<td>50-61</td>
</tr>
<tr>
<td>Type of Contents</td>
<td>code</td>
<td>name</td>
<td>name</td>
<td>value</td>
<td>name</td>
<td>value</td>
</tr>
</tbody>
</table>

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, \(<\), \(\geq\), 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.
<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>2</th>
<th>3</th>
<th>3</th>
<th>4</th>
<th>4</th>
<th>5</th>
<th>5</th>
<th>5</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**NAME**

- 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.
<table>
<thead>
<tr>
<th>NAME</th>
<th>DATA-SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROWS</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>OBJECT</td>
</tr>
<tr>
<td>L</td>
<td>ROW1</td>
</tr>
<tr>
<td>E</td>
<td>GRID1</td>
</tr>
<tr>
<td>E</td>
<td>GRID2</td>
</tr>
<tr>
<td>COLS</td>
<td></td>
</tr>
<tr>
<td>X1</td>
<td>GRID1</td>
</tr>
<tr>
<td>X2</td>
<td>GRID2</td>
</tr>
<tr>
<td>SET1</td>
<td>'MARKER'</td>
</tr>
<tr>
<td>D11</td>
<td>OBJECT</td>
</tr>
<tr>
<td>D11</td>
<td>GRID1</td>
</tr>
<tr>
<td>D12</td>
<td>OBJECT</td>
</tr>
<tr>
<td>D12</td>
<td>GRID1</td>
</tr>
<tr>
<td>D13</td>
<td>OBJECT</td>
</tr>
<tr>
<td>D13</td>
<td>GRID1</td>
</tr>
<tr>
<td>D14</td>
<td>OBJECT</td>
</tr>
<tr>
<td>D14</td>
<td>GRID1</td>
</tr>
<tr>
<td>SET2</td>
<td>'MARKER'</td>
</tr>
<tr>
<td>D21</td>
<td>OBJECT</td>
</tr>
<tr>
<td>D21</td>
<td>GRID2</td>
</tr>
<tr>
<td>D22</td>
<td>OBJECT</td>
</tr>
<tr>
<td>D22</td>
<td>GRID2</td>
</tr>
<tr>
<td>ENCSET</td>
<td>'MARKER'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RHS</th>
<th>LIMITS</th>
<th>ROW1</th>
<th>1.00000</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOUNDS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UP SEPBOUND D11</td>
<td>1.00000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UP SEPBOUND D12</td>
<td>1.00000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UP SEPBOUND D13</td>
<td>1.00000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UP SEPBOUND D14</td>
<td>1.00000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UP SEPBOUND D21</td>
<td>1.00000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UP SEPBOUND D22</td>
<td>1.00000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**Section 1 - Rows**

<table>
<thead>
<tr>
<th>Number</th>
<th>Activity</th>
<th>Slack Activity</th>
<th>Lower Limit</th>
<th>Upper Limit</th>
<th>Dual Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BS</td>
<td>1.17768</td>
<td>1.17768</td>
<td>NONE</td>
<td>1.00000</td>
</tr>
<tr>
<td>2</td>
<td>LL</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
</tr>
<tr>
<td>A</td>
<td>EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
</tr>
<tr>
<td>A</td>
<td>EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
</tr>
</tbody>
</table>

**Example**

**Section 2 - Columns**

<table>
<thead>
<tr>
<th>Number</th>
<th>Activity</th>
<th>Input Cost</th>
<th>Lower Limit</th>
<th>Upper Limit</th>
<th>Reduced Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>X1</td>
<td>75000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
</tr>
<tr>
<td>6</td>
<td>X2</td>
<td>45000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
</tr>
<tr>
<td>7</td>
<td>Ch1</td>
<td>43750</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
</tr>
<tr>
<td>8</td>
<td>Ch2</td>
<td>31250</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
</tr>
<tr>
<td>9</td>
<td>Ch3</td>
<td>18750</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
</tr>
<tr>
<td>10</td>
<td>Ch4</td>
<td>6250</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
</tr>
<tr>
<td>11</td>
<td>Ch21</td>
<td>18750</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
</tr>
<tr>
<td>12</td>
<td>Ch22</td>
<td>6250</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
</tr>
</tbody>
</table>
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 $\Delta x^j_i$, $\Delta f^j_i$, and $\Delta g^j_k$ 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 $x^0_i$, the starting point. The next three parameters are used to partition a predetermined feasible region uniformly. A partition is automatically created between $x^0_i$ 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 $x^0_i$, 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, $\Delta f^j_i$ or $\Delta g^j_k$, which are the coefficients for the special separable variables, $D^j_i$. 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.
calculate the function value at point j.

calculate the difference of $\Delta f_i^j$ or $\Delta g_i^j$
for partition j and variable i.

Yes

are there more functions?

calculate the grid value, $-\Delta x_i^j$

PUNCH:
$\Delta f_i^j, \Delta g_i^j, -\Delta x_i^j$
for partition j
for variable i
$k = 1, \ldots, P$

are there more partitions?

are there more variables?

PUNCH:
the upper bounds data

STOP

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 columns represent functions 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 \( \text{FUNC} \) is called by the main program it is supplied with the desired row and column number. \( \text{FUNC} \) 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 

\[
(J - 1) \ast \text{NROWS} + I,
\]

where \( I \) is the row, \( J \) is the column, and \( \text{NROWS} \) 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, \( x \). \( \text{FUNC} \) returns the desired function value. Figure 2.12 exemplifies the construction of \( \text{FUNC} \) 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 \( \text{NFUN} \), the number of variables, and \( \text{NROWS} \), 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 \( \text{START} \), \( \text{INTER} \), \( \text{ALOW} \), and \( \text{UPPER} \). \( \text{START} \) is the initial starting point, \( x^0_1 \). Its value is usually zero unless the user is working with a function not defined at
<table>
<thead>
<tr>
<th>row 1</th>
<th>column 1</th>
<th>column 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$F(1, 1) = 2x_1 - (x_1)^2$</td>
<td>$F(1, 2) = x_2 - (x_2)^2$</td>
</tr>
<tr>
<td>row 2</td>
<td>$F(2, 1) = (x_1)^2$</td>
<td>$F(2, 2) = 2(x_2)^2$</td>
</tr>
</tbody>
</table>

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 \( \leq \text{ALOW} < \text{UPPER} \). 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, \( D^I_1 \), 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.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>row 1</td>
<td>ROW101</td>
</tr>
<tr>
<td>row 2</td>
<td>ROW102</td>
</tr>
<tr>
<td>row 3</td>
<td>ROW103</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>row N</td>
<td>ROW100+N</td>
</tr>
</tbody>
</table>

\( N = \# \text{matrix rows} = \text{NROWS} \)
<table>
<thead>
<tr>
<th>SYMBOLS</th>
<th>EXPLANATION</th>
<th>FORMAT</th>
</tr>
</thead>
<tbody>
<tr>
<td>NFUN</td>
<td>Number of separable variables</td>
<td>(2T5)</td>
</tr>
<tr>
<td>NROWS</td>
<td>Number of matrix ( F(I,J) ) rows</td>
<td></td>
</tr>
<tr>
<td>START</td>
<td>Initial starting point ( X_0 )</td>
<td></td>
</tr>
<tr>
<td>INTER</td>
<td>Number of uniform partitions between ALOW and UPPER</td>
<td>(F10.0,110,2F10.0)</td>
</tr>
<tr>
<td>ALOW</td>
<td>Lower bound of the uniform region</td>
<td></td>
</tr>
<tr>
<td>UPPER</td>
<td>Upper bound of the uniform region</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.5. Input data formats for the FORTRAN program.
<table>
<thead>
<tr>
<th>Card 1</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NFUN</td>
<td>NROWS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Card 2</td>
<td>0.0</td>
<td>4</td>
<td>0.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>Card 3</td>
<td>0.0</td>
<td>2</td>
<td>0.0</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>START</td>
<td>INTER</td>
<td>ALOW</td>
<td>UPPER</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2.13. Sample input data to generate separable data for Example 4.
<table>
<thead>
<tr>
<th>SET101</th>
<th>'MARKER'</th>
<th>'SEPORG'</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1001</td>
<td>ROW101</td>
<td>0.43750</td>
</tr>
<tr>
<td>P1001</td>
<td>GRID101</td>
<td>-0.25000</td>
</tr>
<tr>
<td>P1002</td>
<td>ROW101</td>
<td>0.31250</td>
</tr>
<tr>
<td>P1002</td>
<td>GRID101</td>
<td>-0.25000</td>
</tr>
<tr>
<td>P1003</td>
<td>ROW101</td>
<td>0.18750</td>
</tr>
<tr>
<td>P1003</td>
<td>GRID101</td>
<td>-0.25000</td>
</tr>
<tr>
<td>P1004</td>
<td>POW101</td>
<td>0.06250</td>
</tr>
<tr>
<td>P1004</td>
<td>GRID101</td>
<td>-0.25000</td>
</tr>
<tr>
<td>SET102</td>
<td>'MARKER'</td>
<td>'SEPORG'</td>
</tr>
<tr>
<td>P1005</td>
<td>ROW101</td>
<td>0.18750</td>
</tr>
<tr>
<td>P1005</td>
<td>GRID102</td>
<td>-0.25000</td>
</tr>
<tr>
<td>P1006</td>
<td>POW101</td>
<td>0.06250</td>
</tr>
<tr>
<td>P1006</td>
<td>GRID102</td>
<td>-0.25000</td>
</tr>
</tbody>
</table>

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 FORNFLN & NROWS
NFLN = 2
NROWS = 2

ECHO CHECK FOR VARIOUS 1
START = C*O
INTER = 4
ALOW = C*C
UPPER = 1.0C

ECHO CHECK FOR VARIOUS 2
START = C*C
INTER = 2
ALCW = C*C
UPPER = 0.5C

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

\[
\begin{align*}
\text{variable 1} & \quad \text{GRID101} \\
\text{variable 2} & \quad \text{GRID102} \\
\vdots & \quad \\
\vdots & \quad \\
\vdots & \quad \\
\text{variable } m & \quad \text{GRID100+m} \\
\end{align*}
\]

\[m = \# \text{ variables} = \text{NFUN}\]

The special variables \((D^i_1)\) 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.

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

\(P = \text{number of matrix rows (NROWS)}\)

\(n_i = \text{number of partitions for variable } i \text{ (either INTER or INTER+1)}\)

\(m = \text{number of variables used in separable functions (NFUN)}\)

\(* = \text{rounded up to the nearest integer.}\)

There are minor limitations on this program. They are

\(\text{NFUN} \leq 899, \text{NROWS} \leq 899\)

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 \( \text{Row} (50) \), and the possibility 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

\[
S = \ln(1 - 0.2x_1) + \ln(1 - 0.15x_2) + \ln(1 - 0.10x_3) + \ln(1 - 0.35x_4) + \ln(1 - 0.25x_5)
\]

subject to the following constraints.

\[
\begin{align*}
x_1^2 + 2x_2^2 + 3x_3^2 + 4x_4^2 + 2x_5^2 & \leq 110 \\
\end{align*}
\]
CONTROL PROGRAM COMPILER - MPS/360 V2-M8

C001   PPORGAM
C002   INITIAL
C064   MOVE(XDATA,'DATA-SET')
C065   MOVE(XPRNAME,'EXAMPLE')
C066   CONVERT
C067   DDCCUT
C068   MOVE(XOBJ,'ROW101')
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 =

**NAME** | **ACTIVITY** | DEFINED AS
--- | --- | ---
FUNCTIONAL | 1.17708 | RCW101
RESTRAINTS | | LIMITS
Rounds... | | SEPACOUND

---

**EXAMPLE**

**SECTION 1 - ROWS**

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>RCK</th>
<th>AT ACTIVITY</th>
<th>SLACK ACTIVITY</th>
<th>LOWER LIMIT</th>
<th>UPPER LIMIT</th>
<th>DUAL ACTIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RCW101</td>
<td>PS</td>
<td>1.17708</td>
<td>1.17708</td>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>2</td>
<td>RCW102</td>
<td>LL</td>
<td>1.00000</td>
<td></td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>3</td>
<td>GRIC101</td>
<td>EC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>4</td>
<td>GRIC102</td>
<td>EC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**EXAMPLE**

**SECTION 2 - COLUMNS**

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>COLUMN</th>
<th>AT ACTIVITY</th>
<th>INPUT COST</th>
<th>LOWER LIMIT</th>
<th>UPPER LIMIT</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>X1</td>
<td>ES</td>
<td>0.75000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>X2</td>
<td>ES</td>
<td>0.45633</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PICC1</td>
<td>LL</td>
<td>1.00000</td>
<td>0.43750</td>
<td></td>
<td>1.00000</td>
</tr>
<tr>
<td>8</td>
<td>PICC2</td>
<td>LL</td>
<td>1.00000</td>
<td>0.31250</td>
<td></td>
<td>1.00000</td>
</tr>
<tr>
<td>9</td>
<td>PICC3</td>
<td>LL</td>
<td>1.00000</td>
<td>0.18750</td>
<td></td>
<td>1.00000</td>
</tr>
<tr>
<td>10</td>
<td>PICC4</td>
<td>LL</td>
<td></td>
<td>0.62500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>PICC5</td>
<td>LL</td>
<td>1.00000</td>
<td>0.18750</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>PICC6</td>
<td>ES</td>
<td>0.33333</td>
<td>0.62500</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---
\[ 7(x_1 + e^{x_1/4}) + 7(x_2 + e^{x_2/4}) + 5(x_3 + e^{x_3/4}) + 9(x_4 + e^{x_4/4}) \\
+ 4(x_5 + e^{x_5/4}) \leq 175 \]
\[ 7(x_1 e^{x_1/4} + 8x_2 e^{x_2/4} + 8x_3 e^{x_3/4} + 6x_4 e^{x_4/4} + 9x_5 e^{x_5/4} \leq 200 \]

Notice that \( \text{START}(x_1^0) \) cannot equal zero for any \( x_1 \) 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 \( \text{START} = 1.0 \) 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 \text{FUNC} \text{is} \text{given} \text{in} \text{Appendix} \text{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 existence 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

ECHC CHECK FOR VARIABLE 1
START = 1.00
INTER = 10
ALLOW = 2.00
UPPER = 3.00

ECHC CHECK FOR VARIABLE 2
START = 1.00
INTER = 10
ALLOW = 2.00
UPPER = 3.00

ECHC CHECK FOR VARIABLE 3
START = 1.00
INTER = 10
ALLOW = 1.50
UPPER = 2.50

ECHC CHECK FOR VARIABLE 4
START = 1.00
INTER = 10
ALLOW = 3.00
UPPER = 4.00

ECHC CHECK FOR VARIABLE 5
START = 1.00
INTER = 10
ALLOW = 2.50
UPPER = 3.50

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

\[ y_0 = \sum_{t=1}^{T_0} \sigma_{0t} c_{0t} \prod_{n=1}^{N} x_{n}^{a_{0tn}}, \sigma_{0t} = \pm 1, c_{0t} > 0 \]

subject to the constraints

\[ y_m = \sum_{t=1}^{T_m} \sigma_{mt} c_{mt} \prod_{n=1}^{N} x_{n}^{a_{mtn}} \leq \sigma_{m} = \pm 1 \]

\[ \sigma_{mt} = \pm 1, \quad m = 1, \ldots, M \]

\[ c_{mt} > 0, \quad x_{n} > 0 \]

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

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

subject to the linear constraints
\[
\sum_{t=1}^{T_0} \sigma_{0t} w_{0t} = \sigma \equiv 1
\]  
(normality condition)

\[
\sum_{m=0}^{M} \sum_{t=1}^{T_m} \sigma_{mt} a_{mnt} w_{mt} = 0, \quad n = 1, \ldots, N
\]  
(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.

\[
w_{00} \equiv 1
\]

\[
\lim_{w_{mt} \to 0} \left( -\frac{c_{mt} w_m}{w_{mt}} \right) = 1
\]

The relationship between the primal and the dual variables are

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

and

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

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

For the case when

\[\sigma = 1\]

\[\sigma_m = 1, \quad m = 1, \ldots, M\]
\( c_{mt} = 1, \quad t = 1, \ldots, T_m \)

A strict inequality relationship exists between \( y_0 \) and \( V(\tilde{w}) \) as

\[ y_0(\tilde{x}) \geq V(\tilde{w}) \]

At the optimum solution, \( \tilde{x}^* \),

\[ y_0(\tilde{x}^*) = V(\tilde{w}^*) \]

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.
\[ V(\tilde{w}) = \sigma \left( \prod_{m=0}^{M} \prod_{t=1}^{T_m} c_{mt}^{\sigma_{mt}} w_{mt}^{\sigma_{mt}} \right) \]
\[ = \sigma \left( \prod_{m=0}^{M} \prod_{t=0}^{T_m} \frac{c_{mt}}{w_{mt}} w_{m0}^{\sigma_{m0}} \right) \]
\[ = \sigma \left( \prod_{m=0}^{M} \prod_{t=0}^{T_m} \frac{c_{mt}}{w_{mt}} \sum_{t=1}^{T_m} \sigma_{mt} w_{mt} \right) \]

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

The linear transformation is
\[ w_{m0} = \sigma_m \sum_{t=0}^{T_m} \sigma_{mt} w_{mt} \]

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

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

\[ x_1 x_2 x_3 \geq 100, \quad x_1 \geq 0 \]

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

\[ y_1 = x_1^2 x_4^{-2} + x_2^2 x_4^{-2} \leq 1 \]

\[ y_2 = \frac{100}{x_1 x_2 x_3} \leq 1 \]

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
\[ V = \begin{pmatrix} \frac{4}{w_{01}} & \frac{10}{w_{02}} & \frac{4}{w_{03}} & \frac{2}{w_{04}} & \frac{1}{w_{11}} \\ \frac{1}{w_{12}} & \frac{100}{w_{21}} & (w_{11} + w_{12}) & (w_{11})^2 & w_{21} \end{pmatrix} \]

subject to the constraints

\[
\begin{align*}
    w_{01} + w_{02} + w_{03} + w_{04} & = 1 \\
    w_{01} + 2w_{11} - w_{21} & = 0 \\
    w_{02} + 2w_{12} - w_{21} & = 0 \\
    w_{03} - w_{21} & = 0 \\
    w_{04} - 2w_{11} - 2w_{12} & = 0
\end{align*}
\]

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

\[
\ln V = w_{01} \ln \left( \frac{4}{w_{01}} \right) + w_{02} \ln \left( \frac{10}{w_{02}} \right) + w_{03} \ln \left( \frac{4}{w_{03}} \right) + w_{04} \ln \left( \frac{2}{w_{04}} \right) + w_{11} \ln \left( \frac{1}{w_{11}} \right) + w_{12} \ln \left( \frac{1}{w_{12}} \right) + w_{21} \ln \left( \frac{100}{w_{21}} \right) + A_1 \ln(A_1) + A_2 \ln(A_2)
\]

In addition to the previous linear constraints 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,
\[ X_1 = 5.085 \]
\[ X_2 = 2.684 \]
\[ X_3 = 7.332 \]
\[ X_4 = 5.741 = \frac{1}{(X_1^2 + X_2^2)^2} \]

Substituting these values into the primal objective function yielded

\[ y_0 = 87.990 \]

Since this is a posynomial problem

\[ 87.98708 < V(\vec{w}^*) = y_0(\vec{x}^*) < 87.990 \]

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 pseudo-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 solution. 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 routine 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 develop 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, \( \mathbf{x} \), is represented as

\[
\mathbf{x} = \begin{bmatrix}
  x_1 \\
  x_2 \\
  \vdots \\
  x_N
\end{bmatrix}
\]

The transpose of a vector, \( \mathbf{x} \), is \( \mathbf{x}^T \). The objective function is represented by

\( f_0(\mathbf{x}) \)

and a vector function of \( M \) constraints is given as

\( \mathbf{f}(\mathbf{x}) = 0 \)

where

\[
\mathbf{f}(\mathbf{x}) = \begin{bmatrix}
  f_1(\mathbf{x}) \\
  f_2(\mathbf{x}) \\
  \vdots \\
  f_M(\mathbf{x})
\end{bmatrix}
\]

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

\[
\frac{\partial f_i}{\partial \mathbf{x}} = \begin{bmatrix}
  \frac{\partial f_i}{\partial x_1} \\
  \frac{\partial f_i}{\partial x_2} \\
  \vdots \\
  \frac{\partial f_i}{\partial x_N}
\end{bmatrix}, \quad i = 0, 1, \ldots, M
\]

The "gradient of a vector function", is represented by
\[
\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \begin{bmatrix}
\begin{bmatrix}
\partial f_1 \\
\partial x_1 \\
\partial x_2 \\
\vdots \\
\partial x_M \\
\end{bmatrix} & \begin{bmatrix}
\partial f_1 \\
\partial x_1 \\
\partial x_2 \\
\vdots \\
\partial x_M \\
\end{bmatrix} & \cdots & \begin{bmatrix}
\partial f_1 \\
\partial x_1 \\
\partial x_2 \\
\vdots \\
\partial x_M \\
\end{bmatrix} \\
\end{bmatrix}
\]

This operation results in an \( M \times N \) matrix.

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

The notation
\[
\overline{a} \leq \overline{X} \leq \overline{b}
\]

implies
\[
a_i \leq X_i \leq b_i , \quad i = 1, \ldots, N
\]

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
\[
f_0(\mathbf{X})
\]
subject to the constraints
\[
\begin{align*}
\mathbf{f}(\mathbf{X}) &= 0 \\
\mathbf{a} &\leq \mathbf{X} \leq \mathbf{b}
\end{align*}
\]

Note that an inequality constraint
\[
c_i(\mathbf{X}) \leq 0
\]

can be redefined as
\[
f_i(\mathbf{X}) = c_i(\mathbf{X}) + s_i = 0, \quad s_i \geq 0
\]
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

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

and the set formed by the boundary conditions as

$$ P = \{ \bar{x} | a_j \leq X_j \leq b_j, \ j = 1, \ldots, N \} $$

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

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

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

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

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

\[ F(\vec{x}) = F(X_2, X_3) = \sqrt{1 - X_2^2 - X_3} - (1 - X_2^2 - X_3) + (X_2 - X_2^2) \]

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

\[ \frac{\partial F}{\partial \vec{x}} = \begin{bmatrix} \frac{\partial F}{\partial X_2} \\ \frac{\partial F}{\partial X_3} \end{bmatrix} \]

\[ = \begin{bmatrix} \frac{-2X_2}{\sqrt{1 - X_2^2 - X_3}} + 1, & \frac{1}{2} \\ \sqrt{1 - X_2^2 - X_3} & \sqrt{1 - X_2^2 - X_3} + 1 \end{bmatrix} \]

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

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

\[ \frac{\partial F}{\partial x_j^*} = 0 \quad \text{if} \quad a_j < x_j^* < b_j \]
\[ \frac{\partial F}{\partial x_j^*} < 0 \quad \text{if} \quad x_j^* = a_j \]
\[ \frac{\partial F}{\partial x_j^*} > 0 \quad \text{if} \quad x_j^* = b_j \]
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

\[
\frac{-1/2}{\sqrt{1 - x_2^2 - x_3}} + 1 = 0
\]

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

\[f_0(x) = f_0(x, y) = F(x)\]

The reduced gradient is

\[\frac{\partial F}{\partial \tilde{x}} = \frac{\partial f_0}{\partial \tilde{x}} + \frac{\partial f_0}{\partial y} \frac{\partial y}{\partial \tilde{x}}\]
\[
\frac{\partial F}{\partial x} = \begin{pmatrix}
\frac{\partial f_0}{\partial x_1} & \cdots & \frac{\partial f_0}{\partial x_j} & \cdots & \frac{\partial f_0}{\partial x_{N-M}} \\
\frac{\partial y_1}{\partial y_1} & \cdots & \frac{\partial y_1}{\partial y_j} & \cdots & \frac{\partial y_1}{\partial y_{M}} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\frac{\partial y_{M}}{\partial y_1} & \cdots & \frac{\partial y_{M}}{\partial y_j} & \cdots & \frac{\partial y_{M}}{\partial y_{M}} \\
\end{pmatrix}
\]

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

\[
\bar{f}(\bar{x}) = \bar{f}(\bar{x}, \bar{y}) = 0
\]

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

\[
\frac{\partial \bar{f}}{\partial \bar{x}} = -\left( \frac{\partial \bar{f}}{\partial \bar{y}} \right)^{-1} \begin{pmatrix}
\frac{\partial \bar{f}}{\partial \bar{x}} \\
\end{pmatrix}
\]

\[
= \begin{pmatrix}
\frac{\partial f_1}{\partial y_1} & \cdots & \frac{\partial f_1}{\partial y_{M}} \\
\frac{\partial y_1}{\partial y_1} & \cdots & \frac{\partial y_{M}}{\partial y_{M}} \\
\vdots & \ddots & \vdots \\
\frac{\partial y_{M}}{\partial y_1} & \cdots & \frac{\partial y_{M}}{\partial y_{M}} \\
\end{pmatrix}^{-1} \begin{pmatrix}
\frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_{N-M}} \\
\frac{\partial x_1}{\partial y_1} & \cdots & \frac{\partial x_{N-M}}{\partial y_{M}} \\
\vdots & \ddots & \vdots \\
\frac{\partial x_{N-M}}{\partial y_1} & \cdots & \frac{\partial x_{N-M}}{\partial y_{M}} \\
\end{pmatrix}
\]

The reduced gradient in the final form becomes,

\[
-T = \frac{\partial F}{\partial x} = \frac{\partial f_0}{\partial x} - \frac{\partial f_0}{\partial y} \left( \frac{\partial \bar{f}}{\partial \bar{y}} \right)^{-1} \begin{pmatrix}
\frac{\partial \bar{f}}{\partial \bar{y}} \\
\end{pmatrix}
\]

(1)
Example 2

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

\[
\begin{align*}
x_1 &= X_2 \\
x_2 &= X_3 \\
y_1 &= X_1
\end{align*}
\]

\[
\frac{\partial f_0}{\partial x} = \begin{bmatrix} \frac{\partial f_0}{\partial x_1} & \frac{\partial f_0}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial f_0}{\partial x_2} & \frac{\partial f_0}{\partial x_3} \end{bmatrix} = [1 - 2x_2, 0]
\]

\[
\frac{\partial f_0}{\partial y_1} = \begin{bmatrix} \frac{\partial f_0}{\partial y_1} \\ \frac{\partial f_0}{\partial x_1} \end{bmatrix} = [1 - 2x_1]
\]

\[
\frac{\partial f}{\partial y} = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} \\ \frac{\partial f_1}{\partial x_1} \end{bmatrix} = [2x_1]
\]

\[
\frac{\partial f}{\partial x} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \end{bmatrix} = [2x_2, 1]
\]

The reduced gradient is

\[
\frac{\partial F}{\partial x} = [1 - 2x_2, 0] - [1 - 2x_1] [2x_1]^{-1} [2x_2, 1]
\]

\[
= \begin{bmatrix} 1 - \frac{x_2}{X_1} \frac{1}{X_1} \\ 1 - \frac{2}{X_1} \frac{1}{X_1} \end{bmatrix}
\]

To put it in terms of Example 1 with \( X = \sqrt{1 - X_2^2 - X_3} \), the reduced gradient becomes
\[
\frac{\partial F}{\partial x} = \begin{cases} 
\frac{-X_2}{\sqrt{1 - X_2^2 - X_3}} + 1, & \frac{1}{2} \frac{1}{\sqrt{1 - X_2^2 - X_3}} + 1
\end{cases}
\]

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, \( \frac{\partial F}{\partial y} \). 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 determinant 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

\[
\bar{y} = \phi(\bar{x})
\]
and each component has continuous first order partial derivatives.

This theorem does not insure a functional relation between \( \vec{y} \) and \( \vec{x} \) if the Jacobian is singular. It also states the existence, but not the form, of the function, \( \vec{\Phi}(\vec{x}) \). 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

\[
L(\vec{X}, \vec{u}) = f_0(\vec{X}) - \vec{u}^T \vec{f}(\vec{X})
\]

The Kuhn-Tucker condition for optimality at \( \vec{X} \) is

\[
\frac{\partial L}{\partial x_j} = \begin{cases} 
< 0 & \text{if } x_j = a_j \\
0 & \text{if } a_j < x_j < b_j \\
> 0 & \text{if } x_j = b_j 
\end{cases}
\]

Therefore

\[
\frac{\partial L}{\partial x} = \frac{\partial f_0}{\partial x} - \vec{u}^T \frac{\partial \vec{f}}{\partial x} = \vec{V}^T
\]  \hspace{1cm} (2)

\[
\frac{\partial L}{\partial y} = \frac{\partial f_0}{\partial y} - \vec{u}^T \frac{\partial \vec{f}}{\partial y} = 0
\]  \hspace{1cm} (3)

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

\[
\vec{u}^T = \frac{\partial f_0}{\partial y} \left( \frac{\partial \vec{f}}{\partial y} \right)^{-1}
\]  \hspace{1cm} (4)
Substituting equation (4) into equation (2) yields

$$\frac{\partial L}{\partial \tilde{x}} = \frac{\partial f}{\partial \tilde{x}} - \frac{\partial f}{\partial \tilde{y}} \left( \frac{\partial \hat{f}}{\partial \tilde{y}} \right)^{-1} \left( \frac{\partial \hat{f}}{\partial \tilde{x}} \right) = \bar{v}^T$$

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-dimensional 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-degeneracy 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, \( \frac{\partial f}{\partial \bar{y}} \), 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 \( \bar{x}^0 = [\bar{x}^0, \bar{y}^0] \).

$$\bar{g}_{0}^T = \frac{\partial f}{\partial \bar{x}} = \frac{\partial f}{\partial \bar{x}} - \frac{\partial f}{\partial \bar{y}} \left( \frac{\partial \hat{f}}{\partial \bar{y}} \right)^{-1} \left( \frac{\partial \hat{f}}{\partial \bar{x}} \right)$$

(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, $\bar{x}$,

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

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

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 $x^0$ is represented by

$$x^-0 + \delta h^-0$$

Likewise, for $y^-0$, the step is

$$y^-0 + \delta k^-0$$

The vector, $k^-0$, determines the direction of movement for $y^-0$.

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

\[
\bar{k}^0 = -\left( \begin{bmatrix} \frac{\partial \bar{f}}{\partial \bar{y}^0} \\ \frac{\partial \bar{f}}{\partial \bar{x}^0} \end{bmatrix} \right)^{-1} \begin{bmatrix} \frac{\partial \bar{f}}{\partial \bar{y}^0} \\ \frac{\partial \bar{f}}{\partial \bar{x}^0} \end{bmatrix} \bar{h}^0
\]

(5)

**Step 2.2**

The problem is now to maximize

\[
f(\bar{x}^0 + \theta \bar{h}^0, \bar{y}^0 + \theta \bar{k}^0)
\]

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

**Step 3**

Calculate

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

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

\[
x^1_j = \begin{cases} 
\text{lower bound, if } x^0_j + \theta h^0_j \leq \text{lower bound} \\
\text{upper bound, if } x^0_j + \theta h^0_j \geq \text{upper bound} \\
x^0_j + \theta h^0_j, \text{ otherwise}
\end{cases}
\]

\( j = 1, \ldots, N-M \)

In general the calculated dependent variables, \( \bar{y}^1 \), do not satisfy the feasibility conditions. It is hoped, though, that the values are "close" to feasibility. Indeed, if \( \theta \) is small this condition exists. Usually the
vector, \( \bar{y}^1 \), 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 existence of

\[
\bar{y} = \phi(\bar{x}^1)
\]

is insured by the Implicit Function Theorem under the non-degeneracy conditions. If a component of \( \bar{y} \) violates a boundary condition, a degeneracy, 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 \( \bar{y}^1 \), then \( \bar{x}^1 \) is out of the functional domain. This is alleviated by reducing \( \theta \) (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 \( \theta \) and returning to step 3.

Letting \( \bar{x}^{-1} \) 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 \( \bar{y}^1 \). If the objective function is not improved,
decrease $\theta$ 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 $\theta$ 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, $x_{N+i}^N$, satisfy the condition

$$0 \leq x_{N+i}^N \leq \infty$$

If, at the starting point, $\bar{x}^0$, the $i^{th}$ 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, $x_{N+i}^N$.

$$f_i(\bar{x}^0) - x_{N+i}^N = 0$$

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

$$f_i(\bar{x}^0) + x_{N+i}^N = 0$$

There are two methods for obtaining a feasible solution, the penalty method and the two-phase method. For the penalty method let $M_i^N$ be a large number.

The objective function to be maximized becomes

$$f(\bar{x}) - \sum_{i \in \text{NF}} M_i^N x_{N+i}^N$$

$\text{NF} = \{i \mid f_i(\bar{x}) \neq 0\}$
Fig. 3.1. Computer flow diagram for the GRG algorithm.
(STEP 1.3) Compute the direction of movement $\vec{r}^*$, for $x^*$.
A simple example is $\vec{r}^* = \vec{p}^*$.

(STEP 2) Compute the direction of movement $\vec{y}^*$, for $y^*$.

(STEP 2.1) $\vec{r}^* = -\left( \frac{\partial f}{\partial y} \right)^{-1} \left[ \frac{\partial f}{\partial x} \right] \vec{h}^*$

(STEP 2.2) Use a one-dimensional search to $\max_{\theta} f_\theta(x^* + \theta \vec{r}^*, y^* + \theta \vec{y}^*)$

(STEP 3) Calculate $\vec{x}' = x^* + \theta \vec{r}^*$, $\vec{y}' = y^* + \theta \vec{y}^*$.
Project $\vec{x}'$ into $\mathbf{p}$.

\[ \forall j \ x'_j = \begin{cases} 
\text{Upper bound if } x'_j + \theta e_j > \text{Upper bound} \\
\text{Lower bound if } x'_j + \theta e_j < \text{Lower bound} \\
\ \ x'_j + \theta e_j \quad \text{Otherwise}
\end{cases} \]

3 \rightarrow 2

Fig. 3.1. (continued)
(STEP 4.1) SET $\theta = \frac{1}{2} \theta$

(SOLVE) \( \overline{f}(\overline{x}, \overline{y}) = 0 \)

DOES \( \overline{y} \)

EXIST?

(CHANGE THE BASIS TO OBTAIN A FEASIBLE SOLUTION)

(SOLVE) \( f_0(\overline{x}', \overline{y}') > f_0(\overline{x}, \overline{y}) \)

(NO)

(SOLVE) \( \frac{1}{2} \theta \)

SET \( \overline{x} = \overline{x}' \)

4

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

\[ - \sum_{i \in \text{NF}} X_i, \quad \text{NF} = \{ i \mid f_i(\bar{X}) \neq 0 \} \]

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

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
\[ f_1(\bar{x}) = x_1^2 + x_2^2 + x_3 - 1 = 0 \]

\[ x_i > 0, \quad i = 1, 2, 3 \]

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 \(\theta\) 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))

\[
\frac{\partial F}{\partial x} = \begin{pmatrix} x_1 (2-x_1) - x_1 (3-x_2) \\ x_2 \end{pmatrix} - \begin{pmatrix} x^3 \\ 2x_2 \end{pmatrix}
\]

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

\[
\bar{k} = \begin{pmatrix} -x_1 \\ -\frac{1}{2x_2} \end{pmatrix}
\]

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

\[ x_1 = x_1^0 + \theta d_1 \]
Table 3.1. Numerical results of Example 3.
\[ x_1 = x_1^0 + \theta d_1 \\
\quad x_2 = x_2^0 + \theta d_2 \\
\quad x_3 = x_3^0 + \theta d_3 \]

where \( d_1, d_2, \) and \( d_3 \) are the directions of move for \( x_1^0, x_2^0, \) and \( x_3^0. \)

\[
\frac{df}{d\theta} = \frac{\partial f}{\partial x_1} \frac{dx_1}{d\theta} + \frac{\partial f}{\partial x_2} \frac{dx_2}{d\theta} = 0
\]
or
\[
\frac{df}{d\theta} = (2 - x_1) d_1 + (3 - x_2) d_2 = 0
\]

\[
\frac{df}{d\theta} = (2 - x_1^0 - \theta d_1) d_1 + (3 - x_2^0 - \theta d_2) d_2 = 0
\]

This gives
\[
\theta = \frac{(2 - x_1^0) d_1 + (3 - x_2^0) d_2}{(d_1^2 + d_2^2)} \tag{8}
\]

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 \( \bar{x}^0 = [0.5, 0.5, 0.5]. \) Dividing this vector into independent and dependent variables gives

\[
\begin{align*}
\bar{x}^0 &= \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \\
\bar{y}^0 &= [x_2] = [0.5]
\end{align*}
\]
Fig. 3.2. Two-dimensional graph of the problem solving procedure of Example 3.
At this point the non-degeneracy of $\bar{y}^1$ is assumed.

**Step 1.1**

From equation (6)

$$\bar{g}^0 = \frac{\partial F}{\partial \bar{x}} = \begin{pmatrix} 0.5(2-0.5) - 0.5(3-0.5) \\ 0.5 \\ 2(0.5) \end{pmatrix} = \begin{pmatrix} -1.0 \\ -2.5 \end{pmatrix}.$$  

**Step 1.2**

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

**Step 1.3**

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

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

**Step 2**

**Step 2.1**

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

$$\bar{k}^0 = \begin{pmatrix} -1.0 \\ -2.5 \end{pmatrix}$$

**Step 2.2**

The optimizing value for $\theta$ 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 $\bar{x}^1$ and $\bar{y}^1$ are

$$\bar{x}^1 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix} + 0.547 \begin{pmatrix} -1.0 \\ -2.5 \end{pmatrix} = \begin{pmatrix} -0.047 \\ -0.867 \end{pmatrix}$$
Fig. 3.3. Optimum solution of Example 3.
\[ y^1 = [0.5] + 0.547 [3.5] = [2.41] \]

The independent variables must be projected into the boundary set, \( \overline{a} \leq \overline{x} \leq \overline{b} \). The projected independent vector is

\[ x^{-1} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \]

**Step 4**

The dependent variable is solved for via the constraint, \( f_1(\overline{x}) = 0 \).

\[ y^{-1} = \begin{bmatrix} x_2 \\ \sqrt{1 - x_1^2 - x_3} \end{bmatrix} \]

\[ y^{-1} = \begin{bmatrix} x_2 \\ 1.0 \end{bmatrix} \]

**Step 4.1**

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

**Step 4.2**

In this step, improvement of the objective is checked.

\[ f_0(\overline{x}_0) = 2.25 < 2.50 = f_0(\overline{x}_1) \]

**Step 5**

At this point, \( \overline{x}_1 \) becomes the starting point, \( \overline{x}_0 \), for the second iteration.

\[ x_0^{-1} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \]

**Iteration 2**

Since the slack variable, \( X_3 \), is zero, the next problem step is shown graphically in Fig. 3.2.
\[ x^0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad y^0 = [1] \]

**Step 1.1**

From equation (6),

\[
\frac{\partial F}{\partial x^0} = \begin{pmatrix} 1(2-0) - 0(3-1) \\ 1(1) \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}.
\]

**Step 1.2**

Since \( x^0_3 = 0 \), the lower bound, and \( g^0_2 \leq 0 \), the projected reduced gradient differs from \(-g^0\).

\[ p^0 = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \]

**Step 1.3**

\[ h = p^0 = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \]

**Step 2**

**Step 2.1**

From equation (7),

\[ k = [0, -1] \begin{pmatrix} 2 \\ 0 \end{pmatrix} = [0] \]

**Step 2.2**

From equation (8),

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

**Step 3**

\[ \bar{x}^1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + 1 \begin{pmatrix} 2 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \]
\[ y^1 = [1] + 1 [0] = [1] \]

\[ \tilde{x}^1 = \tilde{x}^1 \]

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

**Step 4**

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

**Step 4.1**

At this point, \( \theta \) 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**

\[ \theta = 0.5 \]

\[ \tilde{x}^1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0.5 \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]

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

\[ \tilde{x}^1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]

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

**Step 4**

\[ \tilde{y}^1 = [X_2] = \sqrt{1 - x_1^2 - x_3} \]

\[ y^1 = [X_2] = [0] \]
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 $\theta$ 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

$$\theta = 0.25$$

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

Refer to point C in Fig. 3.2.

Step 4

Solving for the dependent variable yields

$$y^{-1} = [0.816]$$

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

$$\bar{x}^0 = \begin{bmatrix} 0.5 \\ 0.816 \\ 0.816 \\ 0 \end{bmatrix}$$
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

\[
\begin{bmatrix}
\frac{2\sqrt{13}}{13} \\
\frac{3\sqrt{13}}{13} \\
0
\end{bmatrix}
= 
\begin{bmatrix}
0.554700 \\
0.832050 \\
0.0
\end{bmatrix}
\]

The third iteration yields

\[
\bar{\bar{x}}^1 = 
\begin{bmatrix}
0.554 \\
0.834 \\
0.0
\end{bmatrix}
\]

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

3.7 THE PSEUDO-NEWTON METHOD AND CHANGING THE BASIS

In general, numerical calculations are necessary for evaluating

\[
\bar{\bar{y}} = \nabla (\bar{\bar{x}})
\]

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

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

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

One of the most efficient procedures is the generalized Newton method.
The Taylor series expansion around the point $\bar{y}^1$ yields an approximate value for $\bar{f}(\bar{x}_1, \bar{y}_1)$ as

$$\bar{f}(\bar{x}_1, \bar{y}_1) \approx \bar{f}(\bar{x}^1, \bar{y}^1) + \frac{\partial \bar{f}}{\partial \bar{y}_1}(\bar{y}^1 - \bar{y}_1)$$

Since $\bar{f}(\bar{x}_1, \bar{y}_1) = 0$, the iterative formula becomes

$$\bar{y}_1 \approx \bar{y}_1^{i+1} = \bar{y}_1^i - \left(\frac{\partial \bar{f}}{\partial \bar{y}_1}\right)_i^{-1} \bar{f}(\bar{x}_1, \bar{y}_1^i)$$

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 $\bar{g}$ is reduced by $1/2$. This causes the independent vector, $\bar{x}_1$, and the starting point, $\bar{y}_1$, to be closer to the feasible point $[x^0, y^0]$.

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 degeneracy occurs with a component, $\bar{y}_1^t$, converging outside of $P$, the value of that component is set to its lower or upper bound. The degenerate, dependent component, $\bar{y}_1^t$, is then interchanged with an independent component, $\bar{x}_j^t$, 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 $\hat{y}_t$.

Improvement of the objective function is not insured when a change of basis occurs during an iteration. Improvement is insured for values of $\theta$ less than a maximum value $\theta^*$ 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 $\theta$ 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 degeneracy 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

\[
\begin{align*}
x_1^2 + x_2^2 & \leq 4 \\
x_1^2 + x_3^2 & \leq 1 \\
x_1 & \geq 0 \quad i = 1, 2, 3
\end{align*}
\]

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

\[
\begin{align*}
x_1^2 + x_2^2 + x_4^2 & = 4 \\
x_1^2 + x_3^2 + x_5^2 & = 1 \\
x_1 & \geq 0, \; i = 1, 2, 3, 4, 5.
\end{align*}
\]

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

\[
\min \{ |\alpha_{i,j}|(b_j - x_j), |\alpha_{i,j}|(x_j - a_j) \}, \text{ for } \forall j
\]
\[ a_{ij} = \left( \begin{array}{c} \frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial x} \end{array} \right) \]

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, \( x^0 \), 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.

\[ NV \quad = \quad \text{the number of variables in the original problem.} \]

\[ NIN \quad = \quad \text{the number of inequality constraints and, therefore, the number of slack variables added.} \]

\[ NIN1 \quad = \quad \text{the number of inequality constraints not satisfied at the starting point and, therefore, the number of artificial variables added for this reason.} \]
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 chosen 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

\[ f_i(\bar{X}) \leq 0, \quad i = 1, \ldots, NC \]

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

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

\[ f_0(\bar{X}) = \sum_{j=1}^{5} \ln (1 - \exp(B_j \cdot X_j)) \]

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), JVA(100) GEGA 30
COMMON A, ALFA, X, XC, XI, XS, Y, C, VC, IBAS, IHB, IVC, JVA, IVB GEGA 40
COMMON NV, NC, NK, NEG, NIN, NTV, NV1, NEV, NEWL, NT0, NIT, IN1, NIN2, NIN3, NIGLGA 50
1N4, NVNIN1, NVNIN2, NVNIN3, INDEX, II, IR, IR1, IS, IS1, IT, IBP, ICDB, JCDB, KCGGGA 60
2DB, KFIL, KLIN, KRED, KD, FILL, PHI, PSI, PS13, TB, TD, TC, EPSIL, EPSIL0, EPSIL2GGA 70
COMMON KFUNC, KGRAD, KCONT, KINV1, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KCGGGA 80
1RENI, KREN2, KINV, KCDBA1, KREN1, KREN21 GEGA 90
COMMON IUIREC, DEI, FTA, JKO, LC, YSORT GEGA 100

Fig. 4.1. GREG common block definition statements.
\[ f_1(\bar{x}) = \sum_{j=1}^{5} C_{1j} \cdot x_j^2 \leq 110 \]

\[ f_2(\bar{x}) = \sum_{j=1}^{5} C_{2j} \cdot (x_j + \exp(x_j/4)) \leq 175 \]

\[ f_3(\bar{x}) = \sum_{j=1}^{5} C_{3j} \cdot x_j \exp(x_j/4) \leq 200 \]

**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 artificial 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 (≤ 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,
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 con-
straints 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 initialize 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, \( f_i(\vec{x}) \), i = 1, ..., NC,
in the same order as sequenced in CPHI. The partial derivatives are repre-
sented in terms of the FORTRAN variable XC(j), j = 1, ..., NV.

Constant values may also be initialized in JACOB. It is the third sub-
routine 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 initiali-
zation purposes. Fig. 4.5 shows an example of GRADFI.
SUBROUTINE CPHI

DIMENSION A(0,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150) GEGA 20
1Y1(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IWA(100) GEGA 30
COMM A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IWA,IVB GEGA 40
COMMON NV,NC,NK,NEG,NIN,NTV,NVI,NEV,NEVL,NG1,NIN1,NIN2,NIN3,NG2 GEGA 50
1H,4,NV1,NV12,NV13,NV14,NV15,NV16,NV17,NV18,NV19,NV20,KIN1,KIN2,KIN3,KIN4,KG1 GEGA 60
2CH,KFIL,KLIN,KREN,KO,KFL,PHI,PSI,PS13,TB,TD,TC,EPSIL,EPSIL0,EPSIL2,EG1 GEGA 70
COMMON KFCNC,KGRAD,KCONT,KIN1,KINV1,KIN2,KCDBA,KJACO,KMAX1,KMAX2,KEGA GEGA 80
IREN1,KREN2,KINV,KCDBA1,KREN11,KREN241 GEGA 93
COMMON IDIREC,DELF1,ETA,JKO,LC,YSORT GEGA 100

CC1(1,1)=1.0
IF(IT)100,101,101
CC2(1,1)=7.0
CC3(1,1)=7.0
CC4(1,1)=2.0
CC5(1,1)=2.0
CC6(1,1)=2.0
CC7(1,1)=3.0
CC8(1,1)=3.0
CC9(1,1)=5.0
CC10(1,1)=8.0
CC11(1,1)=8.0
CC12(1,1)=8.0
CC13(1,1)=8.0
CC14(1,1)=8.0
CC15(1,1)=8.0
CC16(1,1)=8.0
CC17(1,1)=8.0
CC18(1,1)=8.0
CC19(1,1)=8.0
CC20(1,1)=8.0
CC21(1,1)=8.0
CC22(1,1)=8.0
CC23(1,1)=8.0
CC24(1,1)=8.0
CC25(1,1)=8.0
CC26(1,1)=8.0
CC27(1,1)=8.0
CC28(1,1)=8.0
CC29(1,1)=8.0
CC30(1,1)=8.0
CC31(1,1)=8.0
CC32(1,1)=8.0
CC33(1,1)=8.0
CC34(1,1)=8.0
CC35(1,1)=8.0
CC36(1,1)=8.0
CC37(1,1)=8.0

100 VC1(1)=0.0
101 VC1(1)=VC1(1)+CC1(1,1)*XC1(1,1)**2
VC1(1)=VC1(1)-110.0
VC2(1)=0.0
VC3(1)=0.0
VC4(1)=0.0
VC5(1)=0.0
VC6(1)=0.0
VC7(1)=0.0
VC8(1)=0.0
VC9(1)=0.0
VC10(1)=0.0
VC11(1)=0.0
VC12(1)=0.0
VC13(1)=0.0
VC14(1)=0.0
VC15(1)=0.0
VC16(1)=0.0
VC17(1)=0.0
VC18(1)=0.0
VC19(1)=0.0
VC20(1)=0.0
VC21(1)=0.0
VC22(1)=0.0
VC23(1)=0.0
VC24(1)=0.0
VC25(1)=0.0
VC26(1)=0.0
VC27(1)=0.0
VC28(1)=0.0
VC29(1)=0.0
VC30(1)=0.0
VC31(1)=0.0
VC32(1)=0.0
VC33(1)=0.0
VC34(1)=0.0
VC35(1)=0.0
VC36(1)=0.0
VC37(1)=0.0

Fig. 4.3. An example of CPHI.
Fig. 4.4. An example of JACOB.
SUBROUTINE GRADFI

DIMENSION A(50,100), ALFA(50,50), X(15C), XG(150), XI(150), XS(150)

1, Y(150), C(150), VC(50), IBAS(50), IHB(100), IVC(50), IVA(100)

COMMON A, ALFA, X, XG, XI, XS, Y, C, VC, IBAS, IHB, IVC, IVA, IVB

COMMON NV, NR, NK, NEG, NIN1, NTV, NV1, NEV, NEVL, NTC, NIN1, NIN2, NIN3, NICEGA

COMMON 1N4, NINV1, NINV2, NINV3, INDEX, 1I, IR, IR1, IS, IS1, IT, IBD, JCB, JCD, KCDB, KCGE

COMMON KCB, KFIL, KLIN, KREN, KD, F11, PM, PSI, PS13, TB, TC, EPSIL, EPSL0, EPSIL2, KGE

COMMON KFPNC, KGRAC, KCONT, KINV1, KINV2, KCDBA, KJAC, KMAX1, KMAX2, KGE

COMMON IKREC, DELTFI, ETA, JK0, LC, YSORT

COMMON B(5), CC(3,5)

CO101 J=1,5

CO09 AA=EXP(B(J)*XG(J))

101 C(J)=B(J)*AA/(AA-1.*C)

CO11 RETURN

CO12 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 $1.0 \times 10^{-7}$ can yield erroneous results.
Table 4.1. Parameters

<table>
<thead>
<tr>
<th>FORTRAN Program</th>
<th>Symbols</th>
<th>Explanations</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. NV</td>
<td></td>
<td>the number of original problem variables.</td>
<td>0</td>
</tr>
<tr>
<td>2. NIN</td>
<td></td>
<td>the number of inequality constraints.</td>
<td>0</td>
</tr>
<tr>
<td>3. NEG</td>
<td></td>
<td>the number of equality constraints.</td>
<td>0</td>
</tr>
<tr>
<td>4. NEVL</td>
<td></td>
<td>the maximum number of iterations for Newton's Method.</td>
<td>20*</td>
</tr>
<tr>
<td>5. NTØ (NT zero)</td>
<td></td>
<td>the maximum number of bisections in the parabolic interpolation process when maximizing a function of a single variable.</td>
<td>6*</td>
</tr>
<tr>
<td>6. ITET</td>
<td></td>
<td>the number of previous iterations used to help determine a maximum value for θ.</td>
<td>20*</td>
</tr>
<tr>
<td>7. ICONJ</td>
<td></td>
<td>equals 1 if conjugate directions are desired when maximizing, otherwise zero.</td>
<td>1</td>
</tr>
<tr>
<td>8. IDIAG</td>
<td></td>
<td>equals 1 if diagonal directions are desired when maximizing, otherwise zero.</td>
<td>0</td>
</tr>
<tr>
<td>9. ITMAX</td>
<td></td>
<td>the maximum number of iterations.</td>
<td>50</td>
</tr>
<tr>
<td>10. KFIL</td>
<td></td>
<td>equals 1 if the cost function is linear, otherwise, zero.</td>
<td>0</td>
</tr>
<tr>
<td>11. KLIN</td>
<td></td>
<td>equals 1 if all of the constraints are linear, otherwise, zero.</td>
<td>0</td>
</tr>
<tr>
<td>12. NCØ (NC zero)</td>
<td></td>
<td>equals zero for linear programming problems and is &gt; the number of constraints for nonlinear programming problems.</td>
<td>10</td>
</tr>
<tr>
<td>13. ITSOR</td>
<td></td>
<td>the iteration which recording of the intermediate output starts. (1 ≤ ITSOR ≤ ITMAX)</td>
<td>1*</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td><strong>Table 4.1. (continued)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14. ISOLSR</td>
<td>the solution is printed out every ISOLSR iterations. For small problems, ISOLSR = ITMAX. For large problems, ISOLSR &lt; ITMAX.</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>15. EPSIL</td>
<td>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})</td>
<td>0.1E0*</td>
<td></td>
</tr>
<tr>
<td>16. EPSILØ (EPSIL zero)</td>
<td>is used as a stopping criteria for Newton's Method. (EPSILØ \geq 10^{-7})</td>
<td>0.1E-02</td>
<td></td>
</tr>
<tr>
<td>17. EPSIL1</td>
<td>is used as a stopping criteria if the problem is declared convex. (EPSIL1 \geq 10^{-7})</td>
<td>0.1E-02</td>
<td></td>
</tr>
<tr>
<td>18. EPSIL2</td>
<td>is used as a stopping criterion by using the norm of the reduced gradient. (EPSIL2 \geq 10^{-7})</td>
<td>0.1E-02</td>
<td></td>
</tr>
<tr>
<td>19. FC</td>
<td>equals zero if the problem is non-convex; equals one if the problem is convex. This parameter affects only the stopping criteria.</td>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>

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

123456789

first card &PARM NV=10,NIN=3,NEG=2,PC=1.0,
second card EPSILO=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 GRADPI. 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.

/*! 
//LKD.SYSMOD DD DSN=yyyyyyyyyyyyyyyyyyyyyyyyy(name),DISP=(NEW,KEEP),UNIT=2314
// VOL=SER=nnnnnnn,SPACE=(TRK,(10,5,3),RLSE)
//GO.SYSIN DD *

/*

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=yyyyy.xxxxxxxxx(name),DISP=(NEW,KEEP),UNIT=2314
// VOL=SER=nnnnn,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.
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.xxxxxxxxx,DISP=SHR,UNIT=2314,VOL=SER=nnnnnnn
//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.
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.
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 \( \varepsilon_2 \) (EPSIL2) times the norm of the projected reduced gradient of the first iteration that has no artificial variables.
2. Let

\[ n = \max \; n_i, \forall i \]

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

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

\[ b_i = \text{the upper bound for an independent variable } x_i. \]

\[ a_i = \text{the lower bound for an independent variable } x_i. \]

The program is terminated when

\[ n < \varepsilon_3, \varepsilon_3 = \frac{\varepsilon_1}{n}, \varepsilon_1 = \text{EPSILL}. \]

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

\[ \Delta \phi < \varepsilon_1, \varepsilon_1 = \text{EPSILL} \]

\[ \Delta \phi = \sum n_i, \forall i \]

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

If \( \bar{V} \) is a conjugate direction the iteration is repeated and \( \bar{V} \)

is set equal to \( \bar{c} \). If the condition occurs again an inversion followed by a search for a new basis is performed. The program is terminated if \( \bar{c} \cdot \bar{V} < 0 \) 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 \( \theta^* \) such that

\[ \theta^* = \max \left( \theta/10, \theta^{\text{int}} \right) \]

\( \theta^{\text{int}} \) is an interpolated value between (zero and \( \theta \)).

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

$$|f_o(x^o + \theta h^o, y^o + \theta k^o) - f_o(x^o, y^o) | < \varepsilon^* \quad |f_o(x^o, y^o) |$$

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

7. The program is terminated when

$$|f_o(x^o + \theta h^o, y^o + \theta k^o) - f_o(x^o, y^o) | < \varepsilon^* \quad |f_o(x^o, y^o) |$$

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 \( \theta \)
due to failure to re-enter the feasible region or improve the objective
function.

8. The program terminates when

$$|f_o(x^o + \theta h^o, y^o + \theta k^o) - f_o(x^o, y^o) | < \varepsilon^* \quad |f_o(x^o, y^o) |$$

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 $\varepsilon^* \mid f(\bar{x}^0, \bar{y}^0)$ 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 precision).
PARAMETRES  (parameters)

M  5
NIN  6
NEG  0
NEVL  20
NTD  6
ITEI  20
ICONJ  1
ICTAG  0
ITMAX  50
KFIL  0
KLIN  0
NCO  10
ITSRR  1
ISOILRS  10
EPSIL  0.1E 00
EPSILO  0.1E-04
EPSIL1  0.1E-02
EPSIL2  0.1E-02
PC  0.1

Table 4.2. Translation of the five output sections of the GREG program.
GRADIENT REDUIT GENERALISE

**Nombre d'variables naturelles** 5  
**Nombre total de variables** 11  
**Nombre de contraintes** 6

**Epsilon de Newton** 0.100E-04 (stopping condition for Newton's method)
**Epsilon test gradient** 0.100E-02 (stopping condition for gradient test)

**Fonction économique** 0.117187500000000E-01  
(objective function)

<table>
<thead>
<tr>
<th>Variable naturelle (starting point)</th>
<th>Bonne supérieure (upper bound)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x(1) ) 0.7861999512188E 02</td>
<td>( x(1) ) 0.1000000000000E 03</td>
</tr>
<tr>
<td>( x(2) ) 0.3343998718261E 02</td>
<td>( x(2) ) 0.4500000000000E 02</td>
</tr>
<tr>
<td>( x(3) ) 0.3106999216540E 02</td>
<td>( x(3) ) 0.4500000000000E 02</td>
</tr>
<tr>
<td>( x(4) ) 0.4409999084472E 02</td>
<td>( x(4) ) 0.4500000000000E 02</td>
</tr>
<tr>
<td>( x(5) ) 0.3521998596191E 02</td>
<td>( x(5) ) 0.4500000000000E 02</td>
</tr>
</tbody>
</table>

**Valeur des contraintes**  
(constraint values)

<table>
<thead>
<tr>
<th>Contrainte</th>
<th>Valeur</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C(1) )</td>
<td>-0.2113647640927E 00</td>
</tr>
<tr>
<td>( C(2) )</td>
<td>-0.9178435253906E 02</td>
</tr>
<tr>
<td>( C(3) )</td>
<td>-0.1105728149414E 02</td>
</tr>
<tr>
<td>( C(4) )</td>
<td>-0.8692718509859E 01</td>
</tr>
<tr>
<td>( C(5) )</td>
<td>-0.4872711181640E 01</td>
</tr>
<tr>
<td>( C(6) )</td>
<td>-0.1272888183593E 00</td>
</tr>
</tbody>
</table>

Table 4.2. (continued)
<table>
<thead>
<tr>
<th>IT</th>
<th>PHI</th>
<th>DIR. GRAD.</th>
<th>NO</th>
<th>YN</th>
<th>DELTA F</th>
<th>ETA</th>
<th>NCDH</th>
<th>NCN</th>
<th>NITN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1181757612506E 03</td>
<td>NO 2</td>
<td>0.35E 03</td>
<td>0.19E 04</td>
<td>NCDH 0</td>
<td>NCN 2</td>
<td>NITN 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.1182347500000E 03</td>
<td>LC 3</td>
<td>0.95E 03</td>
<td>0.57E 03</td>
<td>ETA 0.70E 03</td>
<td>NCDH 2</td>
<td>NCN 8</td>
<td>NITN 20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.118234986294000E 03</td>
<td>LC 3</td>
<td>0.43E 03</td>
<td>0.38E 02</td>
<td>ETA 0.38E 02</td>
<td>NCDH 0</td>
<td>NCN 2</td>
<td>NITN 7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.118073723119000E 03</td>
<td>NO 2</td>
<td>0.83E 03</td>
<td>0.36E 02</td>
<td>ETA 0.36E 02</td>
<td>NCDH 0</td>
<td>NCN 3</td>
<td>NITN 12</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.118073723119000E 03</td>
<td>NO 4</td>
<td>0.81E 03</td>
<td>0.43E-02</td>
<td>ETA 0.43E-02</td>
<td>NCDH 0</td>
<td>NCN 3</td>
<td>NITN 12</td>
<td></td>
</tr>
</tbody>
</table>

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.
DELTA F = Δφ 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)
Table 4.2. (continued)
<table>
<thead>
<tr>
<th>NOMBRE DE SORTIE</th>
<th>7 (stopping condition)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOMBRE D'ITERATIONS</td>
<td>6 (number of iterations)</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P CALCULANT LES CONTRAINTES</td>
<td>93 (number of times the constraints were calculated)</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P CALCULANT LA FONCTION ECONOMIQUE</td>
<td>69 (number of times the objective function was calculated)</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P CALCULANT LE GRADIENT DE LA FONCTION ECONOMIQUE</td>
<td>14 (number of times the gradient was calculated)</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P CALCULANT L'INVERSE AVEC BASE DONNÉE</td>
<td>4 (number of inverse calculations with the given base)</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P CALCULANT L'INVERSE AVEC RECHERCHE DE BASE</td>
<td>1 (number of inverse calculations with another base)</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P EFFECTUANT LES CHANGEMENTS DE BASE</td>
<td>3 (number of base changes)</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P CALCULANT LE JACOBIEN DES CONTRAINTES</td>
<td>8 (number of times the Jacobian of the constraints was calculated)</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P EFFECTUANT UNE RECHERCHE DE MAX SANS DICHOTOMIE</td>
<td>14 (number of maximums found without bisection)</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P EFFECTUANT UNE RECHERCHE DE MAX AVEC DICHOTOMIE</td>
<td>0 (number of maximums found by bisection)</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P EFFECTUANT LA RENTREE DANS LE DOMAINE</td>
<td>26 (number of reentries into the feasible region)</td>
</tr>
</tbody>
</table>

| 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

\[ f_0(x) = (2x_1 - \frac{1}{2}x_2^2) + (3x_2 - \frac{1}{2}x_2^2) \]
subject to the constraint

\[ f_1(\bar{x}) = \bar{x}_1^2 + \bar{x}_2^2 - 1 \leq 0 \]

\[ x_i \geq 0, \quad i = 1, 2, 3 \]

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

\[ Q = \frac{R_3}{(1-R_1)(1-R_4)}^2 \]
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, \( R_i \), \( i = 1, 2, 3, 4 \).

\[ \text{weight} = \frac{4}{\prod_{i=1}^{4} R_i^{\alpha_i}} \]

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

\[ K_{11} R_{11}^{\alpha_1} + K_{22} R_{22}^{\alpha_2} + K_{33} R_{33}^{\alpha_3} + K_{44} R_{44}^{\alpha_4} \leq C \]

and minimize

\[ \text{weight} = K_{11} R_{11}^{\alpha_1} + K_{22} R_{22}^{\alpha_2} + K_{33} R_{33}^{\alpha_3} + K_{44} R_{44}^{\alpha_4} \]

subject to the constraints

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

\[ R_i > R_1, \min, i = 1, 2, 3, 4 \]

Numerical values for the parameters were given as
\[ K_1 = 100, \ K_2 = 100, \ K_3 = 200, \ K_4 = 150 \]
\[ C = 800, \ R_{s,\text{min}} = 0.9 \]
\[ a_i = 0.6, \ R_{i,\text{min}} = 0.5, \ i = 1,2,3,4 \]

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

\[
R_s = \prod_{i=1}^{N} \left(1 - (1 - R_i)^{M_i}\right)
\]

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 $c_i M_i^i$ represents the cost of the units at stage $i$ and $c_i \exp(M_i^{i/4})$ is the additional cost for interconnecting parallel units.

The weight constraint was given as

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

The value $w_i M_i^i$ is the weight of the units at stage $i$ and the factor $\exp(M_i^{i/4})$ 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

- $R_1 = 0.80$, $p^1 = 1.0$, $c^1 = 7.0$, $w^1 = 7.0$
- $R_2 = 0.85$, $p^2 = 2.0$, $c^2 = 7.0$, $w^2 = 8.0$
- $R_3 = 0.90$, $p^3 = 3.0$, $c^3 = 5.0$, $w^3 = 8.0$
- $R_4 = 0.65$, $p^4 = 4.0$, $c^4 = 9.0$, $w^4 = 6.0$
- $R_5 = 0.75$, $p^5 = 2.0$, $c^5 = 4.0$, $w^5 = 9.0$

The problem is (changing $M_i^i$ to $X_i$)

maximize

$$f_0 = \ln(1 - 0.2 X_1) + \ln(1 - 0.15 X_2) + \ln(1 - 0.10 X_3) + \ln(1 - 0.35 X_4) + \ln(1 - 0.25 X_5)$$

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

\[
f_0(\bar{X}) = \sum_{i=1}^{10} b_i x_{i+5} - \frac{5}{2} \sum_{i=1}^{5} \sum_{j=1}^{5} c_{ij} x_i x_j - 2 \sum_{j=1}^{5} d_j x_j^3
\]

subject to the quadratic constraints

\[
\sum_{j=1}^{10} a_{ij} x_{j+5} - e_i - 2 \sum_{j=1}^{5} c_{ij} x_j - 3 d_1 x_1^2 \leq 0
\]

\[i = 1, 2, 3\]

\[
\sum_{j=1}^{10} a_{ij} x_{j+5} - e_i - 2 \sum_{j=1}^{5} c_{ij} x_j - 3 d_1 x_1^2 + x_{i+12} = 0
\]

\[i = 4, 5\]

\[0 \leq x_k \leq 100, \quad k = 1, \ldots, 17\]

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

\[
f_0 = -32.34867
\]

\[
x_1 = 0.2994560
\]

\[
x_2 = 0.3332918
\]

\[
x_3 = 0.4006066
\]
<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>d_{i1}</td>
<td>4</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>e_{i1}</td>
<td>-15</td>
<td>-27</td>
<td>-36</td>
<td>-18</td>
<td>-12</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>b_{11}</td>
<td>-40</td>
<td>-2</td>
<td>-1/4</td>
<td>-4</td>
<td>-4</td>
<td>-1</td>
<td>-40</td>
<td>-60</td>
<td>5</td>
</tr>
</tbody>
</table>

Value of $c_{ij}$:

<table>
<thead>
<tr>
<th>i</th>
<th>j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30</td>
<td>-20</td>
<td>-10</td>
<td>32</td>
<td>-10</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>-20</td>
<td>39</td>
<td>-6</td>
<td>-31</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>-10</td>
<td>-6</td>
<td>10</td>
<td>-6</td>
<td>-10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>-31</td>
<td>-6</td>
<td>39</td>
<td>-20</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>-10</td>
<td>32</td>
<td>-10</td>
<td>-20</td>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1 Parameter values for the modified Colville test problem two.
<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>-16</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>0.4</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>-3.5</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>-2</td>
<td>0</td>
<td>-4</td>
<td>-1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>-9</td>
<td>-2</td>
<td>1</td>
<td>-2.8</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>0</td>
<td>-4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>13</td>
<td>-1</td>
<td>-2</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.1 (continued)
$X_4 = 0.4287636$

$X_5 = 0.2243463$

$X_6 = X_7 = 0.0$

$X_8 = 0.5175596$

$X_9 = 0.0$

$X_{10} = 0.3081506$

$X_{11} = 0.1193264$

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

$X_{14} = 0.1027665$

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

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

$$f_0(x) = 5.3578547 X_3^2 + 0.835681 X_1 X_5 + 37.293239 X_1 + 40792.141$$

subject to

$$85.334407 + 0.0056858 X_2 X_5 + 0.0006262 X_1 X_4 - 0.0022053 X_3 X_5 \geq 0$$

$$32.51249 + 0.0071317 X_2 X_5 + 0.0029955 X_1 X_2 + 0.0021813 X_3^2 \geq 90$$

$$9.300961 + 0.0047026 X_2 X_5 + 0.0012547 X_1 X_3 + 0.0019085 X_3 X_4 \geq 20$$

$$78 \leq X_1 \leq 102$$

$$33 \leq X_2 \leq 45$$

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

\[ |f_0| = 87.98774 \]
\[ X_1 = 5.087201 \]
\[ X_2 = 2.681470 \]
\[ X_3 = 7.33074 \]

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 originally 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 natural 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.

\[
\frac{dT-E}{dt} = -K(T-E)
\]

\(T\) = water temperature in °F

\(E\) = equilibrium temperature in °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 °F

\(T_2\) = water temperature at a downstream point in °F

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

The condition \(T-E > 0\) 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.

\[
\frac{dL}{dt} = -K_1L
\]

\(L\) = BOD concentration in mg/l
$K_1 = \text{deoxygenation rate coefficient in days}^{-1}$

$t = \text{time in days}$

This differential equation has the simple solution

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

where

$L_1 = \text{initial BOD concentration in mg/l at time} = t_1$

$L_2 = \text{BOD concentration in mg/l at time} = t_2$

$t_2 - t_1 = \text{flow time in days}$

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

$D = \text{DO deficit in mg/l}$

$K_2 = \text{reeration rate coefficient in days}^{-1}$

$t = \text{time in days}$

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

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

\[+ D_1 \exp(-K_2 (t_2-t_1))\]

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 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

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

\[-D_1 \exp(-K_2 (t_2-t_1))\]

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].

\[ DO_s = 14.652 - 0.41022 \, TC + 0.0079910 \, TC^2 - 0.00007774 \, TC^3 \]

where

\[ DO_s = \text{DO saturation level in mg/l} \]
\[ TC = \text{temperature in } ^\circ\text{C} \]

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

Likewise, the coefficients \( K_1 \) and \( K_2 \) are related to temperature by the relationships [19,11]

\[ K_1 = K_{1,20} \times (1.047)^{(TC-20)} \]
\[ K_2 = K_{2,20} \times (1.024)^{(TC-20)} \]

where

\[ K_{1,20} = \text{deoxygenation rate coefficient at 20 } ^\circ\text{C in days}^{-1} \]
\[ K_{2,20} = \text{reoxygenation rate coefficient at 20 } ^\circ\text{C in days}^{-1} \]
\[ TC = \text{temperature in } ^\circ\text{C} \]

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.

\[ X_T_n = \text{thermal input in BTU/hr at the beginning of stage n} \]
\[ X_B_n = \text{organic, BOD, input in lbs/day at the beginning of stage n} \]
\[ Q_n = \text{uniform flow within stage n in ft}^3/\text{sec} \]
\[ T_F_n = \text{time of flow from the beginning to the end of stage n, in days} (t_2-t_1) \]
Fig. 6.1. An example of the discrete staging of the river basin model [17].
It is assumed that $X_T^n$, $X_B^n$, $Q^n$, and $T_F^n$ 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

$$\text{thermal input in } ^\circ F = \frac{C_1 \cdot X_T^n}{Q^n}$$

where

$C_1$ is a conversion constant

$$C_1 = 4.44970 \times 10^{-6} \left( \frac{\text{hr} \cdot \text{ft}^3}{\text{sec} \cdot \text{BTU}} \right)$$

The conversion of BTU to the rise in water temperature in $^\circ F$ and of water volume to mass is relative to the water temperature at 39.1 $^\circ F$. 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.

$$\text{BOD input in mg/l} = \frac{C_2 \cdot X_B^n}{Q^n}$$
\[ C_2 = 0.185404 \left( \frac{\text{days} \cdot \text{ft}^3 \cdot \text{mg}}{\text{sec} \cdot \text{lbs} \cdot \text{ft}} \right) \]

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 \( n \).

\[
t_{\text{rise}_n} = \left[ \frac{C_1 \cdot X_T}{Q_n} \right] \left[ \frac{100 - d(T,n)}{100} \right]
\]

The expression represents the rise in stream temperature after a cooling treatment of \( d(T,n) \) per cent.

\[
d(T,n) = \% \text{ of thermal input cooled to the equilibrium water temperature in stage } n.
\]

The maximum stream temperature occurs at the thermal pollution source and is represented as

\[
t_{\text{temp}_n} = ST_n + \left[ \frac{C_1 \cdot X_T}{Q_n} \right] \left[ \frac{100 - d(T,n)}{100} \right]
\]

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.

\[
t_{\text{rise}_n} \leq TRS_n
\]

\[
t_{\text{temp}_n} \leq TMAX_n
\]

\[
\min DO_n \geq MDO_n
\]

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**

\[
S_{T_n+1} = E + (T_n - E) \exp (-K \cdot T_{F_n})
\]

\[
T_n = \text{temp}_n = S_{T_n} + \left[ \frac{C_1 \cdot X_{T_n}}{Q_n} \right] \left[ \frac{100 - d(T, n)}{100} \right]
\]

where

\[
S_{T_n} = \text{incoming temperature for stage n, in } ^\circ\text{F}
\]

\[
S_{T_n+1} = \text{outgoing temperature for stage n and the incoming temperature for stage n+1, in } ^\circ\text{F}
\]

**BOD**

\[
S_{B_n+1} = L_n \exp (-K \cdot T_{F_n})
\]

\[
L_n = S_{B_n} + \left[ \frac{C_2 \cdot X_{B_n}}{Q_n} \right] \left[ \frac{100 - d(B, n)}{100} \right]
\]

where \(d(B, n)\) = % treatment of BOD at the beginning of stage n

\[
S_{B_n} = \text{incoming BOD for stage n, in mg/\ell}
\]

\[
S_{B_n+1} = \text{outgoing BOD for stage n and the incoming BOD for stage n+1, in mg/\ell}
\]

**DO**

\[
S_{O_n+1} = D_O - \frac{K \cdot L_n}{K_2 - K_1} \left[ \exp(-K \cdot T_{F_n}) - \exp(-K_2 \cdot T_{F_n}) \right] - D_1 \exp(-K_2 \cdot T_{F_n})
\]

\[
D_1 = \text{initial DO saturation level - } S_{O_n}
\]

\[
= \text{initial DO deficit for stage n}
\]

\[
S_{O_n} = \text{incoming DO for stage n in mg/\ell}
\]
\[ S_{n+1}^0 = \text{outgoing DO for stage n and the incoming DO for stage n+1}. \]

For all cases, \( n = 1, \ldots, N-1 \).

In summary, the input of pollution into a stage is represented by the constant values \( X_T^n \) and/or \( X_B^n \). 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 \( S_0 \) 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

\[
\begin{align*}
\text{t-rise}_n & \leq \text{TRS}_n \\
\text{temp}_n & \leq \text{TMAX}_n \\
\min \text{DO}_n & \geq \text{MDO}_n
\end{align*}
\]

\( n = 1, \ldots, N \)

where \( C_n \) is the treatment cost as a 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.
\[ X_i = d(T, i), \quad X_{N+i} = d(B, i) \]

\[ i = 1, \ldots, N, \quad N = \text{number of stages} \]

The total number of decision variables is \(2 \times N\). The temperature rise constraint is then given as

\[ f_i(\bar{X}) = \left[ \frac{C_i \cdot XT_i}{100 Q_i} \right] \left[ 100 - X_i \right] - \text{TRS}_i \leq 0 \quad i = 1, \ldots, N \]

The first \(N\) constraints consist of the temperature rise constraint for each stage. The maximum temperature constraint is formulated as

\[ f_{N+i}(\bar{X}) = \text{ST}_i + \left[ \frac{C_i \cdot XT_i}{100 Q_i} \right] \left[ 100 - X_i \right] - \text{TMAX}_i \leq 0 \quad i = 1, \ldots, N \]

This set of \(N\) constraints contains the maximum temperature constraints for each stage. The final set of constraints is for the minimum DO restriction.

\[ f_{2N+i}(\bar{X}) = - \min \left[ \text{DO} \right] + \text{MDO}_i \leq 0 \quad i = 1, \ldots, N \]

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 \(\text{ST}_i\), \(\text{SB}_i\), and \(\text{SO}_i\) are calculated. Only \(\text{ST}_1\), \(\text{SB}_1\), and \(\text{SO}_1\) remain constant. The values of the other state variables depend on the values of the previous decision variables. \(\text{ST}_i\), \(\text{SB}_i\), and \(\text{SO}_i\) 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 \(t = T_F\) for stage \(1\).
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 derivatives are determined as

\[
\frac{\partial f_i}{\partial X_i} = - \left[ \frac{C_1 \cdot X T_1}{100 Q_1} \right]
\]

\[
\frac{\partial f_i}{\partial X_j} = 0, \; j = 1, \ldots, N \text{ and } j \neq i
\]

\[
\frac{\partial f}{\partial X_{N+1}} = 0, \; j = 1, \ldots, N
\]

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.

\[
\frac{\partial f_{N+1}}{\partial X_i} = - \left[ \frac{C_1 \cdot X T_1}{100 Q_1} \right]
\]

\[
\frac{\partial f_{N+1}}{\partial X_j} = 0, \; j > i
\]

\[
\frac{\partial f}{\partial X_{N+1}} = \frac{\partial S T_i}{\partial X_j}, \; j < i
\]
\[ \frac{\partial f_{N+1}}{\partial X_{N+j}} = 0, \ j = 1, \ldots, N \]

The partial derivatives of the state variable, \( ST_j \), are calculated inductively.

\[ ST_i = XE_{i-1} + (T_i - XE_{i-1}) \exp (-K_{i-1} TF_{i-1}) \]

\[ T_i = ST_{i-1} + \frac{C_i \cdot XT_{i-1}}{100 Q_{i-1}} \left[ 100 - X_i \right] \]

\[ K_{i-1} = (K) \text{ the temperature dissipation rate coefficient specified for a given stage, } i - 1. \text{ This subscripting has not been used in the previous discussion, but this, and all other constants, are generally different in each stage.} \]

Thus,

\[ \frac{\partial ST_i}{\partial X_{i-1}} = - \left[ \frac{C_i \cdot XT_{i-1}}{100 Q_{i-1}} \right] \exp (-K_{i-1} TF_{i-1}) \]

\[ \frac{\partial ST_i}{\partial X_{i-k}} = - \left[ \frac{C_i \cdot XT_{i-k}}{100 Q_{i-k}} \right] \exp \left( - \sum_{m=1}^{k} K_{i-m} TF_{i-m} \right), \ 0 < k < i \]

By letting \( j = i - k \),

\[ \frac{\partial f_{N+1}}{\partial X_{j}} = - \left[ \frac{C_j \cdot XT_j}{100 Q_j} \right] \exp \left( \sum_{\ell=j}^{i-1} K_{\ell} TF_{\ell} \right), \ j < i \]

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. \[ \text{FMIN1} = \min [DO] \text{ when } X_j + X_j + \Delta \]

2. \[ \frac{\partial f_{2N+1}}{\partial X_j} = \left[ \frac{\text{FMIN1} - \text{FMIN}}{\Delta} \right] \]

The minus sign comes from the minus sign in the constraint function

\[ f_{2N+1} = - \min [DO_i] + \text{MDO}_i \leq 0 \]

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 contributors 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].
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, discontinuities resulting in
new stages occurred, 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 occurring
in September during the previous ten years was used as the basis for a conserva-
tive 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
<table>
<thead>
<tr>
<th>n</th>
<th>Location</th>
<th>$X_{T_n}$</th>
<th>$X_{B_n}$</th>
<th>$Q_n$</th>
<th>$X_{F_n}$</th>
<th>$K$</th>
<th>$K_{1,20}$</th>
<th>$K_{2,20}$</th>
<th>$T_{F_n}$</th>
<th>distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Clayton to McDonough and Atkinson</td>
<td>0</td>
<td>1.1$\times 10^5$</td>
<td>820</td>
<td>80.7</td>
<td>1.24</td>
<td>1.00</td>
<td>0.90</td>
<td>0.05</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>McDonough and Atkinson to Cedar Creek</td>
<td>2.4$\times 10^9$</td>
<td>0</td>
<td>870</td>
<td>80.7</td>
<td>1.29</td>
<td>1.00</td>
<td>0.90</td>
<td>1.87</td>
<td>38</td>
</tr>
<tr>
<td>3</td>
<td>Cedar Creek to Yates</td>
<td>0</td>
<td>0</td>
<td>885</td>
<td>80.7</td>
<td>1.10</td>
<td>1.00</td>
<td>0.90</td>
<td>0.17</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Yates to Yellowjacket Creek</td>
<td>1.6$\times 10^9$</td>
<td>0</td>
<td>920</td>
<td>80.7</td>
<td>1.34</td>
<td>1.00</td>
<td>0.90</td>
<td>2.40</td>
<td>44</td>
</tr>
</tbody>
</table>

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.

\[ X = \% \text{ treatment} \]

\[
\text{cost} = 0.04X, \quad 0 \leq X < 20 \\
\text{cost} = 0.03X + 0.2, \quad 20 \leq X < 30 \\
\text{cost} = 0.01X + 0.8, \quad 30 \leq X < 40 \\
\text{cost} = \left( \frac{0.01}{3} \right) X - \left( \frac{0.40}{3} - 1.2 \right), \quad 40 \leq X < 70 \\
\text{cost} = 0.015X + 0.25, \quad 70 \leq X < 90 \\
\text{cost} = 0.06X - 3.8, \quad 90 \leq X < 95 \\
\text{cost} = 0.98X - 91.2, \quad 95 \leq X < 100
\]

The cost for the treatment of the thermal waste was given as (in $)

\[ X = \% \text{ treatment} \]

Cool Cost = \( \frac{817,000}{0.9} \) \[ 1 - \exp \left( -0.023 \times X \right) \]
(McDonough and Atkinson)

Cool Cost = \( \frac{575,000}{0.9} \) \[ 1 - \exp \left( -0.023 \times X \right) \]
(Yates)

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 \times 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 simultaneous differential equations to define the constraints. This makes it conceivable 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 desirable. 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


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.


* - 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
C. JUNE 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 AL Ow & 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 215
C. THERE NOW OCCURS ONE DATA CARD FOR EACH VARIABLE CONTAINING
C. START, INTER, ALow, & UPPER, FORMAT F10.0, I10, 2F10.0
C.
C. 900 FORMAT(5X,5HEPROR/5X,3HAIN44,4,15) NFUN=THE NUMBER OF VARIABLES
C. 901 FORMAT(5X,5HEPROR/5X,3HINVALID VALUE FOR NFUN OR NROWS)
C. 902 FORMAT(5X,5HEPROR/5X,27HINTER EQUALS 0 FOR VARIABLE, 19)
C. 1000 FORMAT(215)
C. 2000 FORMAT(4X,1HP,4X,5X,3HROW,13,4X,F12.5,3X,3HRUW,13,4X,F12.5)
C. 3000 FORMAT(4X,1HP,4X,5X,4HGRID,13,3X,F12.5)
C. 3001 FORMAT(4X,1HP,4X,5X,3HROW,13,4X,F12.5,3X,4HGRID,13,3X,F12.5)
C. 4000 FORMAT(4X,3HSET,13,4X,8HMARKER*,17X,8HSEPORD)
C. 5000 FORMAT(F10.0, I10, 2F10.0)
C. 6000 FORMAT(5X,23HFOCHO CHECK FOR VARIABLE,15/5X,6HSTART=,F10.2,
C. 15X,6HINTER=,I10 /5X,6HLOWL=,F10.2/5X,6HUPPER=,F10.2/7)
C. 7000 FORMAT(4X,6HENDSET,4X,8HMARKER*,17X,8HSEPEND/80(1H*))
C. 8000 FORMAT(1X,14HSEPBOUND P,14,9X,3H1.0)
C. 9000 FORMAT(5X,27HFOCHO CHECK FOR NFUN & NROWS/
C. 15X,6HNFUN =,I5/5X,6HNFUCS=,I5/1)
C.
C. UNIT NUMBERS
C.
C. KREAD=5
C. NPRINT=6
C. NPUNCH=7
C.
C. READ=CARD INPUT, NPRINT=PRINT OUTPUT, NPUNCH=PUNCH OUTPUT
C.
C. DIMENSION ROW(501,F150,50)
C. READ(NREAD,1000) NFUN,NROWS
C. WRITE(NPRINT,5000) NFUN,NROWS
C. IF(NFUN.LT.1 OR NROWS.LT.1) GO TO 96
C. NKK=0
C. NKK=1000
C.
C. CO CALCULATIONS FOR EACH VARIABLE
C.
C. 00400 KX=1,NFUN
C. IAX=1
C. READ(NREAD,50CD1) START, INTER, ALow, UPPER
WRITE(NPRINT,6000) KK,START,INTER,ALOW,UPPER

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 DETERMINE UNIFORM PARTITION STEP-SIZE
AN=INTER
CELT= (UPPER-ALOW)/AN
LL=KK+100
WRITE(NPRINT,4000) LL
AN=NROWS+KK

C CHECK ALOW .NE. START. CREATE ANOTHER PARTITION IF TRUE
IJ=0
IF(ALOW-START)2,3,2
2 X=START
AX=ALOW
IJ=1
INTER=INTER+1
IAK=2
DO10IK=1,NROWS
10 ROW(IK)=0.0
GO TO 8

C CALCULATE PARTITION VALUES
3 IJ=IJ+1
DO10IK=1,NROWS
10 ROW(IK)=0.0
XX=IJ-IAK
XX=XX+CELT+ALOW
AX=XX*DELTA

C CALCULATE FUNCTION DIFFERENCES
8 DO100J=1,NROWS
CALL FUNC(J,KK,AX,F,NFUN,NROWS)
ROW(J)=F(J,KK)
CALL FUNC(J,KK,XX,F,NFUN,NROWS)
100 ROW(J)=ROW(J)-F(J,KK)

C CREATE GRID VALUE
GRID=X-AX
II=IJ+KKK

C START PUNCH LEO
IR=0
16 IR=IR+1
IF(IR.GT.NROWS) GU TO 90
IF(IR(ROW(IR)))17,20,17
17 IRR=IR+1
GO TO 20
19 IRRR=IRR+1
20 IF(IRRR.GT.NRRWS) GO TO 89
   IF(ROW(IRR) .EQ. 19, 75)
89 IRRR=IRR+100
   WRITE(NPNCX,3CO1) II, IRRR, ROW(II), LL, GRID
   IR=NN
   GO TO 200
90 IR=NN
   WRITE(NPNCX,3CO0) II, LL, GRID
   GO TO 200
75 IRR=IR+100
   IMM=IRR+100
   WRITE(NPNCX,2CO0) II, IRRR, ROW(II), IMM, ROW(IRR)
199 IR=IRR
200 IF(IIR.LT.NN) GO TO 16
   IF(IJ.LT.INTER) GO TO 3
   KKK=KKK+INTER
   AKK=AKK+INTER
400 CONTINUE
   WRITE(NPNCX,7CO0)
C
C******CREATE BOUNDS SECTION
C
   COLI=1, MKK
   II=I+1000
   WRITE(NPNCX,8CO0) II
   STOP
96 WRITE(NPRT,9C1)
   STOP
97 WRITE(NPRT,9C2) KK
   STOP
98 WRITE(NPRT,9C01) 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 FUNCJ, KK, X, F, NFUN, NRCWS
DIMENSION F(56, 50)
NM=(KK-1)*NRCWS+J
GO TO(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20), KM
1 F(1, 1)=ALOG(1.0-2.0**X)
RETURN
2 F(2, 1)=-X**2
RETURN
3 F(3, 1)=7.0*(X*EXP(X/4.0))
RETURN
4 F(4, 1)=7.0*X*EXP(X/4.0)
RETURN
5 F(1, 2)=ALOG(1.0-15.0**X)
RETURN
6 F(2, 2)=2.0*X**2
RETURN
7 F(3, 2)=7.0*(X*EXP(X/4.0))
RETURN
8 F(4, 2)=8.0*X*EXP(X/4.0)
RETURN
9 F(1, 3)=ALOG(1.0-1.0**X)
RETURN
10 F(2, 3)=3.0*X**2
RETURN
11 F(3, 3)=5.0*(X*EXP(X/4.0))
RETURN
12 F(4, 3)=8.0*X*EXP(X/4.0)
RETURN
13 F(1, 4)=ALOG(1.0-35.0**X)
RETURN
14 F(2, 4)=4.0*X**2
RETURN
15 F(3, 4)=9.0*(X*EXP(X/4.0))
RETURN
16 F(4, 4)=6.0*X*EXP(X/4.0)
RETURN
17 F(1, 5)=ALOG(1.0-25.0**X)
RETURN
18 F(2, 5)=2.0*X**2
RETURN
19 F(3, 5)=4.0*(X*EXP(X/4.0))
RETURN
20 F(4, 5)=9.0*X*EXP(X/4.0)
RETURN
END
CONTROL PROGRAM COMPILER - MCO/360 V2-MA

C001  PROGRAM
C002  INITIAL
C004  MOVE(XUATA,'DATA-SET')
C005  MOVE(XPBNAME,'SEPAREABLE')
C006  CONVERT
C007  HCOUT
C008  MOVE(XOBJ,'ROW101')
C009  MOVE(XRHS,'LIMITS')
C010  TITLE('PRIMAL-LOCAL OPTIMUM')
C011  XSYTLR=-1  $# BOUNDED VARIABLES AT LOWER LIMIT
C012  SETUP('BOUND', 'SEPBOUND', 'MAX')
C013  PRIMAL
C014  SOLUTION  $# LOCAL OPTIMUM
C015  TITLE('PRIMAL-GLOBAL OPTIMUM')
C016  XSYTLR=+1  $# BOUNDED VARIABLES AT UPPER LIMIT
C017  SETUP('BOUND', 'SEPBOUND', 'MAX')
C018  PRIMAL
C019  SOLUTION  $# GLOBAL OPTIMUM
C020  TITLE('DUAL-GLOBAL OPTIMUM')
C021  SETUP('BOUND', 'SEPBOUND', 'MAX')
C022  DUAL
C023  SOLUTION  $# GLOBAL OPTIMUM-DUAL
C024  EXIT
C025  PEND
**PRIMAL-DUAL OPTIMUM**

**SOLUTION (OPTIMAL)**

**TIME = 0.371 MINS. ITERATION NUMBER = 34**

---

**FUNCTION**

- PRI1
- PRI2
- PRI3
- PRI4

**ACTIVITY**

- 1.12776
- 38.55374
- 79.69700
- 151.77100

**DEFINED AS**

- LIMITS
- SEBOUND

---

**PRIMAL-DUAL OPTIMUM**

**SECTION 1 - PRIMAL**

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>AT ACTIVITY</th>
<th>SLACK ACTIVITY</th>
<th>LOWER LIMIT</th>
<th>UPPER LIMIT</th>
<th>DUAL ACTIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PRI1</td>
<td>1.12776</td>
<td>-</td>
<td>NONE</td>
<td>1.00000</td>
</tr>
<tr>
<td>2</td>
<td>PRI2</td>
<td>38.55374</td>
<td>13.94436</td>
<td>NONE</td>
<td>9.00000</td>
</tr>
<tr>
<td>3</td>
<td>PRI3</td>
<td>79.69700</td>
<td>22.29840</td>
<td>NONE</td>
<td>101.49200</td>
</tr>
<tr>
<td>4</td>
<td>PRI4</td>
<td>151.77100</td>
<td>-</td>
<td>NONE</td>
<td>151.21600</td>
</tr>
<tr>
<td>5</td>
<td>GR1</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>GR2</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>GR3</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>GR4</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>GR5</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
</tbody>
</table>
### PRIMAL-GLOBAL OPTIMUM

**SOLUTION (OPTIMAL)**

**TIME = 0.41 MINS. ITERATION NUMBER = 38**

- **FUNCTIONAL**
  - 1.12476
- **RESTRICTIONS**
  - LIMIT
- **BCL 105**

### PRIMAL-GLOBAL OPTIMUM

**SECTION 1 - ROWS**

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>AT ACTIVITY</th>
<th>SLACK ACTIVITY</th>
<th>LOWER LIMIT</th>
<th>UPPER LIMIT</th>
<th>DUAL ACTIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PC101C1 BS</td>
<td>1.12476</td>
<td>NONE</td>
<td>NONE</td>
<td>1.00000</td>
</tr>
<tr>
<td>2</td>
<td>PC101C2 RS</td>
<td>84.95514</td>
<td>11.94426</td>
<td>NONE</td>
<td>9.000000</td>
</tr>
<tr>
<td>3</td>
<td>PC101C3 BS</td>
<td>79.68340</td>
<td>22.29440</td>
<td>NONE</td>
<td>101.94260</td>
</tr>
<tr>
<td>4</td>
<td>PC101C4 UL</td>
<td>151.71870</td>
<td>NONE</td>
<td>151.21800</td>
<td>0.00048</td>
</tr>
<tr>
<td>5</td>
<td>PC101C1 EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>PC101C2 EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>PC101C3 EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>PC101C4 EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PC101C5 EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td></td>
</tr>
<tr>
<td>NUMBER</td>
<td>COLUMN</td>
<td>ACTIVITY</td>
<td>INPUT COST</td>
<td>LOWER LIMIT</td>
<td>UPPER LIMIT</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>----------</td>
<td>------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>BS</td>
<td>24700000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>12</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>13</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>15</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>16</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>17</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>18</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>19</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>20</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>21</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>22</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>24</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>26</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>27</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>28</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>29</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>30</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>31</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>32</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>33</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>34</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>35</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>36</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>37</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>38</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>39</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>40</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>41</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>42</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>43</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>44</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>45</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>46</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>47</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>48</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>49</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>50</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>51</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>52</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>53</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>54</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>55</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>56</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>57</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>58</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>59</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>60</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>61</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>62</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>63</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>64</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>65</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>66</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>67</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>68</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>69</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>70</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>71</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>72</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>73</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>74</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>75</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>76</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>77</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>78</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>79</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>80</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>81</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>82</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>83</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>84</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>85</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>86</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>87</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>88</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>89</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>90</td>
<td>HS</td>
<td>24320000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NUMBER</td>
<td>COLUMN</td>
<td>AT</td>
<td>ACTIVITY</td>
<td>INPUT COST</td>
<td>LOWER LIMIT</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>------</td>
<td>----------</td>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>59</td>
<td>PIC45</td>
<td>LL</td>
<td>1.000FU</td>
<td>.29247</td>
<td>.</td>
</tr>
<tr>
<td>60</td>
<td>PIC46</td>
<td>LL</td>
<td>1.000FU</td>
<td>.00417</td>
<td>.</td>
</tr>
<tr>
<td>61</td>
<td>PIC47</td>
<td>LL</td>
<td>1.000FU</td>
<td>.00361</td>
<td>.</td>
</tr>
<tr>
<td>62</td>
<td>PIC48</td>
<td>LL</td>
<td>1.000FU</td>
<td>.00314</td>
<td>.</td>
</tr>
<tr>
<td>63</td>
<td>PIC49</td>
<td>LL</td>
<td></td>
<td>.00272</td>
<td>.</td>
</tr>
<tr>
<td>64</td>
<td>PIC50</td>
<td>LL</td>
<td></td>
<td>.00216</td>
<td>.</td>
</tr>
<tr>
<td>65</td>
<td>PIC51</td>
<td>LL</td>
<td></td>
<td>.00245</td>
<td>.</td>
</tr>
<tr>
<td>66</td>
<td>PIC52</td>
<td>LL</td>
<td></td>
<td>.00376</td>
<td>.</td>
</tr>
<tr>
<td>67</td>
<td>PIC53</td>
<td>LL</td>
<td></td>
<td>.00195</td>
<td>.</td>
</tr>
<tr>
<td>68</td>
<td>PIC54</td>
<td>LL</td>
<td></td>
<td>.00184</td>
<td>.</td>
</tr>
<tr>
<td>69</td>
<td>PIC55</td>
<td>LL</td>
<td></td>
<td>.00117</td>
<td>.</td>
</tr>
</tbody>
</table>
### DUAL-GLOBAL OPTIMUM

**SOLUTION: OPTIMAL**

**TIME**: 31

<table>
<thead>
<tr>
<th><strong>EN</strong></th>
<th><strong>NAME</strong></th>
<th><strong>DEFINITION</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FUNCTIONAL</strong></td>
<td>1.12976</td>
<td>RO-101</td>
</tr>
<tr>
<td><strong>RESTRICT</strong></td>
<td>LIMIT</td>
<td>SATELLITE</td>
</tr>
</tbody>
</table>

### DUAL-GLOBAL OPTIMUM

**SECTION 1 - ROWS**

<table>
<thead>
<tr>
<th>NUMBER</th>
<th><strong>AT</strong></th>
<th><strong>ACTIVITY</strong></th>
<th><strong>SLACK ACTIVITY</strong></th>
<th><strong>LOWER LIMIT</strong></th>
<th><strong>UPPER LIMIT</strong></th>
<th><strong>DUAL ACTIVITY</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>18</td>
<td>1.12976</td>
<td>1.12976</td>
<td>NONE</td>
<td>NONE</td>
<td>1.00000</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>84.19214</td>
<td>13.94445</td>
<td>NONE</td>
<td>98.62800</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>21</td>
<td>72.49970</td>
<td>22.92594</td>
<td>NONE</td>
<td>101.99300</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>151.21140</td>
<td>22.92594</td>
<td>NONE</td>
<td>151.21800</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>21</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>8</td>
<td>21</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>9</td>
<td>21</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
<td>1.00000</td>
<td>0.00000</td>
</tr>
<tr>
<td>NUMBER</td>
<td>COLUMN</td>
<td>AT</td>
<td>ACTIVITY</td>
<td>INPUT COST</td>
<td>LOWER LIMIT</td>
<td>UPPER LIMIT</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>----</td>
<td>----------</td>
<td>------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>10</td>
<td>X1</td>
<td>BS</td>
<td></td>
<td>2.76GCFO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>X2</td>
<td>BS</td>
<td></td>
<td>2.32GCFO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>X3</td>
<td>BS</td>
<td></td>
<td>2.11GCFO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>X4</td>
<td>BS</td>
<td></td>
<td>3.56GCFO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>X5</td>
<td>BS</td>
<td></td>
<td>3.83GCFO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>P12C1</td>
<td>UL</td>
<td></td>
<td>1.21GCFO</td>
<td>1.8232</td>
<td>1.60000</td>
</tr>
<tr>
<td>16</td>
<td>P12C2</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00417</td>
<td>1.60000</td>
</tr>
<tr>
<td>17</td>
<td>P13C3</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00523</td>
<td>1.00000</td>
</tr>
<tr>
<td>18</td>
<td>P13C4</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00443</td>
<td>1.00000</td>
</tr>
<tr>
<td>19</td>
<td>P13C5</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00275</td>
<td>1.00000</td>
</tr>
<tr>
<td>20</td>
<td>P13C6</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00319</td>
<td>1.00000</td>
</tr>
<tr>
<td>21</td>
<td>P13C7</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00270</td>
<td>1.00000</td>
</tr>
<tr>
<td>22</td>
<td>P13C8</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00300</td>
<td>1.00000</td>
</tr>
<tr>
<td>23</td>
<td>P14C9</td>
<td>LL</td>
<td></td>
<td>0.00195</td>
<td>1.00000</td>
<td>0.00034</td>
</tr>
<tr>
<td>24</td>
<td>P15C10</td>
<td>LL</td>
<td></td>
<td>0.00166</td>
<td>1.00000</td>
<td>0.00073</td>
</tr>
<tr>
<td>25</td>
<td>P15C11</td>
<td>LL</td>
<td></td>
<td>0.00141</td>
<td>1.00000</td>
<td>0.00107</td>
</tr>
<tr>
<td>26</td>
<td>P15C12</td>
<td>LL</td>
<td></td>
<td>1.00GCFO</td>
<td>1.1976</td>
<td>1.00000</td>
</tr>
<tr>
<td>27</td>
<td>P15C13</td>
<td>LL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00397</td>
<td>1.00000</td>
</tr>
<tr>
<td>28</td>
<td>P15C14</td>
<td>LL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00337</td>
<td>1.00000</td>
</tr>
<tr>
<td>29</td>
<td>P15C15</td>
<td>LL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00270</td>
<td>1.00000</td>
</tr>
<tr>
<td>30</td>
<td>P15C16</td>
<td>BS</td>
<td></td>
<td>2.92GCFO</td>
<td>0.00273</td>
<td>1.00000</td>
</tr>
<tr>
<td>31</td>
<td>P15C17</td>
<td>LL</td>
<td></td>
<td>0.00184</td>
<td>1.00000</td>
<td>0.00045</td>
</tr>
<tr>
<td>32</td>
<td>P15C18</td>
<td>LL</td>
<td></td>
<td>0.00152</td>
<td>1.00000</td>
<td>0.00090</td>
</tr>
<tr>
<td>33</td>
<td>P15C19</td>
<td>LL</td>
<td></td>
<td>0.00135</td>
<td>1.00000</td>
<td>0.00037</td>
</tr>
<tr>
<td>34</td>
<td>P15C20</td>
<td>LL</td>
<td></td>
<td>0.0034</td>
<td>1.00000</td>
<td>0.00136</td>
</tr>
<tr>
<td>35</td>
<td>P16C21</td>
<td>LL</td>
<td></td>
<td>0.00086</td>
<td>1.00000</td>
<td>0.00086</td>
</tr>
<tr>
<td>36</td>
<td>P16C22</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00219</td>
<td>1.00000</td>
</tr>
<tr>
<td>37</td>
<td>P16C23</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.07143</td>
<td>1.00000</td>
</tr>
<tr>
<td>38</td>
<td>P16C24</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00269</td>
<td>1.00000</td>
</tr>
<tr>
<td>39</td>
<td>P16C25</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00299</td>
<td>1.00000</td>
</tr>
<tr>
<td>40</td>
<td>P16C26</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00418</td>
<td>1.00000</td>
</tr>
<tr>
<td>41</td>
<td>P16C27</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00331</td>
<td>1.00000</td>
</tr>
<tr>
<td>42</td>
<td>P16C28</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00262</td>
<td>1.00000</td>
</tr>
<tr>
<td>43</td>
<td>P16C29</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00298</td>
<td>1.00000</td>
</tr>
<tr>
<td>44</td>
<td>P16C30</td>
<td>LL</td>
<td></td>
<td>0.00165</td>
<td>1.00000</td>
<td>0.00060</td>
</tr>
<tr>
<td>45</td>
<td>P16C31</td>
<td>LL</td>
<td></td>
<td>0.00131</td>
<td>1.00000</td>
<td>0.00039</td>
</tr>
<tr>
<td>46</td>
<td>P16C32</td>
<td>LL</td>
<td></td>
<td>0.00164</td>
<td>1.00000</td>
<td>0.00019</td>
</tr>
<tr>
<td>47</td>
<td>P16C33</td>
<td>LL</td>
<td></td>
<td>0.00282</td>
<td>1.00000</td>
<td>0.00153</td>
</tr>
<tr>
<td>48</td>
<td>P16C34</td>
<td>LL</td>
<td></td>
<td>1.30GCFO</td>
<td>0.30696</td>
<td>1.00000</td>
</tr>
<tr>
<td>49</td>
<td>P16C35</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00469</td>
<td>1.00000</td>
</tr>
<tr>
<td>50</td>
<td>P16C36</td>
<td>UL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00399</td>
<td>1.00000</td>
</tr>
<tr>
<td>51</td>
<td>P16C37</td>
<td>LL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00598</td>
<td>1.00000</td>
</tr>
<tr>
<td>52</td>
<td>P16C38</td>
<td>LL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00572</td>
<td>1.00000</td>
</tr>
<tr>
<td>53</td>
<td>P16C39</td>
<td>LL</td>
<td></td>
<td>1.00GCFO</td>
<td>0.00288</td>
<td>1.00000</td>
</tr>
<tr>
<td>54</td>
<td>P16C40</td>
<td>LL</td>
<td></td>
<td>0.00250</td>
<td>1.00000</td>
<td>0.00039</td>
</tr>
<tr>
<td>55</td>
<td>P16C41</td>
<td>LL</td>
<td></td>
<td>0.00346</td>
<td>1.00000</td>
<td>0.00069</td>
</tr>
<tr>
<td>56</td>
<td>P16C42</td>
<td>LL</td>
<td></td>
<td>0.00197</td>
<td>1.00000</td>
<td>0.00018</td>
</tr>
<tr>
<td>57</td>
<td>P16C43</td>
<td>LL</td>
<td></td>
<td>0.00188</td>
<td>1.00000</td>
<td>0.00013</td>
</tr>
<tr>
<td>58</td>
<td>P16C44</td>
<td>LL</td>
<td></td>
<td>0.00149</td>
<td>1.00000</td>
<td>0.00014</td>
</tr>
<tr>
<td>NUMBER</td>
<td>COLUMN</td>
<td>AT ACTIVITY</td>
<td>INPUT COST</td>
<td>LOWER LIMIT</td>
<td>UPPER LIMIT</td>
<td>REDUCE COST</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-------------</td>
<td>------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>59</td>
<td>PIC44</td>
<td>LL</td>
<td>1.006GR0</td>
<td>.26594</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>PIC46</td>
<td>UL</td>
<td>1.006GR0</td>
<td>.00417</td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>PIC47</td>
<td>LL</td>
<td>1.006GR0</td>
<td>.00352</td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>PIC48</td>
<td>LL</td>
<td>1.006GR0</td>
<td>.00314</td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>PIC49</td>
<td>LL</td>
<td></td>
<td>.00272</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>PIC50</td>
<td>LL</td>
<td></td>
<td>.00316</td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>PIC51</td>
<td>LL</td>
<td></td>
<td>.00205</td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>PIC52</td>
<td>LL</td>
<td></td>
<td>.00178</td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>PIC53</td>
<td>LL</td>
<td></td>
<td>.00355</td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>PIC54</td>
<td>LL</td>
<td></td>
<td>.00135</td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>PIC55</td>
<td>LL</td>
<td></td>
<td>.00117</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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.

ALLOW can never equal zero (because xn(0) does not exist) and UPPER < 1.

The data for the variables are ordered as

\[ w_{01}, w_{02}, \ldots, w_{04}, w_{11}, w_{12}, w_{21}, A_1, A_2. \]

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(15)
1001 FORMAT(20A4)
2000 FORMAT(4X,1HP,14,5X,6HOBJECT,4X,F12.8,3X,4HMKIC,13,3X,F12.5)
4000 FORMAT(4X,3HSFST,13,4X,8H'MARKER',17X,6H'SEPHSG')
5000 FORMAT(F10.0,110,2F10.0)
6000 FORMAT(5X,23H'ECHO CHECK FOR VARIABLE,15/5X,6HSTART=',F10.5/15X,6HINTER=',110/5X,6HALOK=',F10.5/5X,6HUPPER=',F10.5/7000 FORMAT(6HBOUND')S)
7001 FORMAT(6HBOUND')S)
8000 FORMAT(1X,14H'SEPBOUND P',14,9X,3H1,0)
9000 FORMAT(5X,35H'THE NUMBER OF SEPARATED VARIABLES=',15/15X,35H'THE NUMBER OF DEPENDENT VARIABLES=',15/)
   AREAD=5
   NPRINT=6
   NDISK=9
   DIMENSION A(20)
   DOUBLE PRECISION OBJ,F
   COMMON K,START,NFC
   CATA SEPA,ENDA/4HSEPA,4HENDA/
   REWIND NDISK
20 READ(NREAD,10C1) A
   IF(A(1)=EQ.SEPA) GO TO 21
   WRITE(NDISK,10C1) A
   GO TO 20
21 READ(NREAD,1CC0) NFUN,NCON
   WRITE(NPRINT,9C00) NFUN,NCON
   IF(NCON.GT.NFUN) STOP9999
   NFC=NFUN-NCON
   KKK=0
   KKK=1000
   CD400KKK=1,NFUN
   IAX=1
   READ(NREAD,6CC0) START,INTER,ALOW,UPPER
   WRITE(NPRINT,6C00) KK,START,INTER,ALOW,UPPER
   IF(INTER.GT.5C) INTER=5C
   IF(ALOW.EQ.UPPER) STOP8888
   IF(INTER.EQ.0) STOP7777
   AN=INTER
   DELTA=(UPPER-ALOW)/AN
   LL=KK+100
   WRITE(NDISK,4C00) LL
   JJ=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(AX,F)
   OBJ=OBJ-F
   GRID=X-AX
   II=IJ+KKK
WRITE(NDISK,2C00) II,OBJ,LL,GRID
IF (IJ-INTER) 3,301,301
301 KKK=KKK+INTER
NKK=NKK+INTER
400 CONTINUE
WRITE(NDISK,7C00)
22 READ(NREAD,1C01) A
IF (A(I) EQ ENCA) GO TO 23
WRITE(NDISK,1C01) A
GO TO 22
23 WRITE(NDISK,7C01)
DO I=1,NKK
II=I+1000
WRITE(NDISK,8C00) II
WRITE(NDISK,1C01) A
END FILE NDISK
STOP
END

SUBROUTINE FUMC(X,F)
COMMON KK,STAPT,NFC
DOUBLE PRECISION SS,XX,F
XX=X
IF (KK,GT,NFC) GO TO 1
XX=START
F=XX*LOG(SS/XX)
RETURN
1 F=XX*LOG(XX)
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)
//
<table>
<thead>
<tr>
<th>NAME</th>
<th>DATA-SET</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROWS</td>
<td></td>
</tr>
<tr>
<td>N OBJECT</td>
<td>GRID101</td>
</tr>
<tr>
<td>E GRID101</td>
<td>1.0 NORM 1.0</td>
</tr>
<tr>
<td>E GRID102</td>
<td>1.0 ORTH1 2.0</td>
</tr>
<tr>
<td>E GRID103</td>
<td>ORTH2 2.0</td>
</tr>
<tr>
<td>E GRID104</td>
<td>ORTH3 1.0</td>
</tr>
<tr>
<td>E GRID105</td>
<td>GRID106 1.0 ORTH4 1.0</td>
</tr>
<tr>
<td>E GRID106</td>
<td>ORTH2 1.0</td>
</tr>
<tr>
<td>E GRID107</td>
<td>ORTH4 1.0</td>
</tr>
<tr>
<td>E GRID108</td>
<td>ORTH1 1.0</td>
</tr>
<tr>
<td>E GRID109</td>
<td>ORTH3 1.0</td>
</tr>
<tr>
<td>E CONST1</td>
<td>GRID108 1.0</td>
</tr>
<tr>
<td>E CONST2</td>
<td>GRID109 1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>GRID101 1.0 NORM 1.0</td>
</tr>
<tr>
<td>W1</td>
<td>ORTH1 1.0</td>
</tr>
<tr>
<td>W2</td>
<td>GRID102 1.0 NORM 1.0</td>
</tr>
<tr>
<td>W2</td>
<td>ORTH2 1.0</td>
</tr>
<tr>
<td>W3</td>
<td>GRID103 1.0 NORM 1.0</td>
</tr>
<tr>
<td>W3</td>
<td>ORTH3 1.0</td>
</tr>
<tr>
<td>W4</td>
<td>GRID104 1.0 NORM 1.0</td>
</tr>
<tr>
<td>W4</td>
<td>ORTH4 1.0</td>
</tr>
<tr>
<td>W5</td>
<td>GRID105 1.0 ORTH1 2.0</td>
</tr>
<tr>
<td>W5</td>
<td>ORTH4 -2.0 CONST1 -1.0</td>
</tr>
<tr>
<td>W6</td>
<td>GRID106 1.0 ORTH2 2.0</td>
</tr>
<tr>
<td>W6</td>
<td>ORTH4 -2.0 CONST1 -1.0</td>
</tr>
<tr>
<td>W7</td>
<td>GRID107 1.0 ORTH1 -1.0</td>
</tr>
<tr>
<td>W7</td>
<td>ORTH2 -1.0 ORTH3 -1.0</td>
</tr>
<tr>
<td>W7</td>
<td>CONST2 -1.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SEPARABLE DATA</th>
<th>9 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.0</td>
<td>20  228  238</td>
</tr>
<tr>
<td>10.0</td>
<td>20  298  308</td>
</tr>
<tr>
<td>4.0</td>
<td>20  328  338</td>
</tr>
<tr>
<td>2.0</td>
<td>20  125  135</td>
</tr>
<tr>
<td>1.0</td>
<td>20  045  055</td>
</tr>
<tr>
<td>1.0</td>
<td>20  010  020</td>
</tr>
<tr>
<td>100.0</td>
<td>20  328  338</td>
</tr>
<tr>
<td>1.0</td>
<td>20  060  070</td>
</tr>
<tr>
<td>1.0</td>
<td>20  328  338</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RHS</th>
<th>LIMITES</th>
<th>GRID101 4.0 GRID102 10.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIMITES</td>
<td>GRID103 4.00 GRID104 2.0</td>
<td></td>
</tr>
<tr>
<td>LIMITES</td>
<td>GRID105 1.0 GRID106 1.0</td>
<td></td>
</tr>
<tr>
<td>LIMITES</td>
<td>GRID107 100.0 GRID108 1.0</td>
<td></td>
</tr>
<tr>
<td>LIMITES</td>
<td>GRID109 1.0 NORM 1.0</td>
<td></td>
</tr>
</tbody>
</table>

<p>| ENDATA |</p>
<table>
<thead>
<tr>
<th>ECHO CHECK FOR VARIABLE</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>START=</td>
<td>4.00000</td>
</tr>
<tr>
<td>INTER=</td>
<td>2C</td>
</tr>
<tr>
<td>ALCW =</td>
<td>0.2280C</td>
</tr>
<tr>
<td>UPPER=</td>
<td>0.2380C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ECHO CHECK FOR VARIABLE</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>START=</td>
<td>10.00000</td>
</tr>
<tr>
<td>INTER=</td>
<td>2C</td>
</tr>
<tr>
<td>ALCW =</td>
<td>0.2980C</td>
</tr>
<tr>
<td>UPPER=</td>
<td>0.30800</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ECHO CHECK FOR VARIABLE</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>START=</td>
<td>4.00000</td>
</tr>
<tr>
<td>INTER=</td>
<td>2C</td>
</tr>
<tr>
<td>ALCW =</td>
<td>0.3280C</td>
</tr>
<tr>
<td>UPPER=</td>
<td>0.33800</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ECHO CHECK FOR VARIABLE</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>START=</td>
<td>2.00000</td>
</tr>
<tr>
<td>INTER=</td>
<td>2C</td>
</tr>
<tr>
<td>ALCW =</td>
<td>0.12500</td>
</tr>
<tr>
<td>UPPER=</td>
<td>0.13500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ECHO CHECK FOR VARIABLE</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>START=</td>
<td>1.00000</td>
</tr>
<tr>
<td>INTER=</td>
<td>2C</td>
</tr>
<tr>
<td>ALCW =</td>
<td>0.04500</td>
</tr>
<tr>
<td>UPPER=</td>
<td>0.05500</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ECHO CHECK FOR VARIABLE</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>START=</td>
<td>1.00000</td>
</tr>
<tr>
<td>INTER=</td>
<td>2C</td>
</tr>
<tr>
<td>ALCW =</td>
<td>0.01000</td>
</tr>
<tr>
<td>UPPER=</td>
<td>0.02000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ECHO CHECK FOR VARIABLE</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>START=</td>
<td>10.00000</td>
</tr>
<tr>
<td>INTER=</td>
<td>2C</td>
</tr>
<tr>
<td>ALCW =</td>
<td>0.3280C</td>
</tr>
<tr>
<td>UPPER=</td>
<td>0.33800</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ECHO CHECK FOR VARIABLE</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>START=</td>
<td>1.00000</td>
</tr>
<tr>
<td>INTER=</td>
<td>2C</td>
</tr>
<tr>
<td>ALCW =</td>
<td>0.06000</td>
</tr>
<tr>
<td>UPPER=</td>
<td>0.07000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ECHO CHECK FOR VARIABLE</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>START=</td>
<td>1.00000</td>
</tr>
<tr>
<td>INTER=</td>
<td>2C</td>
</tr>
<tr>
<td>ALCW =</td>
<td>0.32800</td>
</tr>
<tr>
<td>UPPER=</td>
<td>0.33800</td>
</tr>
</tbody>
</table>
PROGRAM
INITIALZ
MOVE(XDATA,'DATA-SET')
MOVE(XPDBNAME,'EXAMPLE')
CONVERT
BCRUT
MOVE(XOBJ,'OBJECT')
MOVE(XRHS,'LIMITS')
TITLE('GEOMETRIC PROGRAMMING')
SETUP('BOUND','SEPBOUND','MAX','SCALE')
PRIMAL
SOLUTION
EXIT
PEND
### GEOMETRIC PROGRAMMING

**SOLUTION (OPTIMAL)**

**TIME = 0.84 MINS. ITERATION NUMBER = 125**

---

<table>
<thead>
<tr>
<th>NAME</th>
<th>ACTIVITY</th>
<th>DEFINED AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUNCTIONAL</td>
<td>4.47719</td>
<td>OBJECT</td>
</tr>
<tr>
<td>RESTRAINTS</td>
<td></td>
<td>LIMITS</td>
</tr>
<tr>
<td>BOUNDARY</td>
<td></td>
<td>SEPBOUND</td>
</tr>
</tbody>
</table>

---

### GEOMETRIC PROGRAMMING

#### SECTION 1 - ROWS

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>NAME</th>
<th>AT...</th>
<th>ACTIVITY</th>
<th>SLACK ACTIVITY</th>
<th>LOWER LIMIT</th>
<th>UPPER LIMIT</th>
<th>DUAL ACTIVITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OBJECT</td>
<td>BS</td>
<td>4.47719</td>
<td>4.47719-</td>
<td>NONE</td>
<td>NONE</td>
<td>1.000000</td>
</tr>
<tr>
<td>2</td>
<td>GRIC101</td>
<td>EC</td>
<td>4.00000</td>
<td>4.00000</td>
<td>10.00000</td>
<td>10.00000</td>
<td>1.000000</td>
</tr>
<tr>
<td>3</td>
<td>GRIC102</td>
<td>EC</td>
<td>10.00000</td>
<td>10.00000</td>
<td>10.00000</td>
<td>10.00000</td>
<td>1.000000</td>
</tr>
<tr>
<td>4</td>
<td>GRIC103</td>
<td>EC</td>
<td>2.00000</td>
<td>2.00000</td>
<td>2.00000</td>
<td>2.00000</td>
<td>1.000000</td>
</tr>
<tr>
<td>5</td>
<td>GRIC104</td>
<td>EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.000000</td>
</tr>
<tr>
<td>6</td>
<td>GRIC105</td>
<td>EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.000000</td>
</tr>
<tr>
<td>7</td>
<td>GRIC106</td>
<td>EC</td>
<td>100.00000</td>
<td>100.00000</td>
<td>100.00000</td>
<td>100.00000</td>
<td>1.000000</td>
</tr>
<tr>
<td>8</td>
<td>GRIC107</td>
<td>EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.000000</td>
</tr>
<tr>
<td>9</td>
<td>GRIC108</td>
<td>EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.000000</td>
</tr>
<tr>
<td>10</td>
<td>GRIC109</td>
<td>EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.000000</td>
</tr>
<tr>
<td>11</td>
<td>NCPH</td>
<td>EC</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.00000</td>
<td>1.000000</td>
</tr>
<tr>
<td>12</td>
<td>CTPH1</td>
<td>EC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>13</td>
<td>CTPH2</td>
<td>EC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>14</td>
<td>CTPF3</td>
<td>EC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>15</td>
<td>CTPF4</td>
<td>EC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>16</td>
<td>CCAST1</td>
<td>EC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>17</td>
<td>CCAST2</td>
<td>EC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.000000</td>
</tr>
<tr>
<td>NUMBER</td>
<td>COLUMN</td>
<td>AT</td>
<td>ACTIVITY</td>
<td>INPUT COST</td>
<td>LOWER LIMIT</td>
<td>UPPER LIMIT</td>
<td>REDUCE COST</td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>-----</td>
<td>----------</td>
<td>------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>18</td>
<td>W1</td>
<td>BS</td>
<td>23117</td>
<td></td>
<td></td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td>19</td>
<td>W2</td>
<td>BS</td>
<td>30560</td>
<td></td>
<td></td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td>20</td>
<td>W3</td>
<td>BS</td>
<td>33333</td>
<td></td>
<td></td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td>21</td>
<td>W4</td>
<td>BS</td>
<td>13050</td>
<td></td>
<td></td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td>22</td>
<td>W5</td>
<td>BS</td>
<td>65108</td>
<td></td>
<td></td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td>23</td>
<td>W6</td>
<td>BS</td>
<td>01417</td>
<td></td>
<td></td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td>24</td>
<td>W7</td>
<td>BS</td>
<td>33333</td>
<td></td>
<td></td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td>25</td>
<td>A1</td>
<td>BS</td>
<td>06925</td>
<td></td>
<td></td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td>26</td>
<td>A2</td>
<td>BS</td>
<td>33333</td>
<td></td>
<td></td>
<td></td>
<td>NONE</td>
</tr>
<tr>
<td>27</td>
<td>P1001</td>
<td>UL</td>
<td>0.000C0</td>
<td>65315</td>
<td>1.000C0</td>
<td>7.03294</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>P1002</td>
<td>UL</td>
<td>0.000F0</td>
<td>00093</td>
<td>1.000C0</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>P1003</td>
<td>UL</td>
<td>0.000C0</td>
<td>00093</td>
<td>1.000C0</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>P1004</td>
<td>UL</td>
<td>0.000C0</td>
<td>00093</td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>P1005</td>
<td>UL</td>
<td>0.000C0</td>
<td>00093</td>
<td>1.000C0</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>P1006</td>
<td>UL</td>
<td>0.000C0</td>
<td>00093</td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>P1007</td>
<td>UL</td>
<td>0.000C0</td>
<td>00093</td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>P1008</td>
<td>BS</td>
<td>33333</td>
<td>00093</td>
<td>1.000C0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>P1009</td>
<td>LL</td>
<td>00092</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>P1010</td>
<td>LL</td>
<td>00092</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>P1011</td>
<td>LL</td>
<td>00092</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>P1012</td>
<td>LL</td>
<td>00092</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>P1013</td>
<td>LL</td>
<td>00092</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>P1014</td>
<td>LL</td>
<td>00092</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>P1015</td>
<td>LL</td>
<td>00092</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>P1016</td>
<td>LL</td>
<td>00092</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>P1017</td>
<td>LL</td>
<td>00092</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>P1018</td>
<td>LL</td>
<td>00091</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>P1019</td>
<td>LL</td>
<td>00091</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>P1020</td>
<td>LL</td>
<td>00091</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>P1021</td>
<td>LL</td>
<td>00091</td>
<td></td>
<td>1.00000</td>
<td>0.00000</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>P1022</td>
<td>LL</td>
<td>0.000C0</td>
<td>016495</td>
<td>1.00000</td>
<td>25.09200</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>P1023</td>
<td>LL</td>
<td>0.000C0</td>
<td>00126</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>P1024</td>
<td>LL</td>
<td>0.000C0</td>
<td>00126</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>P1025</td>
<td>LL</td>
<td>0.000C0</td>
<td>00126</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>P1026</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>P1027</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>P1028</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>P1029</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>P1030</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>P1031</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>P1032</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>P1033</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>P1034</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>P1035</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>P1036</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>P1037</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>P1038</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>P1039</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>P1040</td>
<td>LL</td>
<td>0.000C0</td>
<td>00125</td>
<td>1.00000</td>
<td>0.00001</td>
<td></td>
</tr>
</tbody>
</table>
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.
SUBROUTINE PHX
DIMENSION A(150, 100), ALFA(50, 50), X(150), XC(150), XI(150), XS(150) GEGA 20
1, Y(150), C(150), VC(150), IBAS(50), IMH(100), IVC(50), IVA(100) GEGA 30
COMMON A, ALFA, X, XC, XI, XS, YC, VC, IBAS, IMH, 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, IRI, IS, IIS, IT, IBP, ICDB, JCDB, KCID, KCGEA GEGA 60
COMMON KFCNC, KGRAD, KCON, KINV1, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KGEA 70
IREN1, KRE2, KINV, KCDBA1, KRENL1, KRENL2, KRENL3 GEGA 80
COMMON ITRIC, DELT, ETA, JK0, LC, YSORT GEGA 90
PHI=2.0*X(1)-0.5*X(1)**2+3.0*X(2)-0.5*X(2)**2
RETURN
END

SUBROUTINE CPHI
DIMENSION A(150, 100), ALFA(50, 50), X(150), XC(150), XI(150), XS(150) GEGA 20
1, Y(150), C(150), VC(150), IBAS(50), IMH(100), IVC(50), IVA(100) GEGA 30
COMMON A, ALFA, X, XC, XI, XS, YC, VC, IBAS, IMH, 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, IRI, IS, IIS, IT, IBP, ICDB, JCDB, KCID, KCGEA GEGA 60
COMMON KFCNC, KGRAD, KCON, KINV1, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KGEA 70
IREN1, KRE2, KINV, KCDBA1, KRENL1, KRENL2, KRENL3 GEGA 80
COMMON ITRIC, DELT, ETA, JK0, LC, YSORT GEGA 90
VC(1) = X(1)**2 + X(2)**2 - 1.0
RETURN
END
SUBROUTINE JACOB
DIMENSION A(0,100), 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, NVNINL, NVNIN2, NVNIN3, INDEX, II, IR, IR1, IS, IS1, IT, IBP, ICDB, JCDB, KCGEGA 60
2DB, KFIL, KLIN, KREN, KD, FI1, PHI, PSI1, PSI3, TB, TD, TC, EPSIL, EPSIL0, EPSIL2GEA 70
COMMON KFNC, KGRA, KCONT, KINV1, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KCGEGA 80
1REN1, KREN2, KINV, KCDBA1, KREN1L, KREN2L GEGA 90
COMMON IDREC, DELTFI, ETA, JK0, LC, YSORT GEGA 100
C006 A(1,1) = 2.0 - X(1)
C007 A(1,2) = 2.0 - X(2)
C009 RETURN
C010 END

SUBROUTINE GRADFI
DIMENSION A(0,100), 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, NVNINL, NVNIN2, NVNIN3, INDEX, II, IR, IR1, IS, IS1, IT, IBP, ICDB, JCDB, KCGEGA 60
2DB, KFIL, KLIN, KREN, KD, FI1, PHI, PSI1, PSI3, TB, TD, TC, EPSIL, EPSIL0, EPSIL2GEA 70
COMMON KFNC, KGRA, KCONT, KINV1, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KCGEGA 80
1RE'1, KREN2, KINV, KCDBA1, KREN1L, KREN2L GEGA 90
COMMON IDREC, DELTFI, ETA, JK0, LC, YSORT GEGA 100
C007 C(1) = 2.0 - X(1)
C008 C(2) = 3.0 - X(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

MV  2
NIV  1
NP2  1
NWEVL  20
MTD  6
ITG  20
LCONJ  9
LMIAG  6
LTMAX  20
KFL  0
KLIR  0
NCO  10
NISO  1
JSI3SA  20
EPSIL  0.1E 00
EPSILO  0.1E-06
EPSILI  0.1E-04
EPSIL2  0.1E-04
PC  0.1E 01

GRADIENT RENIQUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES  2
NOMBRE TOTAL DE VARIABLES  3
NOMBRE DE CONTRAINTES  1
EPSILON DE NEWTON  0.100E-06
EPSILON TEST GRADIENT  0.100E-04

FOUCTION ECONOMIQUE  0.22500000000000E 01

BCRAE INFERIEUR    VARIABLE NATURIELLE    BORNE SUPERIEURE

XII (1)  9.0  
XII (2)  9.0  
X (1)  0.99999999999999E 00  
X (2)  0.99999999999999E 00  
XS (1)  0.10000000000000E 02  
XS (2)  0.10000000000000E 02

VALEUR DES CONTRAINTES

(1 1)  -0.99999999999999E 00
IT 1  PHI 0.27466024375000E 01  DIR.GRAD. NO 0  YN 0.29E 01  DELTA.FI 0.38E 02  ETA 0.24E 02  NCDH 0  NCA 2  VTR 2
IT 2  PHI 0.29303584499101E 01  DIR.GRAD. NO 0  YN 0.27E 01  DELTA.FI 0.39E 02  ETA 0.22E 02  NCDH 0  NCA 2  VTR 2
IT 3  PHI 0.30653376117981E 01  DIR.GRAD. NO 0  YN 0.27E 01  DELTA.FI 0.39E 02  ETA 0.21E 02  NCDH 0  NCA 2  VTR 2
IT 4  PHI 0.30774073699141E 01  DIR.GRAD. NO 0  YN 0.26E 01  DELTA.FI 0.34E 02  ETA 0.21E 02  NCDH 1  NCA 3  VTR 11
IT 5  PHI 0.31054845639437E 01  DIR.GRAD. NO 1  YN 0.39E 00  DELTA.FI 0.39E 00  ETA 0.39E 00  NCDH 0  NCA 4  VTR 23
IT 6  PHI 0.31105481231549E 01  DIR.GRAD. NO 1  YN 0.39E 00  DELTA.FI 0.39E 00  ETA 0.39E 00  NCDH 0  NCA 4  VTR 23
IT 7  PHI 0.31055507659412E 01  DIR.GRAD. NO 1  YN 0.39E 00  DELTA.FI 0.10E 02  ETA 0.10E 02  NCDH 0  NCA 3  VTR 1
PHI 0.31055507659412E 01

DUREE DU CALCUL 50 CENTISECONDES

VARIABLE NATURELLE

| Xi 1 | Xi 2 |
| 0.05459195375544E 00 | 0.83212280213438E 00 |

VARIABLE DUALE ASSOCIEE

| Vi 1 | Vi 2 |
| 0.56648254394531E-03 | 0. |

CONTRAINTES

CONTRAINTE 1 0.0

VUE DE SORTIE

NOMBRE DE 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 MONTRE DANS LE DOMAINE 19

NUMEVT TOTAL D'ITERATIONS 44
SUBROUTINE PHI

DIMENSION ALFA(100), ALFA(50,50), X(150), XCI(150), XI(150), XS(150) GEGA 20
1, Y(150), C(150), VC(50), IBAS(50), IHB(150), IVC(150), IVA(100) GEGA 30
COMMON A, ALFA, X, XCI, XI, XS, Y, VC, IBAS, IHB, IVC, IVA, IVA
COMMON NV, NC, VK, NIN, NTV, NV1, NEV, NEVL, NTO, N1N2, N1N3, NIGEGA 50
1N4, NVNIN1, NVNIN2, NVNIN3, INDEX1, I1, I1R, I1R1, IS1, IT, IBP, ICDB, JCD, KGEA 60
200, KFICL, KLIN, KREN1, KREN2, KFIL, PHI, PSI, PSII, TB, TC, EPSI1, EPSI0, EPSI2, GEGA 70
COMMON KFICL, KGRAO, KCONT, KINVC, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KGEA 80
1REM1, KREN2, KINV, KCDBA1, KREN1, KRE21 GEGA 90
COMMON IDIREC, DELTFI, ETA, JKO, LC, YSORT GEGA 100

XX1=1.0-XC(1)
XX3=1.0-XC(3)
XX4=1.0-XC(4)
PHI=1.0-XC(3)*(XX1*XX4)**2-XX3*(1.0-XC(2)*((1.0-XX1*XX4)**2)
RETURN
END

SUBROUTINE CPHI

DIMENSION A(50,100), ALFA(50,50), X(150), XCI(150), XI(150), XS(150) GEGA 20
1, Y(150), C(150), VC(50), IBAS(50), IHB(150), IVC(150), IVA(100) GEGA 30
COMMON A, ALFA, X, XCI, XI, XS, Y, VC, IBAS, IHB, IVC, IVA, IVA
COMMON NV, NC, VK, NIN, NTV, NV1, NEV, NEVL, NTO, N1N2, N1N3, NIGEGA 50
1N4, NVNIN1, NVNIN2, NVNIN3, INDEX1, I1, I1R, I1R1, IS1, IT, IBP, ICDB, JCD, KGEA 60
200, KFICL, KLIN, KREN1, KREN2, KFIL, PHI, PSI, PSII, TB, TC, EPSI1, EPSI0, EPSI2, GEGA 70
COMMON KFICL, KGRAO, KCONT, KINVC, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KGEA 80
1REM1, KREN2, KINV, KCDBA1, KREN1, KRE21 GEGA 90
COMMON IDIREC, DELTFI, ETA, JKO, LC, YSORT GEGA 100

VW(1)=200.0*XC(1)**0.6+200.0*XC(1)**0.6+200.0*XC(3)**0.6
1+300.0*XC(4)**0.6-800.0
RETURN
END
PARAMETRES

GV  4
GV  1
HOL  6
INFT  20
JCOUJ  1
JCOUJ  6
JVAR  9C
LIN  0
M  1
N  1
P  1
EPSIL  0.1E-06
EPSIL  0.1E-06
EPSILZ  0.1E-06
NC  0.1E 61

GRADIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES 4
NOMBRE TOTAL DE VARIABLES 5
NOMBRE DE CONTRAINTES 4

EPSILON DE RAYON 0.10E-05
EPSILON TEST GRADIENT 0.10E-05

FONCTION ECONOMIQUE 0.BP623362779617E 06

<table>
<thead>
<tr>
<th>BORNE INFÉRIEURE</th>
<th>VARIABLE NATURELLE</th>
<th>BORNE SUPÉRIEURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>XI  1</td>
<td>0.50000000E+00 00</td>
<td>XI  1</td>
</tr>
<tr>
<td>XI  2</td>
<td>0.50000000E+00 00</td>
<td>XI  2</td>
</tr>
<tr>
<td>XI  3</td>
<td>0.50000000E+00 00</td>
<td>XI  3</td>
</tr>
<tr>
<td>XI  4</td>
<td>0.50000000E+00 00</td>
<td>XI  4</td>
</tr>
</tbody>
</table>

VALEUR DES CONTRAINTES

C1  1 | 0.135095654E+03 03
SUBROUTINE PHX
C002
DIMENSION A(F0,100),ALFA(50,50),X(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,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVD GEGA 40
C004
COMMON NV,NK,NEG,NH,NV1,NV2,NEV1,NEV2,NTO,NV1,NV2,NIN1,NIN2,NIM3,NGEGA 50
11,N1,N11,N2,N21,N5,N3,N51,N52,NIN1,NIN2,NIN3,N1GEGA 60
C005
COMMON KFNC,KGNC,KVRNC,KV1NC,KINV1,KV2NC,KVDB1,KVJACO,KMAX1,KMAX2,KGEGA 70
20,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA 80
C006
COMMON INTREC,DELTFI,ETA,JKD,LC,YSORT GEGA 90
C007
PHI=200.0*X(1)***.6+200.0*X(2)***.6+200.0*X(3)***.6
1.0+1.0*X(4)***.6
C008
PHI=-PHI
C009
RETURN
C010
END

SUBROUTINE CPHI
C001
DIMENSION A(F0,100),ALFA(50,50),X(150),XI(150),XS(150) GEGA 20
1,Y(150),C(150),VC(50),IBAS(50),IHB(100),IVC(50),IVA(100) GEGA 30
C002
COMMON A,ALFA,X,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVD GEGA 40
C003
COMMON NV,NK,NEG,NH,NV1,NV2,NEV1,NEV2,NTO,NV1,NV2,NIN1,NIN2,NIN3,NGEGA 50
11,N1,N11,N2,N21,N5,N3,N51,N52,NIN1,NIN2,NIN3,N1GEGA 60
C004
COMMON KFNC,KGNC,KVRNC,KV1NC,KINV1,KV2NC,KVDB1,KVJACO,KMAX1,KMAX2,KGEGA 70
20,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA 80
C005
COMMON INTREC,DELTFI,ETA,JKD,LC,YSORT GEGA 90
C006
COMMON INTREC,DELTFI,ETA,JKD,LC,YSORT GEGA 100
C007
XX1=1.0-X(1)
C008
XX3=1.0-X(3)
C009
XX4=1.0-X(4)
C010
VC(1)=.0-X(1)*X(2)**2-XX3*(1.0-X(2)*X(4))*2
C011
VC(1)=-VC(1)+.9
C012
RETURN
C013
END
DUREE DU CALCUL: **40 CHUTES/SECUNDES**

**VARIABLE NUMERIQUE**

\[ \begin{align*}
\text{XI}(1) &= 0.550000000000000000 \times \text{XI}(1) \\
\text{XI}(2) &= 0.831250000000000000 \times \text{XI}(2) \\
\text{XI}(3) &= 0.243750000000000000 \times \text{XI}(3) \\
\text{XI}(4) &= 0.831250000000000000 \times \text{XI}(4)
\end{align*} \]

**VARIABLE NUMERIQUE ASSOCIEE**

\[ \begin{align*}
\text{VI}(1) &= -0.286410028078761708 \\
\text{VI}(2) &= 0.0 \\
\text{VI}(3) &= -0.014644621017060232 \\
\text{VI}(4) &= -0.1076814587402309
\end{align*} \]

**CONSTRAIT**

\[ -0.119002795950000000 \times \text{UI}(1) = 0.096289292285638032 \]

**NIVEAU DE SORTIE**

\[ 1 \]

**APPELS**

1. CALCLINE
2. CALCULATION LES CONTRAINTES 40
3. CALCLINE LA FONCTION ECONOMIQUE 30
4. APPELS ON SAP CALCULANT LE GRADIENT DE LA FONCTION ECONOMIQUE 12
5. APPELS ON SAP CALCULANT L'INVERSAIRE AVEC BASE INVERSE 0
6. APPELS ON SAP CALCULANT L'INVERSAIRE AVEC RECHERCHE DE PASS 0
7. APPELS ON SAP EFFETUANT LES CONTRAINTES DE BASE 1
8. APPELS ON SAP CALCULANT LE JACOBIN DES CONTRAINTES 7
9. APPELS ON SAP EFFETUANT UNE RECHERCHE DE MAX SANS MIGRATION 4
10. APPELS ON SAP EFFETUANT UNE RECHERCHE DE MAX AVEC MIGRATION 0
11. APPELS ON SAP EFFETUANT LA PRATIQUE AVEC LE жерат 14

**TOTAL D'ITÉRATIONS**
SURROUTINE PHIX

C DIMENSION A(50,100),ALFA(50,50),X(150),XI(150),XS(150) GEGA 20

1,Y(150),C(150),VC(50),IBAS(50),IMB(1CC),IVC(50),IVA(100) GEGA 30

COMMON A,ALFA,X,VC,XI,XS,Y,C,VC,IBAS,IMB,IVC,IVA,IVB GEGA 40

COMMON NV,NC,NK,NEG,NIN,NTV,AV1,NEV,NEV1,NTO,NIN1,NIN2,NIN3,NGEGA 50

1N4,NVNIN1,NVNIN2,NVNIN3,INDEX,II,IR,IR1,IS,IS1,IT,ITB,ICDB,JCDB,KCSEG GEGA 60

2B,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPSIL,EPSIL0,EPSIL2GEGA 70

COMMON KFCNC,KGRAC,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KEGA 80

1REN1,KREN2,KINV,KCDBA1,KREN11,KREN21 GEGA 90

COMMON IDIREC,DELTFI,ETA,JKO,LC,YSORT GEGA 100

COMMON B(5),CC(3,5)

IF(IT)100,101,101

100 B(1)=ALOG(0.2)

B(2)=ALOG(0.1)

B(3)=ALOG(0.1)

B(4)=ALOG(0.3)

B(5)=ALOG(0.25)

PHI=C.0

C015 PHI=C.1,5

102 PHI=PHI+ALOG(1.0-EXP(B(1)*XC(1)))

RETURN

END
SUBROUTINE CPH1

DIMENSION ALFA(50,50), X(150), Y(150), Z(150), XE(150), XI(150), XS(150)

COMMON A(ALFA), X, XE, XI, XS, Y, Z, C, VC, IBAS, ITB, IVC, IVA, IVB

COMMON NV, NC, NK, NNE, NIN, NTN, NV, NEV, Nevada, NTO, NIN1, NIN2, NIN3, NIGE

COMMON TN, TNF, NVNF, TNF2, NNVNF3, INDEX, IT, ITR, ITR1, IT1, ITB1, ICDB, ICDNB, ICGBA

COMMON KCNC, KGRAD, KCOND, KINV1, KINV2, KCDNB, KCABA, KJACO, KMAX1, KMAX2, KCILA

COMMON IREN1, KREN2, KINV, KCDABA, KREN11, KREN21

COMMON B(5), C(3,5)

100 CC(1,1)=1.0

101 VC(1)=0.0

102 VC(1)=VC(1)+CC(1,J)*XC(J)**2

103 VC(2)=VC(2)+CC(2,J)*(XC(J)+EXP(XC(J)/0.01))

104 VC(3)=VC(3)+CC(3,J)*XC(J)*EXP(XC(J)/0.01)

RETURN

END
SUBROUTINE JACOB
CIMENSION A(F0,100),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,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB GEGA 40
COMMON NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEWL,NTO,NIN1,NIN2,NIN3,NGE GEGA 50
1A4,NVVNN1,NVVNN2,NVVNN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KGEGA 60
2CB,KFIL,KLIN,KREN,KD,FI1,PS1,PSI3,TB,TD,TC,EP,SIL,EP,SILO,EP,SI2EGA 70
COMMON KFCN,KGR,C,KGON,KNON1,KNON2,KCDBA,KJACO,KMAX,KMAX2,KGEGA 80
IREN1,KREN2,KINV,KCDBA,KREN1,KREN2 GEGA 90
COMMON IDREC,DELTFI,ETA,JKO,LC,YSORT GEGA 100
CO08 COMMON B(5),CC(3,5) GEGA 100
CO09 CO101J=1,5
CO11 CO102J=1,5
CO12 CO103J=1,5
CO13 CO14 RETURN
CO15 END

SUBROUTINE GRADFI
CIMENSION 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,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB GEGA 40
COMMON NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEWL,NTO,NIN1,NIN2,NIN3,NGE GEGA 50
1A4,NVVNN1,NVVNN2,NVVNN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KGEGA 60
2CB,KFIL,KLIN,KREN,KD,FI1,PS1,PSI3,TB,TD,TC,EP,SIL,EP,SILO,EP,SI2EGA 70
COMMON KFCN,KGR,C,KGON,KNON1,KNON2,KCDBA,KJACO,KMAX,KMAX2,KGEGA 80
IREN1,KREN2,KINV,KCDBA,KREN1,KREN2 GEGA 90
COMMON IDREC,DELTFI,ETA,JKO,LC,YSORT GEGA 100
CO06 COMMON B(5),CC(3,5) GEGA 100
CO07 COMMON B(5),CC(3,5) GEGA 100
CO08 CO101J=1,5
CO09 AA=EXP(B(J)*X(J))
CO10 CO11 RETURN
CO12 END
PARAMETRES

NY  5
NIV  3
NEG  0
NEVL  20
NTO  6
ITET  20
ICDAJ  1
ICIG  0
ITMAX  5C
KFI  0
KLI  0
NCO  1C
ITSCR  1
ISOLA  50
EPSIL  0.1E-00
EPSIL0  0.3E-04
EPSIL  0.1E-02
EPSIL2  0.1E-02
PC  0.0C

GRADIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES  5
NOMBRE TOTAL DE VARIABLES  8
NOMBRE DE CONTRAINTE  3
EPSILON DE NEUTON  0.1000E-04
EPSILON TEST GRADIENT  0.1000E-02

FUNCTION ECONOMIQUE  -0.82864609460831E 00

Borne Inférieure  Variable Naturelle  Borne Supérieure

XII  1)  0.1000000000000000E 01  X(1)  0.2000000000000000E 01  X(1)  0.5000000000000000E 01
XII  2)  0.1000000000000000E 01  X(2)  0.2000000000000000E 01  X(2)  0.5000000000000000E 01
XII  3)  0.1000000000000000E 01  X(3)  0.2000000000000000E 01  X(3)  0.5000000000000000E 01
XII  4)  0.1000000000000000E 01  X(4)  0.2000000000000000E 01  X(4)  0.5000000000000000E 01
XII  5)  0.1000000000000000E 01  X(5)  0.2000000000000000E 01  X(5)  0.5000000000000000E 01

VALEUR DES CONTRAINTE

C1  1)  -0.3000000000000000E 02
C1  2)  -0.31240434279771E 02
C1  3)  -0.39646071747256E 02
CURÉE DU CALCUL 286 CENTISECONDES

VARIABLE NATURELLE

\[
\begin{align*}
X_1 & = 0.26779568176276 \times 10^{1} \\
X_2 & = 0.3536310653876 \times 10^{1} \\
X_3 & = 0.27920246124766 \times 10^{1} \\
X_4 & = 0.23526436826716 \times 10^{1} \\
X_5 & = 0.2670942159302 \times 10^{1} \\
\end{align*}
\]

VARIABLE DUALE ASSOCIÉE

\[
\begin{align*}
V_1 & = -0.27788754663931 \times 10^{-1} \\
V_2 & = 0.1388530490875 \times 10^{-1} \\
V_3 & = 0.1870840788763 \times 10^{-1} \\
V_4 & = 0.19506737586836 \times 10^{-1} \\
V_5 & = 0.28701246176276 \\
\end{align*}
\]

CONTRAINTE

\[
\begin{align*}
\text{CONTRAINE 1} & = -0.13493397216 \times 10^{1} \\
\text{CCONTRAINE 2} & = -0.2303663948438 \times 10^{1} \\
\text{CONTRAINE 3} & = -0.13493397216 \times 10^{1} \\
\end{align*}
\]

VARIABLE DUALE ASSOCIÉE

\[
\begin{align*}
U_1 & = 0.0 \\
U_2 & = 0.0 \\
U_3 & = 0.9608564276276 \times 10^{1} \\
\end{align*}
\]

223
<table>
<thead>
<tr>
<th>NIVEAU DE SORTIE</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOMBRE D'ITERATIONS</td>
<td>9</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P CALCULANT LES CONTRAINTES</td>
<td>147</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P CALCULANT LA FONCTION ECONOMIQUE</td>
<td>110</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P CALCULANT LE GRADIENT DE LA FONCTION ECONOMIQUE</td>
<td>75</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P CALCULANT L'INVERSE AVEC BASE DONNÉE</td>
<td>6</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P CALCULANT L'INVERSE AVEC RÉCHERCHE DE BASE</td>
<td>1</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P EFFECTUANT LES CHANGEMENTS DE BASE</td>
<td>1</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P CALCULANT LE JACOBIEN DES CONTRAINTES</td>
<td>11</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P EFFECTUANT UNE RÉCHERCHE DE MAX SANS DICHOTOMIE</td>
<td>12</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P EFFECTUANT UNE RÉCHERCHE DE MAX AVEC DICHOTOMIE</td>
<td>3</td>
</tr>
<tr>
<td>NOMBRE D'APPELS DU S/P EFFECTUANT LA RENTREE DANS LE DOMAINE</td>
<td>31</td>
</tr>
<tr>
<td>NOMBRE TOTAL D'ITERATIONS</td>
<td>130</td>
</tr>
</tbody>
</table>
SUBROUTINE PHI

DIMENSION A(*,100), ALFA(50,50), X(150), XC(150), XI(150), XS(150)

COMMON A, ALFA, X, XC, XI, XS, Y, C, VC, IBAS, IHB, IVC, IVA, IVB

COMMON NV, NC, NK, NEG, NIN, NTV, NV1, NEV, NEL, NTO, NIN1, NIN2, NIN3, NIG

1N4, NVN1, NVN2, NVN3, INDEX, II, IR, I1, IS, IS1, IT, IH1, IC1D, JC1D, KIG

2DB, KFI, KLIN, KFN, KO, F1, PHI, PSI, PSI1, T0, T01, T1, EPSIL, EPS10, EPS1L2, KIG2

COMMON KFNC, KGRAD, KCONT, KINV1, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KEG

1REN1, KREN2, KI1M, KCD1B, KREN3, KRE21

COMMON IDIREC, DELTFI, ETA, JK0, LC, YSORT

DIMENSION B(10)

COMMON B

IF (IT.GE.0) GO TO 100

READ (5,1001) (B(I), I=1,10)

100 FORMAT (10F6.4)

CONTINUE

PHI = -8.*XC(1)*-3-16.*XC(2)*-3-20.*XC(3)*-12.*XC(4)*-3-4.*XC(5)

1*3-30.*XC(1)*-2-39.*XC(2)*-2-10.*XC(3)*-2-39.*XC(4)*-2-30.*XC(5)

2*2+40.*XC(1)*-2.*XC(2)+20.*XC(3)+64.*XC(1)*-2.*XC(4)+20.*XC(1)+2.*XC(3)

3*12.*XC(2)*XC(3)+62.*XC(2)*XC(4)-64.*XC(2)*XC(5)+12.*XC(3)*XC(4)

40.*XC(3)*XC(5)+40.*XC(4)*XC(5)

GO TO 20

CONTINUE

RETURN

END
SUBROUTINE CPHI
DIMENSION A(150,100),ALFA(150,50),XI(150),XC(150),XI(150),XS(150)

COMMON NV,XC,NK,NEG,NIN,NTV,NVL,NEV,NV,NTO,NIN1,NIN2,NIN3,

1NV,NVNIN1,NVNIN2,NVPNIN3,INDEX,II,IR,IR1,IS,ISL,IT,IMP,ICDB,JCDB,KEG

COMMON KFCNC,KGRAD,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KEG

IREN1,KREN2,KINV,KCDBA1,KREN1,KREN2

COMMON IDTREC,DELTFL,ETA,JKO,LC,YSORT

IF (I.TE.0) GO TO 103
READ (5,*01) (I,J,AIL,J,J,L=1,37)

1001 FORMAT (6(213,F6.0)/6(213,F6.0),/6(213,F6.0)/6(213,F6.0)/6(213,F6.0)/6(213,F6.0))

A(4,16) = 1.
A(5,17) = 1.
A(1,2) = 40.
A(1,3) = 20.
A(1,4) = -64.
A(1,5) = 20.
A(2,1) = 40.
A(2,3) = 12.
A(2,4) = 62.
A(2,5) = -64.
A(3,1) = 20.
A(3,2) = 12.
A(3,4) = 20.
A(3,5) = 20.
A(4,1) = -64.
A(4,2) = 62.
A(4,3) = 12.
A(4,5) = 40.
A(5,1) = 20.
A(5,2) = -64.
A(5,3) = 20.
A(5,4) = 40.

103 CONTINUE

VC(1) = 15.-6*C.*XC(1)+40.*XC(2)+20.*XC(3)-64.*XC(4)+20.*XC(5)-12.*GEX

1XC(1)**2

VC(2) = 27.+4*C.*XC(1)-78.*XC(2)+12.*XC(3)+62.*XC(4)-64.*XC(5)-24.*GEX

1XC(2)**2

VC(3) = 36.+2*C.*XC(1)+12.*XC(2)-20.*XC(3)+12.*XC(4)+20.*XC(5)-30.*GEX

1XC(3)**2

VC(4) = 18.-64.*XC(1)+62.*XC(2)+12.*XC(3)-78.*XC(4)+40.*XC(5)-18.*GEX

1XC(4)**2

VC(5) = 12.+2*C.*XC(1)+64.*XC(2)+20.*XC(3)+40.*XC(4)-60.*XC(5)-6.*GEX

1C(5)**2

DO 30 I=1,5
VC1 = 0.
DO 25 J=6,15
VC1 = VC1+A(1,J)*XC(J)

25 CONTINUE

VC(1) = VC(1)+VC1

30 CONTINUE

VC(4) = VC(4)+XC(16)

VC(5) = VC(5)+XC(17)

RETURN

END

226
SUBROUTINE JACOB
DIMENSION A(50,100), ALFA(50,50), XI(150), XC(150), XI(150), XS(150) 
1, Y(150), C(150), VC(50), IBAS(50), IHB(100), IVC(50), IVA(100)
COMMON A, ALFA, X, XS, XI, X, C, VC, IBAS, IHB, IVC, IVA, IVB
COMMON NV, NC, NK, NEG, NIN, NIV, NTV, NVL, NEV, NEVLYNTO, NIN3, NIN2, NIN1, NGE
1N4, NNVIN1, NNVIN2, NNVIN3, INDEX, II, IR, IR1, IS, IS1, IT, IDP, ICDB, JCDB, KGE
2NB, KF1L, KLIN, KREN, KD, FI1, PHI, PSI, PSI3, TB, TD, XE, EPSIL, EPSIL0, EPSIL2
COMMON KFCN, KGRAD, KCN, KIN1, KINV, KIN2, KCD, KJACO, KMAX1, KMAX2, KGE
1REN1, KREN2, KINV1, KCD1, KREN11, KREN21
COMMON IDIREC, DELTI, ETA, JKO, LC, YSORT
DIMENSION B(10)
COMMON B
A(1,1) = -24.0 * XC(1) - 60.0
A(2,2) = -48.0 * XC(2) - 78.0
A(3,3) = -60.0 * XC(3) - 20.0
A(4,4) = -36.0 * XC(4) - 78.0
A(5,5) = -12.0 * XC(5) - 60.0
RETURN
END
SUBROUTINE GPADFI
DIMENSION A(F0,100),ALFA(150,50),X(150),XC(150),XI(150),XS(150)
COMMON A,ALFA,X,XX,XI,XS,Y,C,VC,IBAS,IBH,IVC,IVA,IVB
COMMON NV,NC,NK,NEG,NIN,NTV,NV1,NV2,NEVL,NTO,NN1,NN2,NIN3,NICE
COMMON NVN1,NVIN2,NVIN3,INDEX,I1,I2,I1R,I1R1,I5,I5S,I1T,IT,ITO,I1DDO,ICDDO,KCGEGN
COMMON KBPNC,KGCR,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KCGEGN
COMMON IDIREC,DELTFI,ETA,JKO,LC,YSORT
DIMENSION B(100)
COMMON B
IF (IT.GE.0) GO TO 100
DO 10 I=6,15
C(I) = B(I-5)
10 CONTINUE
100 CONTINUE
C(1) = -24.*XG(1)**2-60.*XC(1)+40.*XC(2)+20.*XC(3)-64.*XC(4)+20.*XGEGN
1C(5) = GEGN 190
C(2) = -48.*XG(1)**2+40.*XC(1)-78.*XC(2)+12.*XC(3)+62.*XC(4)-64.*XGEGN
1C(5) = GEGN 210
C(3) = -60.*XG(1)**2+20.*XC(1)+12.*XC(2)-20.*XC(3)+12.*XC(4)+20.*XGEGN
1C(5) = GEGN 230
C(4) = -36.*XG(1)**2-64.*XC(1)+62.*XC(2)+12.*XC(3)-78.*XC(4)+40.*XGEGN
1C(5) = GEGN 250
C(5) = -12.*XG(1)**2+20.*XC(1)-64.*XC(2)+20.*XC(3)+40.*XC(4)-60.*XGEGN
1C(5) = GEGN 270
RETURN
END

PARAMETRES

NV 17
NIN 3
NEG 2
NEVL 20
NTO 6
ITET 20
ICNJ 1
IDIA 1
ITMAX 5C
KFIN 0
KLIN 0
NCO 10
ITSF 1
TSOLR 5C
EPSIL 0.1E00
EPSIL0 0.1E-03
EPSIL1 0.1E-03
EPSIL2 0.1E-03
PC 0.0
**GRADIENT REDUIT GENERALISE**

<table>
<thead>
<tr>
<th>NOMBRE DE VARIABLES NATURELLES</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOMBRE TOTAL DE VARIABLES</td>
<td>23</td>
</tr>
<tr>
<td>NOMBRE DE CONTRAINTES</td>
<td>5</td>
</tr>
<tr>
<td>EPSILON DE NEWTON</td>
<td>0.100E-03</td>
</tr>
<tr>
<td>EPSILON TEST GRADIENT</td>
<td>0.100E-03</td>
</tr>
</tbody>
</table>

**FONCTION ECONOMIQUE**  
-0.8f00000000000000E 01

<table>
<thead>
<tr>
<th>variable naturelle</th>
<th>borne inferieur</th>
<th>borne superieur</th>
</tr>
</thead>
<tbody>
<tr>
<td>x( 1)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 2)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 3)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 4)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 9)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 5)</td>
<td>0.10000000000000E 01</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 6)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 7)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 8)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 11)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 12)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 13)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 14)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 15)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 16)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
<tr>
<td>x( 17)</td>
<td>0.0</td>
<td>0.10000000000000E 03</td>
</tr>
</tbody>
</table>

**VALEUR DES CONTRAINTES**

<table>
<thead>
<tr>
<th>ct</th>
<th>valeur</th>
</tr>
</thead>
<tbody>
<tr>
<td>ct 1</td>
<td>0.41000000000000E 02</td>
</tr>
<tr>
<td>ct 2</td>
<td>-0.26000000000000E 02</td>
</tr>
<tr>
<td>ct 3</td>
<td>0.72000000000000E 02</td>
</tr>
<tr>
<td>ct 4</td>
<td>0.79000000000000E 02</td>
</tr>
<tr>
<td>ct 5</td>
<td>0.0</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.790753535400000E 11</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.168571658240000E 11</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.146504797600000E 11</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.149107015400000E 11</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.141035078800000E 11</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.695257340000000E 10</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.520794184000000E 10</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.312465624500000E 10</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.11303720297713E 03</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.6525649209858E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.5915753171821E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.4290324061855E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.39302539686652E 03</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.3632396240234E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.37773712158203E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.37434251948242E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.3704393053731E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.3677925532264E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.36577871399259E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.3701336649507E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.3271161337180E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.3224722300315E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.326250976563E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.326021423399E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.3250156278758E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.325151364135742E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.325049749023E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.3239018279242E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.325015869145E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.325493849151E 02</td>
</tr>
<tr>
<td>IT</td>
<td>PH1-0.3248739524023E 02</td>
</tr>
</tbody>
</table>

DUREE DU CALCUL 1288 CENTISECONDES
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 ÉCONOMIQUE 298

NOMBRE D'APPELS DU S/P CALCULANT LE GRADIENT DE LA FONCTION ÉCONOMIQUE 104

NOMBRE D'APPELS DU S/P CALCULANT L'INVERSE AVEC BASE DONNÉE 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 RENTRÉE DANS LE DOMAINE 115

NOMBRE TOTAL D'ITERATIONS 357
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),IHBI(100),IVC(50),IVA(100) GEGA 30
COMMON A,ALFA,X,VC,XI,XS,Y,C,VC,IBAS,IHBI,IVC,IVA,IVB GEGA 40
COMMON NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTG,NIN1,NIN2,NIN3,NI2 GEGA 50
1N4,NVIN1,NVIN2,NVIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBP,ICDB,JCDB,KGEGA 60
2CB,KFIL,KLIN,KREN,KD,FI1,PHI,PSI,PSI3,TB,TD,TC,EPS1,EPS10,EPS12,KGEGA 70
COMMON KFCNC,KGRAC,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KGEGA 80
1REN1,KREN2,KINV,KGDBA1,KREN11,KREN21 GEGA 90
COMMON B15,CC(3,4) GEGA 100
COMMON B15,CC(3,4) 

100 B(1)=-40792.14
B(2)=37.229324
B(3)=0.835689
B(4)=5.357855

110 PHI=B(1)+B(2)*XC(1)+B(3)*XC(1)*XC(5)+B(4)*XC(3)**2
C013 PHI=PHI+30373.94
C014 PHI=PHI
C015 RETURN
C016 END
SUBROUTINE CPHI

DIMENSION A(0,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)

1,Y(150),C(150),VC(50),IBAS(50),IHB(150),IVC(50),IVA(100)

COMMON A,ALFA,X,VC,VC,VC,VC,IVC,IVA,IVB

COMMON NV,NC,NK,NEG,KIN,NTV,NV1,NEV,NEVL,NTO,NII,NIN2,NIN3,NIN4

1K4,KVIN1,KVIN2,KVIN3,KINDEX,II,IR,IR,IS,IS1,IT,IBP,ICDB,JCDH,JCSTAR

COMMON KFNC,KGRAC,KCON,TI,KM1,KM2,TCDBA,MCJAC,MCX1,MCX2,MCX3

COMMON IC,ICMN,DELIT,ETA,JKO,LC,YSORT

100 CC(1,1)=95.23641

CC(1,2)=0.003886

CC(1,3)=0.004126

CC(1,4)=0.002265

CC(2,1)=80.51749

CC(2,2)=0.0072132

CC(2,3)=0.002496

CC(2,4)=0.002181

CC(3,1)=0.300961

CC(3,2)=0.004703

CC(3,3)=0.001855

CC(3,4)=0.001908

110 VC(1)=CC(1,1)*CC(1,2)*XC(2)*XC(3)+CC(1,3)*XC(1)*XC(4)+CC(1,4)

1*XC(3)*XC(5)

VC(2)=-VC(1)

VC(3)=VC(1)-92.0

VC(4)=VC(2)*VC(2)*XC(2)*XC(5)+CC(2,3)*XC(1)*XC(2)+CC(2,4)

1*XC(3)*XC(4)

VC(5)=VC(3)-110.0

VC(6)=VC(3)*VC(3)*XC(3)*XC(5)+CC(3,3)*XC(1)*XC(3)+CC(3,4)

1*XC(3)*XC(4)

VC(7)=-VC(5)+20.0

VC(8)=VC(5)-25.0

RETURN

END
SUBROUTINE JACOB

COMMON A(50,100),ALFA(50,50),X(150),XC(150),XI(150),XS(150)

1,Y(150),C(150),VC(50),IBAS(50),IHB(1CC),IVC(50),IVA(100)

COMMON A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHB,IVC,IVA,IVB

COMMON NV,NC,NK,NEG,NIN,NTV,NV1,NEV,NEVL,NTO,NIN1,NIN2,NIN3,NIGEGA

COMMON KFNC,KGRAC,KCONT,KINV1,KINV2,KCDBA,KJACO,KMAX1,KMAX2,KUGE

COMMON KREN1,KREN2,KIPV,KCDBA1,KREN11,KREN21

COMMON IDIREC,DELTFI,ETA,JKO,LC,YSORT

A(1,1)=CC(1,3)*XC(4)
A(1,2)=CC(1,2)*XC(5)
A(1,3)=CC(1,3)*XC(5)
A(1,4)=CC(1,3)*XC(1)
A(1,5)=CC(1,2)*XC(2)*CC(1,4)*XC(3)
A(3,1)=CC(2,3)*XC(2)
A(3,2)=CC(2,2)*XC(5)*CC(2,3)*XC(1)
A(3,3)=2.0*CC(2,4)*XC(3)
A(3,5)=CC(2,2)*XC(2)
A(5,1)=CC(3,3)*XC(3)
A(5,3)=CC(3,2)*XC(5)*CC(3,4)*XC(4)*CC(3,3)*XC(1)
A(5,4)=CC(3,4)*XC(3)
A(5,5)=CC(3,2)*XC(3)

CALL=26,6,2

CALLJ=1,5

A(1,1)=A(1-1,J)

RETURN

END
SUBROUTINE GRADFI

DIMENSION A(50,100), ALFA(50,50), X(150), XC(150), XI(150), XS(150)
1, Y(150), C(15C), VC(5C), IBAS(5C), IHB(1CC), IVC(50), IVA(100)

COMMON A, ALFA, X, XC, XI, XS, Y, C, VC, IBAS, IHB, IVC, IVA, IVB

COMMON NV, NC, NK, NEG, NIN, NTV, NTV, NEV, NEVL, NTO, NIN1, NIN2, NIN3, NIGE

1N4, NVIN1, NVIN2, NVIN3, INDX, II, IR, IRL, IS, IS1, IT, IBP, ICDB, JCDAB, KC

COMMON KFCNC, KGRAC, KCONT, KINV1, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KG

1REN1, KREN2, KINV1, KINV2, KCDBA, KREN11, KREN21

COMMON IDTREC, DELTFI, ETA, JKO, LC, YSORT

COMMON B(5), CC(3, 4)

C(1) = B(2) - B(3) * XC(5)

C(3) = 2.0 * B(4) * XC(3)

C(5) = B(3) * XC(1)

RETURN

END
### GRADIENT RESUIT GENERALISE

<table>
<thead>
<tr>
<th>NOMBRE DE VARIABLES NATURELLES</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOMBRE TOTAL DE VARIABLES</td>
<td>11</td>
</tr>
<tr>
<td>NOMBRE DE CONTRAINTES</td>
<td>6</td>
</tr>
<tr>
<td>EPSILON DE NEWTON</td>
<td>0.10CPE-04</td>
</tr>
<tr>
<td>EPSILON TEST GRADIENT</td>
<td>0.10CPE-02</td>
</tr>
</tbody>
</table>

### FONCTION ECONOMIQUE

0.11718750000000E-01

<table>
<thead>
<tr>
<th>ECRAE</th>
<th>VARIABLE NATURELLE</th>
<th>BURHE SUPERIEURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>X(1)</td>
<td>0.78000000000000E+02</td>
<td>X(1)</td>
</tr>
<tr>
<td>X(2)</td>
<td>0.33000000000000E+02</td>
<td>X(2)</td>
</tr>
<tr>
<td>X(3)</td>
<td>0.27000000000000E+02</td>
<td>X(3)</td>
</tr>
<tr>
<td>X(4)</td>
<td>0.27000000000000E+02</td>
<td>X(4)</td>
</tr>
<tr>
<td>X(5)</td>
<td>0.27000000000000E+02</td>
<td>X(5)</td>
</tr>
</tbody>
</table>

### VALEUR DES CONTRAINTES

| C(1)   | -0.2113647+60D9375E 00 |
| C(2)   | -0.4178863525906E+02  |
| C(3)   | -0.1115728149414E+02  |
| C(4)   | -0.8892718509559E+01  |
| C(5)   | -0.8772711816406E+01  |
| C(6)   | -0.1272801835938E+00  |
DUREE DU CALCUL 165 CENTISECONDES

VARIABLE NATURELLE

\[ X_l(1) = 0.780000000000000000 \] \[ X_l(2) = 0.330000000000000000 \] \[ X_l(3) = 0.29993896424375 \] \[ X_l(4) = 0.456666666666666666 \] \[ X_l(5) = 0.36761230464875 \]

VARIABLE DUALE ASSOCIÉE

\[ V_l(1) = -0.489176025390639999 \] \[ V_l(2) = -0.843040921264418888 \] \[ V_l(3) = 0. \] \[ V_l(4) = 0.266324444626406388 \] \[ V_l(5) = 0. \]

CONTRAINTE

\[ CONTRAINTE 1 = -0.152587890625000000 \] \[ CONTRAINTE 2 = -0.152587890625000000 \] \[ CONTRAINTE 3 = 0.0 \] \[ CONTRAINTE 4 = -0.084175109964658000 \] \[ CONTRAINTE 5 = -0.093986741210999999 \] \[ CONTRAINTE 6 = -0.152587890625000000 \]

VARIABLE DUALE ASSOCIÉE

\[ U_l(1) = 0.403186523437599999 \] \[ U_l(2) = 0.0 \] \[ U_l(3) = 0.0 \] \[ U_l(4) = 0.0 \] \[ U_l(5) = 0.0 \] \[ U_l(6) = 0.093757324188888888 \]
<table>
<thead>
<tr>
<th>NIVEAU DE SORTIE</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOMBRE D ITERATIONS</td>
<td>6</td>
</tr>
<tr>
<td>NOMBRE D APPELS DU S/P CALCULANT LES CONTRAINTE</td>
<td>93</td>
</tr>
<tr>
<td>NOMBRE D APPELS DU S/P CALCULANT LA FONCTION ECONOMIQUE</td>
<td>6</td>
</tr>
<tr>
<td>NOMBRE D APPELS DU S/P CALCULANT LE GRADIENT DE LA FONCTION ECONOMIQUE</td>
<td>14</td>
</tr>
<tr>
<td>NOMBRE D APPELS DU S/P CALCULANT L INVERSE AVEC BASE DONNEE</td>
<td>4</td>
</tr>
<tr>
<td>NOMBRE D APPELS DU S/P CALCULANT L INVERSE AVEC RECHERCHE DE BASE</td>
<td>1</td>
</tr>
<tr>
<td>NOMBRE D APPELS DU S/P EFFECTUANT LES CHANGEMENTS DE BASE</td>
<td>3</td>
</tr>
<tr>
<td>NOMBRE D APPELS DU S/P CALCULANT LE JACOBIEN DES CONTRAINTE</td>
<td>8</td>
</tr>
<tr>
<td>NOMBRE D APPELS DU S/P EFFECTUANT UNE RECHERCHE DE MAX SANS DICHOTOMIE</td>
<td>14</td>
</tr>
<tr>
<td>NOMBRE D APPELS DU S/P EFFECTUANT UNE RECHERCHE DE MAX AVEC DICHOTOMIE</td>
<td>0</td>
</tr>
<tr>
<td>NOMBRE D APPELS DU S/P EFFECTUANT LA RENTREE DANS LE DOMAINE</td>
<td>26</td>
</tr>
<tr>
<td>NOMBRE TOTAL D ITERATIONS</td>
<td>74</td>
</tr>
</tbody>
</table>
SUBROUTINE PHIX
DIMENSION A(5,100), ALFA(50,50), X(150), XC(150), XI(150), XS(150) GEGA 20
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 N, NC, NK, NE, NIV, NT, NV1, NEV, NEV1, NTO, NIN1, NIN2, NIN3, NIGEGA 50
1N4, NVNIN1, NVNIN2, NVNIN3, INDEX, IX, IR, IR1, IS, IS1, IT, IBP, ICDB, JCDB, KCGEGA 60
COMMON VIA, KFIL, KLIN, KREN, KFD, FIL, PHI, PSI, PSI3, TB, TD, TC, EPSIL, EPSIO, EPSIL2GEGA 70
COMMON KFC, KGRAD, KCONT, KINV1, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KGEGA 80
1RENI, KREN2, KINV, KCDBA1, KRENI, KREN21 GEGA 90
COMMON IDIREC, DELTFI, ETA, JKO, LC, YSORT GEGA 100
PHI = 4.0*XC(1) - 10.0*XC(2) - 4.0*XC(3) - 2.0*SORT(XC(1)**2 + XC(2)**2)
RETURN
END

SUBROUTINE CPHI
DIMENSION A(5,100), ALFA(50,50), X(150), XC(150), XI(150), XS(150) GEGA 20
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 N, NC, NK, NE, NIV, NT, NV1, NEV, NEV1, NTO, NIN1, NIN2, NIN3, NIGEGA 50
1N4, NVNIN1, NVNIN2, NVNIN3, INDEX, IX, IR, IR1, IS, IS1, IT, IBP, ICDB, JCDB, KCGEGA 60
COMMON VIA, KFIL, KLIN, KREN, KFD, FIL, PHI, PSI, PSI3, TB, TD, TC, EPSIL, EPSIO, EPSIL2GEGA 70
COMMON KFC, KGRAD, KCONT, KINV1, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KGEGA 80
1RENI, KREN2, KINV, KCDBA1, KRENI, KREN21 GEGA 90
COMMON IDIREC, DELTFI, ETA, JKO, LC, YSORT GEGA 100
VC(1) = XC(1)*XC(2)*XC(3)*100.0
RETURN
END
SUBROUTINE JACOB
DIMENSION A(0, 100), ALFA(50, 50), X(150), XC(150), XI(150), XS(150) GEGA 20
Y(150), C(150), VC(50), IBAS(50), IHB(100), IVC(50), IVA(100)

COMMON A, ALFA, X, XC, XI, XS, Y, VC, IBAS, IHB, IVC, IVA, IVB GEGA 30
COMMON N4, N5, NK, NEG, NIN, NTV, NV1, NEV, NEVL, NTO, NIN2, NIN3, NIGE GEGA 40
1N4, NVNIN1, NVNIN2, NVNIN3, INDEX, II, IR, IR1, IS, IS1, IT, IUP, ICD1, JCD1, KCGE GEGA 50
2CB, KFIL, KLIN, KREN, KD, FII, PHI, PSI, PSI3, TB, TD, TC, EPSIL, EPSIL0, EPSIL2 GEGA 60
COMMON KFCNC, KGRAC, KCONT, KINV1, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KCGE GEGA 70
1REN1, KREN2, KINV, KCDBA1, KREN11, KREN21 GEGA 80
COMMON IDREC, DELTFI, ETA, JK0, LC, YSORT GEGA 90

A(1, 1) = -XC(2) * XC(3)
A(1, 2) = -XC(1) * XC(3)
A(1, 3) = -XC(1) * XC(2)
RETURN
END

SUBROUTINE GRADF1
DIMENSION A(0, 100), ALFA(50, 50), X(150), XC(150), XI(150), XS(150) GEGA 20
Y(150), C(150), VC(50), IBAS(50), IHB(100), IVC(50), IVA(100)

COMMON A, ALFA, X, XC, XI, XS, Y, VC, IBAS, IHB, IVC, IVA, IVB GEGA 30
COMMON N4, N5, NK, NEG, NIN, NTV, NV1, NEV, NEVL, NTO, NIN2, NIN3, NIGE GEGA 40
1N4, NVNIN1, NVNIN2, NVNIN3, INDEX, II, IR, IR1, IS, IS1, IT, IUP, ICD1, JCD1, KCGE GEGA 50
2CB, KFIL, KLIN, KREN, KD, FII, PHI, PSI, PSI3, TB, TD, TC, EPSIL, EPSIL0, EPSIL2 GEGA 60
COMMON KFCNC, KGRAC, KCONT, KINV1, KINV2, KCDBA, KJACO, KMAX1, KMAX2, KCGE GEGA 70
1REN1, KREN2, KINV, KCDBA1, KREN11, KREN21 GEGA 80
COMMON IDREC, DELTFI, ETA, JK0, LC, YSORT GEGA 90

SS = SQRT(XC(1) ** 2 + XC(2) ** 2)
C(1) = -4.0 - 2.0 * XC(1) / SS
C(2) = -10.0 - 2.0 * XC(2) / SS
C(3) = -4.0
RETURN
END
PARAMETRES

NIV 3
MIN 1
NEG 0
NFWL 20
NTO 6
ITET 20
ICONS 1
IODAG 0
ITMAX 50
KFIL 0
KRIA 0
NCD 10
ITSCN 1
ITSLSR 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  5
NOMBRE DE CONTRAINTES  1

EPSILON DE NEWTON  0.1000E-04
EPSILON TEST GRADIENT  0.1000E-02

FONCTION ECONOMIQUE  -0.27828427124023E 03

<table>
<thead>
<tr>
<th>XI</th>
<th>BORNE INFERIEURE</th>
<th>VARIABLE NATURELLE</th>
<th>BORNE SUPERIEURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1)</td>
<td>0.0</td>
<td>0.100000000000000E 02</td>
<td>0.100000000000000E 03</td>
</tr>
<tr>
<td>(2)</td>
<td>0.0</td>
<td>0.100000000000000E 02</td>
<td>0.100000000000000E 03</td>
</tr>
<tr>
<td>(3)</td>
<td>0.0</td>
<td>0.100000000000000E 02</td>
<td>0.100000000000000E 03</td>
</tr>
</tbody>
</table>

VALEUR DES CONTRAINTES

(c 1)  -0.900000000000000E 03
DUREE DU CALCUL 83 CENTISECONDES

VARIABLE NATURELLE

\[ x_1 = 0.30872011144408E+01 \]
\[ x_2 = 0.268178374171E+01 \]
\[ x_3 = 0.7330743786729E+01 \]

VARIABLE DUALE ASSOCIEE

\[ v_1 = -0.66871643066406E+02 \]
\[ v_2 = 0.0 \]
\[ v_3 = -0.1033782968946E+02 \]

CONTRAINTE

CONTRAINTE \(1\) \(G_0\)

\[ u_1 = 0.29315400123596E+00 \]

NIVEAU DE SURTIE \(1\)

ACHERE D'ITERATIONS \(9\)

NOMBRE D'APPELS DU S/P CALCULANT LES CONTRAINTE \(167\)

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 \(12\)

NOMBRE D'APPELS DU S/P EFFECTUANT LA JACOBIEN DES CONTRAINTE \(12\)

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 \(31\)

NUMBR: 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 °F
AK(I)  temperature dissipation rate coefficient in (days)$^{-1}$
AK1(I) deoxygenation rate coefficient in (days)$^{-1}$ at 20 °C
AK2(I) reaeration rate coefficient in (days)$^{-1}$ at 20 °C
ST(I)  incoming water temperature in °F
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 °F
TMAX(I) maximum allowable temperature in °F
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 $I = 1, \ldots, NSTAGE$. 
SUBROUTINE PH1
DIMENSION A(50,100), ALFA(50,50), X(15C), XC(150), XI(150), XS(150), GEGA 20
1, Y(15C), CL(150), VC(50), IBAS(50), IFB(1LC), IVC(50), IVA(100) GEGA 30
COMMON A, ALFA, X, XI, XS, Y, C, VC, IBAS, IFB, IVC, IVA, IVB GEGA 40
COMMON NV, NC, NK, NEQ, AN, NTV, NV1, NEV, NEVL, NTO, NII1, NII2, NII3, NI 50
1, N4, NNVN1, NNVN2, NNVN3, INDEX, II, IR, IR, IS, I, IT, IBP, ICDB, JCDB, GCCGA 60
2DP, KFIL, KLIN, KREX, KD, FI1, PHI, PSI, PSI3, TB, TC, TC, EPSIL, EPS10, EPS126 GEGA 70
COMMON KFCNC, KGPRC, KCONS, KINV1, KINV2, KCCBS, KJACO, KMA1, KMA2, KGEGA 80
1, KREN1, KREN2, KINV, KCCBA1, KREN11, KREN21 GEGA 90
COMMON ICTREC, DELTFI, ETA, JKO, LC, YSCRT GEGA 100
COMMON AREA1/X(4), XQ(4), XB(4), XE(4), AK(4), AK1(4), AK2(4), ST(4) GEGA 100
1, SB(4), SD(4), TF(4), THS(4), TMAX(4), PDC(4), NSTAGE 2, CC1, CC2, DELTH
REAL MOD
IF (IT.LT.0) CALL INPUT
PHI=0.0
PHI=PHI*(0.8)*(1.0-EXP(-0.6*XC(2)))
PHI=PHI*(0.675)*(1.0-EXP(-0.6*XC(4)))
IF (XC(5).LT.20.*X) GC TC 10
IF (XC(5).LT.36.*X) GC TC 11
IF (XC(5).LT.46.*X) GC TC 12
IF (XC(5).LT.76.*X) GC TC 13
IF (XC(5).LT.56.*X) GC TC 14
IF (XC(5).LT.96.*X) GC TC 15
PHI=PHI+90.*YC(5)-91.2
GC TC 30
10 PHI=PHI+0.04*YC(5)
GC TC 30
11 PHI=PHI+0.01*YC(5)+0.2
GO TO 30
12 PHI=PHI+0.01*YC(5)+0.8
GO TC 30
13 PHI=PHI+0.01*3.0*XC(5)-10.43+1.2
GO TC 30
14 PHI=PHI+0.015*XC(5)+0.25
GO TC 30
15 PHI=PHI+0.06*XC(5)-3.6
GC TC 30
30 PHI=-PHI
RETURN
END
SUBROUTINE GRDFI
DIMENSION A(FG,160),ALFA(5C,5C),X(15C),XC(150),XI(150),XS(150)
GEGA 20
1,Y(150),C(150),VC(50),IBAS(50),IBB(1C),IVC(50),IVA(1C0)
GEGA 30
COMMON A,LFA,X,XC,XI,XS,Y,C,VC,IBAS,IBB,IVC,IVA,IVB
GEGA 40
COMMON NV,NK,NEC,KIN,NV1,NV1,KNE,KNEW,KTO,K0,NIN1,NIN2,NIN3,
GEGA 50
NIN4,NVIN1,NVIN2,NVIN3,INDEX,II,IR,IR1,IS,IS1,IT,IBB,ICDD,JCDD,KGEG
GEGA 60
COMMON KB1,KF1,FIL,FIN,FIN,KE,FI,F1,PHI,PS1,PSI3,T0,T1,TC,EPS1,EPSI,
GEGA 70
EPSIL2,EPSIL2,GEGA 80
COMMON KGEG1,KGRAC,KCON1,KVIN1,KINV2,KCDBA,KJACC,KNP1,KNP2,KPGE
GEGA 90
COMMON ICIFEC,DELTFI,ETA,KO,LC,YSORT
GEGA 100
COMMON AREA1/X(4),XQ(4),XH(4),XE(4),AK(4),AK1(4),AK2(4),ST(4)
GEGA 110
COMMON SM1(4),SD(4),TF(4),TS(4),TMAX(4),MDC(4),NSTAGE
GEGA 120
2,CC1,CC2,DELT,
REAL MOD
C(2)=-0.817/E-9)*EXP(-0.023*XC(2))*C*C23
C(4)=-0.575/E-9)*EXP(-0.023*XC(4))*C*C23
IF (XC(5)>LT.20.E0) GO TC 10
IF (XC(5)>LT.3.E0) GO TC 11
IF (XC(5)>LT.4.E0) GO TC 12
IF (XC(5)>LT.7.E0) GO TC 13
IF (XC(5)>LT.5.E0) GO TC 14
IF (XC(5)>LT.9.E0) GO TC 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)=-0.013
RETURN
14 C(5)=-0.015
RETURN
15 C(5)=-0.06
RETURN
END
SUBROUTINE GPHI
DIMENSION A(100), ALFA(50, 50), X(150), XI(150), X(150), Y(150) GEGA 20
1, Y(150), C(150), VC(50), IBAS(50), JHD(100), IVC(50), IVA(100) GEGA 30
COMMON A, ALFA, Y, X, X, XI, X, Y, C, VC, IBAS, JHD, IVC, IVA, IVB GEGA 40
COMMON NV, NC, NK, NNE, KIN, NTV, NV1, NEV, NVE, NTE, NTC, NIN1, NIN2, NIN3, NIGEGA 50
N4, NV4, NV5, NV5, NV5, NV5, NVE, NVE, NTE, NTE, NTC, NIN1, NIN2, NIN3, NIGEGA 60
Z1, KPL, KIN, KREA, KRE, KRE, KRE, KRE, KRE, KRE GEGA 70
COMMON KFNC, KGRA, KCONT, KINV1, KINV2, KCCBA, KJACG, KM1, KM2, KGEA 80
IR, IR4, IR4, IR4, IR4, IR4, IR4, IR4, IR4, ST(4) GEGA 90
COMMONG, JTC, DELTA, ETA, JIK, L, MT, Y, SCR GEGA 100
COMMONG, AR, A1, Y(4), XO(4), XB(4), XE(4), AK(4), AK(4), ST(4), 1, S4(4), SD(4), F(4), TRS(4), TMAX(4), MCC(4), NSTAGE GEGA 100
2, CCI, CCI, DELTA GEGA 100
REAL MDO
C*******INITIALIZE
IF(IT+5<3) G0 TO 100
C051=1, NSTAG=1
C052=1, NSTAG=1
X<1>=(CC2*X1(T))/1(CCC0*XC1(I))
X<1>=(CC1*X1(T))/1(CCC0*XC1(I))
5 CONTINUE
C******END OF INITIALIZATION
C******SET UP BOUNDARY CONDITIONS
10C CALL CHANGE(1, NSTAGE-1)
C******END OF BOUNDARY CONDITIONS
C******IN=QUALITY CE*STRANTS
C******I SUBSCRIPTS...MAX TEPF RISE
C******NSTAGE+I SUBSCRIPTS...MAX TEMP
C******2NSTAGE+I SUBSCRIPTS...MIN CC
C0<1>=1, NSTAG=1
VC(1)=X(1)*(1.00-0.05X(1))=X(1)
VC(NSTAGE+1)=(T(1)+X(1)*(1.00-0.05X(1)=TMAX1)
CALL MINDO, MINDO, TMIN, FMIN
VC(2NSTAGE+1)=-FMIN+MINDO(1)
10 CONTINUE
RETURN
END
SUBROUTINE JACOB
DIMENSION A(0,150),ALFA(50,50),X(150),XC(150),XI(150),XS(150) GEGA 20
1,Y(150),C(150),VC(150),IBAS(50),IHC(150),IVC(50),IVA(100) GEGA 30
COMMON A,ALFA,X,XC,XI,XS,Y,C,VC,IBAS,IHC,IVC,IVA,IVB GEGA 40
COMMON N,NC,K,KNEG,NIN,KTV,NVL,NEV,NEVL,NTO,NIN2,NIN3,NICE GEGA 50
1,NV,NVN1,NVNL2,NVNL3,INDEX,I1,J1,IS,IS1,IT,IBP,ICDB,JCDB,KGE GEGA 60
2,PK,KL,YN,KTV,KC,FI,F1,P1,P1,PS13,PS1,PS1,TB,TC,TC,EP1,EP1,EP1 GEGA 70
COMMON KFNC,KGRC,KGCNT,KTV1,KTV2,KDBA,KJD,BJAC,KB1,KB2,KFNC GEGA 80
1,KE1,KRE2,KP,KP,KDBA1,KRE11,KR,K21 GEGA 90
COMMON IT,IC1,IC2,ICT,IC,ETA,JK,LC,Y,SCRT GEGA 100
COMMON IARC,XT4,XT4,XT4,XT4,XT4,XT4,XT4,XT4,XT4,XT4,XT4,XT4 GEGA 100
1,ST4,ST4,ST4,ST4,ST4,ST4,ST4,ST4,ST4,ST4,ST4,ST4,ST4,ST4,ST4,ST4,ST4
2,C1,CC2,DELTA
REAL MOD
C?????INITIALIZE CONSTANTS
IF(I1+G.E.0) GO TO 100
CD6M=1,NSTAGE
A=0.0
K=1
50 A(NSTAGE*K,J)=XT(K)*EXP(A*X)
K=K+1
IF(K.GE.30) GO TO 55
A=AXX+AK(V)*TF(K)
GO TO 50
55 A(I,J)=XT(I)
60 CONTINUE
C?????END OF INITIALIZATION
1CC CALL CHANGE(1,NSTAGE-1)
C?????NUMERICAL PARTIAL DERIVATIVES FOR THE MIN CON CONSTRAINT
D30I=1,NSTAGE
CALL MINDO(I,TMIN,FMIN)
IF(XT(I).EQ.0.0) GO TO 5
XCC=X(I)
X(I)=X(I)+EPLTA
CALL MINDO(I,TMIN,FMIN)
A(2*NSTAGE*I,J)=(FMIN-FMIN)/CEPLTA
X(I)=XCC
5 IF(XB(I).EQ.0.0) GO TO 6
XCC=X(NSTAGE+I)
X(NSTAGE+I)=Y(NSTAGE+I)+CEPLTA
CALL MINDO(I,TMIN,FMIN)
A(2*NSTAGE*I,NSTAGE+I)=(FMIN-FMIN)/CEPLTA
X(NSTAGE+I)=XCC
6 IF(I.GT.0) GO TO 30
I1=1
D20J=1,III
IF(XT(I).EQ.0.0) GO TO 7
XCC=X(I)
X(I)=X(I)+EPLTA
CALL CHANGE(I,III)
CALL MINDO(I,TMIN,FMIN)
A(2*NSTAGE*I,I)=FMIN/FMIN/CEPLTA
X(I)=XCC
7 IF(XB(I).EQ.0.0) GO TO 8
XCC=X(NSTAGE+I)
X(NSTAGE+I)=Y(NSTAGE+I)+CEPLTA
CALL CHANGE(I,III)
CALL MINDO(I,TMIN,FMIN)
\[ A(2 \times \text{NSTAGE} + 1, \text{NSTAGE} + J) = (FMIN - FMIN1) / \text{DELTAR} \]
\[ XG(NSTAGE + J) = \text{XGII} \]

IF (XJ(J) EQ 0.0) AND (XJ(J) EQ 0.0) GC TC 20
CALL CHANGE(J, III)

20 CONTINUE
30 CONTINUE
RETURN
END
SUBROUTINE CHANGE(J,N)
DIMENSION A(50),ALFA(50),X(50),XC(50),XI(50),XS(50) GEGA 20
1,Y(50),C(50),VC(50),IRAS(50),INB(100),IVC(50),IVA(100) GEGA 30
COMMON A,ALFA,X,XC,XI,XS,Y,C,VC,IRAS,INB,IVC,IVA,IWB,IV6 GEGA 40
COMMON NV,NC,NK,NEG,NIN,NTV,NV1,KREV,KEV,NEV,NETO,NIN1,NIN2,NIN3,NIGEGA 50
1N4,NVKN1,NVKN2,NVKN3,INCEX,II,IR,IR1,IS,ISL,IT,ICD,JCDB,KGEGA 60
2DB,KFIL,KLIN,KREN,KC,FI1,FI2,FI3,PS1,PS2,PS3,PSI3,TB,TC,TC,TC,EPSIL,EPSIL0,EPSIL2GE 70
COMMON KFCN,KGRAC,KCONT,KIN1,KIN2,KIN2,KCCBA,KJACC,TKMAX1,TKMAX2,KGEA 80
1KREN1,KREN2,KIN1,KCV,KGDB1,KKREN1,KKREN2 GEGA 90
COMMCN ICIEC,DELFI,ETA,JKG,LO,YSORT GEGA 100
COMMON AR,A1,X(4),XG(4),XH(4),X(l)(4),AK(4),AK1(4),AK2(4),ST(4)
1,SB(4),SN(4),TF(4),TR5(4),TMAX(4),MPD(4),NSTAGE
2,C1,C2,DELFI GEGA 100
REAL M

C*****FUNCTION DEFINITIONS
TEMPC(T)=(5.0/9.0)*(T-32.0)
CS(T)=1.652-0.41032*TEMPC(T)+0.79916-2*TEMPC(T)^2+0.77745-4
1+TEMPC(T)^3
C*****END OF FUNCTION DEFINITIONS
C*****SET UP ROUNDS FOR STAGES J+1 TO N+1
CO201=J+N
TMIX=ST(I)+XT(I)*(100-C-XC(I))
ST(I+1)=X(1)*TMIX-XE(I)*EXP(-AK(I)*TF(I))
BO=SB(I)+XE(I)*(100-C-XC(I)(NSTAGE+1))
AK1=AK(I)*(1.047)**(TEMPC(ST(I+1))-2*C+0)
SB(I+1)=BC*EXP(-AK1*TF(I))
AK2=AK2(I)*(1.024)**(TEMPC(ST(I+1))-2*C+0)
CODEF=CS(TMIX)-SCS(I)
IF(CODEF<0) CODEF=0.0
SO(T+1)=CS(S(I+1))-(AK1*(BCD/(AAK2-AAK1)))*(EXP(-AK1*TF(I))
1+EXP(-AAK2*TF(I)))-CODEF*EXP(-AAK2*TF(I))
20 CONTINUE
RETURN
END

SUBROUTINE MINCC(I, TMIN, FMIN)

DIMENSION A(150, 150), ALFA(50, 50), X(150), XC(150), XI(150), XS(150)

COMMON

COMMON
NAM, NAF, NMF, NMG, NMB, NMA, NMC, NMD, NME, NMF, NMG, NMB, NMA, NMC, NMD, NME

COMMON
V, VF, VV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV, BV,BV
FUNCTION FUNCT(I, TTT)

DIMENSION A(300, 300), ALFA(50, 50), X(150), XC(150), XI(150), XS(150) GEGA 20
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
COMMON NV, NC, NK, NGS, NIN, NTV, NV1, NV2, NEWL, NT1, NIN1, NIN2, NIN3, NIGEC, 30
N4, NV2IN1, NBN21, NBN12, NBN1F3, INGBX, II, IR, IR1, SI, IS1, IT, IOP, ICD, ICPB, KCGA GEGA 50
ZDB, KFIL, KLIN, KR2N, KC, FI1, PHI, PSI1, PSI2, TP, TC, TC, EPSIL, EPS10, EPS1L2, EPS1, 70
COMMON KFNC, KGRAC, KCONT, KINV1, KINV2, KCSBA, KCACC, KMAX1, KMAX2, KGEA 80
KREN1, KREN2, KI'V, KCSBA, KREN1, KREN2 GEGA 90
COMMON ICTRC, DELT1, ETA, JK9, LC, YSCRT
COMMON ARE=I/XT(4), XQ(4), XB(4), XE(4), AK(4), AK1(4), AK2(4), ST(4)
1, SR(4), SQ(4), TF(4), TRS(4), TMAX(4), MOD(4), NSTAGE
2, CI1, CC2, DELTA
REAL MDO

C*****FUNCTION DEFINITIONS
TEMPC(T) = (5.0/9.0)*(T-32.0)
CS(T) = 14.652 - 4.1022*TEMPC(T) + 0.79910E-2*TEMPC(T)**2 - 0.77774E-4
1*TEMPC(T)**3

C*****END OF FUNCTION DEFINITIONS
TMIX = ST(I) + XI(I)*(100.0 - XC(I))
STII = XI(I) + TMIX*XI(I)*EXP(-AK(I)*TTT)
BED = SB(I) + XB(I)*(100.0 - XC(NSTAGE + I))
AAK1 = AK1(I) + (1.0/47.0)**(TEMPC(STII) - 20.0)
AAK2 = AK2(I) + (1.0/47.0)**(TEMPC(STII) - 20.0)
"CDEF="CS(TMIX) - SO(I)
IF(DODEF.LT.0.0) UDDEF = 0.0
FUNCTION = (AAK1*BED/(AAK2-AAK1))*(EXP(-AAK1*TTT)
1-EXP(-AAK2*TTT)) - DODEF*EXP(-AAK2*TTT)
RETURN
END
SUBROUTINE INCLT
COMMON/AREA/XT(4),XQ(4),XB(4),XE(4),AK(4),AK1(4),AK2(4),ST(4)
1,STR(4),SO(4),TF(4),TRS(4),TMAX(4),MODD(4),NSTAGE
2,CC1,CC2,DELTA
COMMON/AREA2/TIMES(4),DEL(4),NF(30)
REAL MODD
C***** NUMBER OF STAGES
NSTAGE=4
READ(5,1000) YT,XQ,XB,XE,AK,AK1,AK2,ST,SB,SC,TF,TRS,TMAX,MODD
READ(5,2000) DELTA
C***** CONVERSION FACTORS
CC1=1.0/(3.067*62.4262)
CC2=0.453592476/(8.64E4*28031605)
C***** DETERMINE NUMBER OF ITERATIONS FOR FIBONACCI SEARCH
NF(1)=1
NF(2)=1
CO701=1,NSTAGE
CO50J=300
N7M3(J)=J-1
NF(J)=NF(J-1)+NF(J-2)
NF1=NF(J-1)
NF=NF(J)
IF(NF*TF(1)) LE 0.0001) GC TC 70
5C CONTINUE
7C CEL(I)=AF1*AF
WRITE(6,3000) XT,XQ,XB,XE,AK,AK1,AK2,ST,SB,SO,TF,TRS,TMAX,MODD
WRITE(6,4000) NSTAGE,DELTA,CC1,CC2
WRITE(6,5000) NTIMES,CEL
RETURN
10CC FORMAT(4G10.6)
20CC FORMAT(F10.0)
30CC FORMAT(4(5X,213.6))
40CC FORMAT(5X,15,?/(5X,E13.6))
50CC FORMAT(4(5X,1F)/4(5X,E13.6))
END
### PARAMETRES

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NV</td>
<td>6</td>
</tr>
<tr>
<td>NIN</td>
<td>12</td>
</tr>
<tr>
<td>NEG</td>
<td>0</td>
</tr>
<tr>
<td>NEWL</td>
<td>20</td>
</tr>
<tr>
<td>NT0</td>
<td>6</td>
</tr>
<tr>
<td>ITET</td>
<td>20</td>
</tr>
<tr>
<td>IONAJ</td>
<td>1</td>
</tr>
<tr>
<td>IOUAS</td>
<td>0</td>
</tr>
<tr>
<td>IMIX</td>
<td>50</td>
</tr>
<tr>
<td>KFIL</td>
<td>0</td>
</tr>
<tr>
<td>KLRA</td>
<td>0</td>
</tr>
<tr>
<td>NCO</td>
<td>20</td>
</tr>
<tr>
<td>ITSCR</td>
<td>1</td>
</tr>
<tr>
<td>ISOLSRC</td>
<td>50</td>
</tr>
</tbody>
</table>

**EPILE**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPSIL1</td>
<td>0.1E-02</td>
</tr>
<tr>
<td>EPSIL0</td>
<td>0.1E-02</td>
</tr>
<tr>
<td>EPSIL2</td>
<td>0.1E-02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>0.0</td>
</tr>
<tr>
<td>EPSIL1</td>
<td>0.24E03</td>
</tr>
<tr>
<td>EPSIL0</td>
<td>0.87E03</td>
</tr>
<tr>
<td>EPSIL2</td>
<td>0.85E00</td>
</tr>
<tr>
<td>EPSIL1</td>
<td>0.4C</td>
</tr>
<tr>
<td>EPSIL0</td>
<td>0.92E03</td>
</tr>
<tr>
<td>EPSIL2</td>
<td>0.92E03</td>
</tr>
<tr>
<td>EPSIL1</td>
<td>0.80E00</td>
</tr>
<tr>
<td>EPSIL0</td>
<td>0.10E01</td>
</tr>
<tr>
<td>EPSIL2</td>
<td>0.90E00</td>
</tr>
<tr>
<td>EPSIL1</td>
<td>0.80E00</td>
</tr>
<tr>
<td>EPSIL0</td>
<td>0.10E01</td>
</tr>
<tr>
<td>EPSIL2</td>
<td>0.90E00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPSIL1</td>
<td>0.1E00</td>
</tr>
<tr>
<td>EPSIL0</td>
<td>0.1E00</td>
</tr>
<tr>
<td>EPSIL2</td>
<td>0.1E00</td>
</tr>
</tbody>
</table>

### Notes

- **EPSIL** represents various error tolerances.
- The values given are in scientific notation.
- The table provides a list of parameters and their corresponding values.
GRADIENT REDUIT GENERALISE

| NOMBRE DE VARIABLES NATURELLES | 8          |
| NOMBRE TOTAL DE VARIABLES     | 24         |
| NOMBRE DE CONTRAINTES         | 12         |
| EPSILON DE NEWTON             | 0.100E-02  |
| EPSILON TEST GRADIENT         | 0.100E-02  |

FONCTION ECONOMIQUE 0 <= C

<table>
<thead>
<tr>
<th>Borne inférieure</th>
<th>Variable naturelle</th>
<th>Borne supérieure</th>
</tr>
</thead>
<tbody>
<tr>
<td>X( 1) 0.0</td>
<td>X( 1) 0.0</td>
<td>X( 1) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X( 2) 0.0</td>
<td>X( 2) 0.0</td>
<td>X( 2) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X( 3) 0.0</td>
<td>X( 3) 0.0</td>
<td>X( 3) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X( 4) 0.0</td>
<td>X( 4) 0.0</td>
<td>X( 4) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X( 5) 0.0</td>
<td>X( 5) 0.0</td>
<td>X( 5) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X( 6) 0.0</td>
<td>X( 6) 0.0</td>
<td>X( 6) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X( 7) 0.0</td>
<td>X( 7) 0.0</td>
<td>X( 7) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X( 8) 0.0</td>
<td>X( 8) 0.0</td>
<td>X( 8) 0.10000000000000E 03</td>
</tr>
</tbody>
</table>

VALEUR DES CONTRAINTES

<table>
<thead>
<tr>
<th>i</th>
<th>C(i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-C.10000000000000E 02</td>
</tr>
<tr>
<td>2</td>
<td>-C.22750283287598E 01</td>
</tr>
<tr>
<td>3</td>
<td>-C.10000000000000E 02</td>
</tr>
<tr>
<td>4</td>
<td>-C.22613993154797E 01</td>
</tr>
<tr>
<td>5</td>
<td>-C.12300003981758E 02</td>
</tr>
<tr>
<td>6</td>
<td>-C.24576376957133E-01</td>
</tr>
<tr>
<td>7</td>
<td>-C.11200057982939E 02</td>
</tr>
<tr>
<td>8</td>
<td>-C.36490631103516E 01</td>
</tr>
<tr>
<td>9</td>
<td>-C.20126660265503E 01</td>
</tr>
<tr>
<td>10</td>
<td>-C.71467209862030E 01</td>
</tr>
<tr>
<td>11</td>
<td>-C.30198316574097E 01</td>
</tr>
<tr>
<td>12</td>
<td>-C.22101259231567E 01</td>
</tr>
</tbody>
</table>
IT 1 PHI-0.775471E757CCGE 04 DIR-GRAC NC 1C YN 0.34E 03 DELTAFI 0.37E 05 ETA 0.33E 05 NCDB 0 NCN 2 NITN 2
IT 2 PHI-0.58994921075CCOE 04 DIR-CONJ LC 2 YN 0.29E 03 DELTAFI 0.26E 05 ETA 0.16E 05 NCDB 0 NCN 2 NITN 2
IT 3 PHI-0.46764522875CCOE 04 DIR-CONJ LC 2 YN 0.27E 04 DELTAFI 0.26E 05 ETA 0.27E 14 NCDB 0 NCN 2 NITN 2
IT 4 PHI-0.43253955375CCGE 04 DIR-GRAC NC 1C YN 0.29E 03 DELTAFI 0.26E 05 ETA 0.16E 05 NCDB 1 NCN 2 NITN 5
IT 5 PHI-0.294.1310566406E 04 DIR-GRAC NC 11 YN 0.16E 04 DELTAFI 0.26E 05 ETA 0.16E 14 NCDB 1 NCN 4 NITN 25
IT 6 PHI-0.9695307.34375CCOE 03 DIR-GRAC NO 11 YN 0.16E 04 DELTAFI 0.26E 05 ETA 0.16E 14 NCDB 1 NCN 3 NITN 18
IT 7 PHI-0.16048212651392E 01 DIR-GRAC 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 ANNULÉES
PHI-G-16048212651392E 01 YN 0.0 DELTAFI 0.0 ETA 0.0

DURÉE DU CALCUL 3035 CENTISECONDES
### VARIABLE NATURELLE

<table>
<thead>
<tr>
<th>$x(1)$</th>
<th>$0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x(2)$</td>
<td>$1.85419311523445E+02$</td>
</tr>
<tr>
<td>$x(3)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$x(4)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$x(5)$</td>
<td>$6.695965576172E+02$</td>
</tr>
<tr>
<td>$x(6)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$x(7)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$x(8)$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

### VARIABLE CUALE ASSOCIEE

<table>
<thead>
<tr>
<th>$v(1)$</th>
<th>$0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v(2)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$v(3)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$v(4)$</td>
<td>$-1.469441169856E+01$</td>
</tr>
<tr>
<td>$v(5)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$v(6)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$v(7)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$v(8)$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

### CONTRAINE

| CONTRAINE 1 | $-1.00000000000000E+02$ |
| CONTRAINE 2 | $-3.9945068359375E+03$ |
| CONTRAINE 3 | $-1.00000000000000E+02$ |
| CONTRAINE 4 | $-2.2613583154297E+01$ |
| CONTRAINE 5 | $-1.236050515768E+02$ |
| CONTRAINE 6 | $-2.301610131839E+01$ |
| CONTRAINE 7 | $-1.1424006958680E+02$ |
| CONTRAINE 8 | $-3.8182373646675E+01$ |
| CONTRAINE 9 | $-3.0946960449219E+01$ |
| CONTRAINE 10 | $-9.9563398632813E+03$ |
| CONTRAINE 11 | $-1.1565604873678E+01$ |
| CONTRAINE 12 | $-1.4793292541594E+01$ |

### VARIABLE CUALE ASSOCIEE

<table>
<thead>
<tr>
<th>$u(1)$</th>
<th>$0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u(2)$</td>
<td>$0.1113684642635E+00$</td>
</tr>
<tr>
<td>$u(3)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$u(4)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$u(5)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$u(6)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$u(7)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$u(8)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$u(9)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$u(10)$</td>
<td>$0.34952521324158E+01$</td>
</tr>
<tr>
<td>$u(11)$</td>
<td>$0$</td>
</tr>
<tr>
<td>$u(12)$</td>
<td>$0$</td>
</tr>
</tbody>
</table>
NIVEAU DE SORTIE 1

NOMBRE D'ITERATIONS 7

NOMBRE D'APPELS CL S/P CALCULANT LES CONTRAINTES 75
NOMBRE D'APPELS CL S/P CALCULANT LA FONCTION ECONOMIQUE 39
NOMBRE D'APPELS CL S/P CALCULANT LE GRADIENT DE LA FONCTION ECONOMIQUE 15
NOMBRE D'APPELS CL S/P CALCULANT L'INVERSE AVEC BASE DONNÉE 4
NOMBRE D'APPELS CL S/P CALCULANT L'INVERSE AVEC RECHERCHE DE BASE 0
NOMBRE D'APPELS CL S/P EFFECTUANT LES CHANGEMENTS DE BASE 4
NOMBRE D'APPELS CL S/P CALCULANT LE JACOBIEN DES CONTRAINTES 9
NOMBRE D'APPELS CL S/P EFFECTUANT UNE RECHERCHE DE MAX SANS DICHOTOMIE 7
NOMBRE D'APPELS CL S/P EFFECTUANT UNE RECHERCHE DE MAX AVEC DICHOTOMIE 0
NOMBRE D'APPELS CL S/P EFFECTUANT LA RENTREE DANS LE DOMAINE 17

NOMBRE TOTAL D'ITERATIONS 60
<table>
<thead>
<tr>
<th>PARAMETRES</th>
</tr>
</thead>
<tbody>
<tr>
<td>MW</td>
</tr>
<tr>
<td>NIN</td>
</tr>
<tr>
<td>NCG</td>
</tr>
<tr>
<td>NEVL</td>
</tr>
<tr>
<td>NTO</td>
</tr>
<tr>
<td>ITET</td>
</tr>
<tr>
<td>ICNQ</td>
</tr>
<tr>
<td>IDs</td>
</tr>
<tr>
<td>ITMAX</td>
</tr>
<tr>
<td>KFIL</td>
</tr>
<tr>
<td>KLIN</td>
</tr>
<tr>
<td>MO</td>
</tr>
<tr>
<td>MME</td>
</tr>
<tr>
<td>MME</td>
</tr>
<tr>
<td>EPSIL</td>
</tr>
<tr>
<td>EPSIL0</td>
</tr>
<tr>
<td>EPSIL1</td>
</tr>
<tr>
<td>EPSIL2</td>
</tr>
</tbody>
</table>

\[
\begin{array}{cccc}
0.0 & 0.243000E+10 & 0.6 & 0.160000E+00 \\
0.820000E+03 & 0.270000E+03 & 0.385000E+01 & 0.200000E+01 \\
0.110000E+06 & 0.0 & 0.0 & 0.0 \\
0.657000E+02 & 0.807000E+02 & 0.807000E+02 & 0.807000E+02 \\
0.124000E+01 & 0.124000E+01 & 0.124000E+01 & 0.124000E+01 \\
0.100000E+01 & 0.100000E+01 & 0.100000E+01 & 0.100000E+01 \\
0.500000E+02 & 0.900000E+02 & 0.900000E+02 & 0.900000E+02 \\
0.807000E+02 & 0.0 & 0.0 & 0.0 \\
0.100000E+01 & 0.0 & 0.0 & 0.0 \\
0.670000E+01 & 0.0 & 0.0 & 0.0 \\
0.500000E+01 & 0.187000E+01 & 0.170000E+01 & 0.240000E+01 \\
0.100000E+02 & 0.100000E+02 & 0.100000E+02 & 0.100000E+02 \\
0.300000E+02 & 0.930000E+02 & 0.930000E+02 & 0.930000E+02 \\
0.300000E+01 & 0.300000E+01 & 0.300000E+01 & 0.300000E+01 \\
\end{array}
\]

\[
\begin{array}{cccc}
4 & 0.100000E+01 & 0.444966E+00 & 0.185460E+00 \\
14 & 22 & 17 & 22 \\
0.618000E+00 & 0.618000E+00 & 0.618000E+00 & 0.618000E+00 \\
\end{array}
\]

261
GRANCIENT REDUIT GENERALISE

NOMBRE DE VARIABLES NATURELLES : 8
NOMBRE TOTAL DE VARIABLE : 20
NOMBRE DE CONTRAINTES : 12

EPSILON DE NEHTON : 0.100E-02
EPSILON TEST GRADIENT : 0.100E-02

FONCTION ECONOMIQUE : -C.87915740966797E 01

<table>
<thead>
<tr>
<th>BORNE INFERIEUR</th>
<th>VARIABLE NATURELLE</th>
<th>BORNE SUPERIEURE</th>
</tr>
</thead>
<tbody>
<tr>
<td>X(1) 0.0</td>
<td>X(1) 0.0</td>
<td>X(1) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X(2) 0.0</td>
<td>X(2) 0.10000000000000E 03</td>
<td>X(2) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X(3) 0.0</td>
<td>X(3) 0.0</td>
<td>X(3) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X(4) 0.0</td>
<td>X(4) 0.10000000000000E 03</td>
<td>X(4) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X(5) 0.0</td>
<td>X(5) 0.10000000000000E 03</td>
<td>X(5) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X(6) 0.0</td>
<td>X(6) 0.0</td>
<td>X(6) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X(7) 0.0</td>
<td>X(7) 0.0</td>
<td>X(7) 0.10000000000000E 03</td>
</tr>
<tr>
<td>X(8) 0.0</td>
<td>X(8) 0.0</td>
<td>X(8) 0.10000000000000E 03</td>
</tr>
</tbody>
</table>

VALEUR DES CONTRAINTES

C(1) -C.10000000000000E 02
C(2) -C.10000000000000E 02
C(3) -C.10000000000000E 02
C(4) -C.10000000000000E 02
C(5) -C.12300000000000E 02
C(6) -C.12300000000000E 02
C(7) -C.12300000000000E 02
C(8) -C.12300000000000E 02
C(9) -C.12300000000000E 02
C(10) -C.12300000000000E 02
C(11) -C.34234234234234E 01
C(12) -C.34234234234234E 01
<table>
<thead>
<tr>
<th>IT</th>
<th>PHI</th>
<th>PHI-C</th>
<th>DIR-CONJ</th>
<th>NC</th>
<th>YN</th>
<th>DELTA F</th>
<th>ETA</th>
<th>NCDB</th>
<th>NCE</th>
<th>NT</th>
<th>NTN</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.298460569741858 01</td>
<td>0.298460569741858 01</td>
<td>DIR-CONJ</td>
<td>16</td>
<td>5</td>
<td>9 E-02</td>
<td>9 E-02</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.26766376495361E 01</td>
<td>0.26766376495361E 01</td>
<td>DIR-CONJ</td>
<td>16</td>
<td>5</td>
<td>9 E-02</td>
<td>9 E-02</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.267471516779287 01</td>
<td>0.267471516779287 01</td>
<td>DIR-CONJ</td>
<td>16</td>
<td>5</td>
<td>9 E-02</td>
<td>9 E-02</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.22851672975638 01</td>
<td>0.22851672975638 01</td>
<td>DIR-CONJ</td>
<td>16</td>
<td>5</td>
<td>9 E-02</td>
<td>9 E-02</td>
<td>0</td>
<td>10</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.1604867853687E 01</td>
<td>0.1604867853687E 01</td>
<td>DIR-GRAD</td>
<td>36</td>
<td>5</td>
<td>11 E-01</td>
<td>55 E-00</td>
<td>0</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

CLREE CALCL 3541 CENTISECONDES
### VARIABLE NATURELLE

<table>
<thead>
<tr>
<th>( x_i )</th>
<th>( x_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>2</td>
<td>( 0.18541931152344E+02 )</td>
</tr>
<tr>
<td>3</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>4</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>5</td>
<td>( 0.66891937258599E+02 )</td>
</tr>
<tr>
<td>6</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>7</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>8</td>
<td>( 0.0 )</td>
</tr>
</tbody>
</table>

### VARIABLE CUALE ASSOCIÉE

<table>
<thead>
<tr>
<th>( v_i )</th>
<th>( v_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>2</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>3</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>4</td>
<td>( -0.14694441109896E-01 )</td>
</tr>
<tr>
<td>5</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>6</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>7</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>8</td>
<td>( 0.0 )</td>
</tr>
</tbody>
</table>

### CONTRAINTE

<table>
<thead>
<tr>
<th>CONTRAINE</th>
<th>( C_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( -0.10000000000000E+02 )</td>
</tr>
<tr>
<td>2</td>
<td>( -0.9990568359375E-03 )</td>
</tr>
<tr>
<td>3</td>
<td>( -0.10000000000000E+02 )</td>
</tr>
<tr>
<td>4</td>
<td>( -0.22613983154278E+01 )</td>
</tr>
<tr>
<td>5</td>
<td>( -0.1230303051758E+02 )</td>
</tr>
<tr>
<td>6</td>
<td>( -0.2301010318259E+01 )</td>
</tr>
<tr>
<td>7</td>
<td>( -0.1146403695806E+02 )</td>
</tr>
<tr>
<td>8</td>
<td>( -0.3810253468753E+01 )</td>
</tr>
<tr>
<td>9</td>
<td>( -0.3794631195067E+04 )</td>
</tr>
<tr>
<td>10</td>
<td>( -0.2170272412194E+02 )</td>
</tr>
<tr>
<td>11</td>
<td>( -0.11567068099976E+01 )</td>
</tr>
<tr>
<td>12</td>
<td>( -0.14771051406808E+01 )</td>
</tr>
</tbody>
</table>

### VARIABLE CUALE ASSOCIÉE

<table>
<thead>
<tr>
<th>( U_i )</th>
<th>( U_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>2</td>
<td>( 0.11103846442639E+00 )</td>
</tr>
<tr>
<td>3</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>4</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>5</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>6</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>7</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>8</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>9</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>10</td>
<td>( 0.34952521324158E-01 )</td>
</tr>
<tr>
<td>11</td>
<td>( 0.0 )</td>
</tr>
<tr>
<td>12</td>
<td>( 0.0 )</td>
</tr>
</tbody>
</table>
NIVEAU DE SEPTIÈME

NOMBRE D'ITERATIONS 5

NOMBRE D'APPELS CL S/P CALCULANT LES CONTRAINTES 105
NOMBRE D'APPELS CL S/P CALCULANT LA FONCTION ÉCONOMIQUE 57
NOMBRE D'APPELS CL S/P CALCULANT LE GRADIENT DE LA FONCTION ÉCONOMIQUE 21
NOMBRE D'APPELS CL S/P CALCULANT L'INVERSE AVEC BASE COMMUNE 4
NOMBRE D'APPELS CL S/P CALCULANT L'INVERSE AVEC RECHERCHE DE BASE 0
NOMBRE D'APPELS CL S/P EFFECTUANT LES CHANGEMENTS DE BASE 3
NOMBRE D'APPELS CL S/P CALCULANT LE JACOBIEN DES CONTRAINTES 9
NOMBRE D'APPELS CL S/P EFFECTUANT UNE RECHERCHE DE MAX SANS DICHOTOMIE 5
NOMBRE D'APPELS CL S/P EFFECTUANT UNE RECHERCHE DE MAX AVEC DICHOTOMIE 0
NOMBRE D'APPELS CL S/P EFFECTUANT LA RENTRÉE 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.