13

APPLICATION OF MATHEMATICAL PROGRAMMING TECHNIQUES TO POWER SYSTEM OPTIMIZATION

by S. Arungu-Olende

A Thesis submitted for the Degree of Doctor of Philosophy in Engineering

DEPARTMENT OF ELECTRICAL ENGINEERING, IMPERIAL COLLEGE
OF SCIENCE AND TECHNOLOGY, (UNIVERSITY OF LONDON),
LONDON, S.W.7.

1968

To my parents;

To Christine, Kenneth and Stephen.

ABSTRACT

The aim of this thesis is to give a unified presentation of mathematical programming techniques; and to show, by means of examples, how far these techniques can be used to solve power system optimization problems.

Part 1 is concerned with the theoretical foundations underlying the principal optimization methods in current use; including the simplex method and its variants, logarithmic potential method, integer programming, dynamic programming, maximum principle, geometric programming, and several sequential unconstrained methods for solving non-linear programming problems.

The sequential methods rely heavily on unconstrained optimization techniques. Consequently, the latter, including gradient, modified gradient and non-gradient methods are also discussed with regard to their merits and limitations.

Some decomposition techniques are examined. Several of these have been successfully applied to solve large linear programming problems. Others e.g. diakoptics or decomposition by dynamic programming are still in the early development stage; and further research work is required before these can be used with confidence.

Part II aims to show that a number of power system problems can be handled by mathematical programming methods. Examples include: utilization of hydro-electric resources over a given period of time, a variable-head hydro-electric-thermal scheduling problem (non-linear programming), the balance between power and other uses of water resources (non-linear programming), electrical transmission network design (linear and non-linear programming) generation expansion - plant mix - problem (dynamic programming).

Several suggestions for further work, both in the development (and refinement) of the programming techniques; and in the application of the methods in the power systems field, are indicated.

ACKNOWLEDGEMENTS

I should like to express my deep appreciation to Dr. M.J. Short, Lecturer (my supervisor), and Mr. B.J. Cory, Senior Lecturer, both of the department of electrical engineering, Imperial College of Science and Technology, for their many illuminating and constructive suggestions and discussions in the course of my research work.

I am also indebted to Professor J.R. Mortlock (Visiting Professor) for his continual interest in my work. Professor Mortlock also introduced me to Balfour Beatty and Company Limited, to which I am sincerely thankful - especially to Mr. D. Barr and his staff - for general discussions arising from their long-term experience in power systems planning and design in East Africa.

I am grateful to the Rockefeller Foundation, New York, for financial support during the most part of my research studentship at the Imperial College of Science and Technology, London.

Finally, I should like to thank fellow research students in the Power Systems group for very useful general discussions. I am particularly obliged to A.M. Sasson for beneficial discussions on his computational experience with some of the 'sequential unconstrained'

optimization techniques.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.0 Aims

Mathematical Programming was born in the late 1940's. Since then it has witnessed a phenomenal growth. Research in the field has received an enviable stimulus and has attracted some very fine brains, drawn from many disciplines, including mathematics, physical sciences, engineering and economics. The number of papers published has been impressive, both in quality and quantity.

However, the literature has been scattered in a large variety of journals, many of which are concerned with disciplines other than engineering. Consequently many engineers are completely ignorant of the existence of these publications.

Moreover, much of the work has concentrated only on specific aspects of the Mathematical Programming techniques, thus tending to blur the existence of the many ideas common to the techniques.

Furthermore, the literature on Mathematical Programming has acquired a very high degree of mathematical sophistication and abstraction: a fact which has tended to obscure the conceptual foundations underlying the techniques.

A need, therefore, exists for a more unified presentation of the Mathematical Programming techniques. This should highlight the major developments in the field and should emphasize the concepts underlying the various procedures and the relationships that exist amongst the procedures, together with their inherent limitations.

The aim of the thesis is twofold. First, to fulfil the need stated above. Second, to investigate the feasibility of applying some of the techniques to the study of Power System Planning, Design and Operating problems.

In general, Mathematical Programming concerns analysis of problems of the type: Find the minimum (maximum) of a function (called the ''objective function'') when the variables are subject to inequality or equality constraints.

Special forms of Mathematical Programming include Linear- and Quadratic- programming. In Linear programming, the objective function is a linear combination of the variables and the constraints are linear. Quadratic programming refers to the case where the objective function is a second degree form in the variables; the constraints are linear.

One of the major advances in the field has been the development of dynamic programming. This is based on the concept of multi-stage decision processes. At each stage, a decision is made, following which the next stage is reached. The successive stages are related by known transformation rules. One set of the said decision sequence constitutes the 'best' series ; i.e., Optimizes the given function.

1.1. Systems Approach[22,23,56]

Over the years, the systems we deal with have become increasing large and complex. This has necessitated the evolution of a systematic design approach so as to integrate the many system components while at the same time paying due attention to the inter-relationship between the components.

This new approach is finding wider and wider application in a large variety of situations. It is generally referred to as the ''Systems Approach'' or ''Systems Design''.

Systems Approach may, therefore, be loosely defined as an organized plan in the process of decision-making in any design (or analysis) context. As is illustrated in Fig. 1. it involves complementary practice of both natural and human sciences, together with systems ergonomics and systems engineering.

The study of Ergonomic (Human Factors) systems has received a significant impetus over the past 20 years. This has resulted in genuine improvement in the design of complex systems of which human beings are components. A good account of the development is contained in [73]. Most of the discussions to follow, therefore focus on Systems Engineering aspects of the Systems Approach.

The Systems Engineering method, too, recognizes the fact that each system is an integrated whole even though it is made up of diverse sub-systems (structures). It further recognizes that each system may have a number of objectives; and that the balance between these objectives may differ widely from system to system.

In essence, the method seeks to optimize the over-all system's functions according to specified objectives and to achieve the best compatibility of its parts.

Hence, design optimization of any kind forms an integral part of the Systems approach.

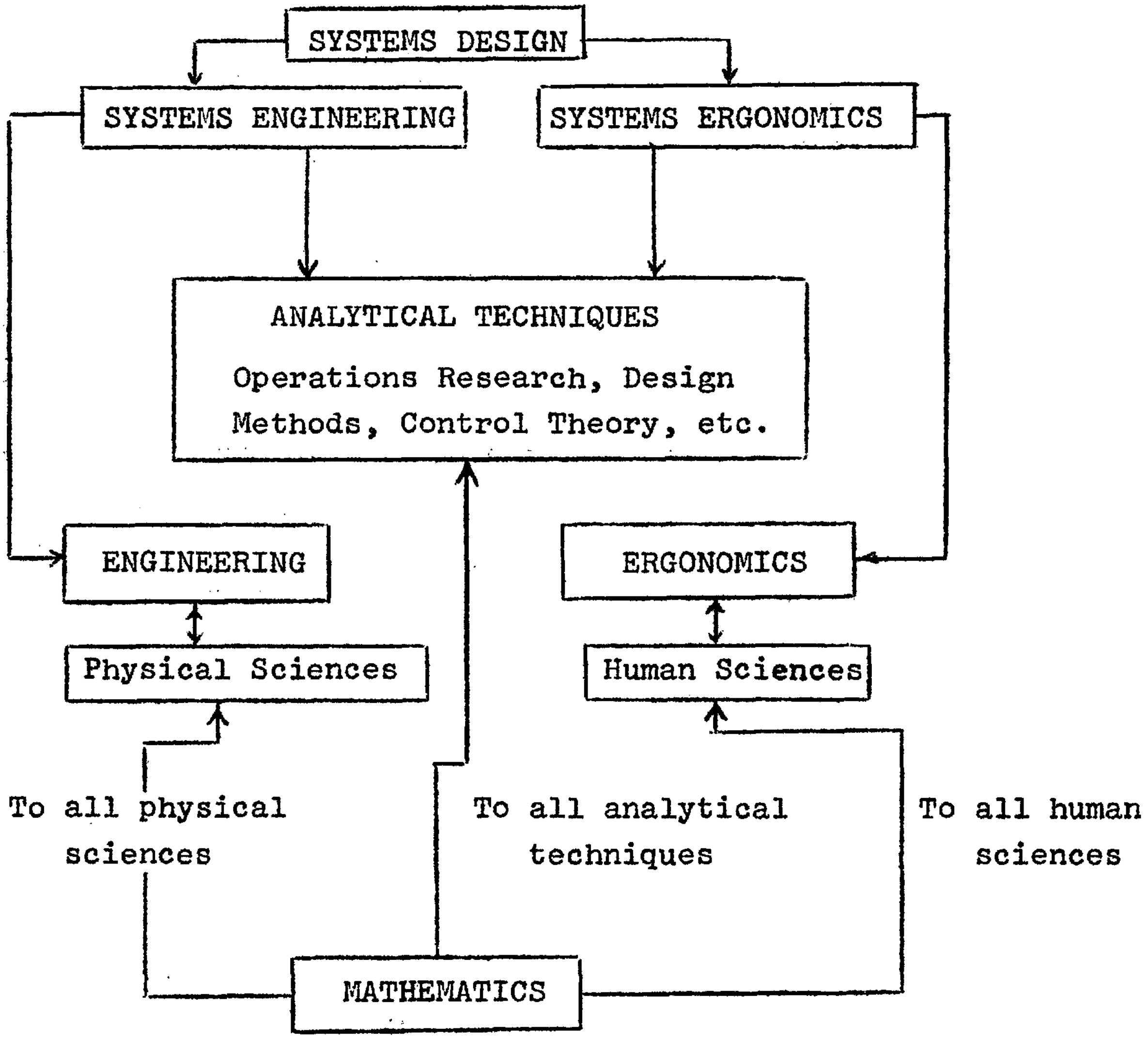

SYSTEMS DESIGN
SYSTEMS ENGINEERING
SYSTEMS ERGONOMICS
ANALYTICAL TECHNIQUES
Operations Research, Design Methods, Control Theory, etc.
ENGINEERING
ERGONOMICS
Physical Sciences
Human Sciences
To all physical sciences
To all analytical techniques
To all human sciences
MATHEMATICS

FIGURE 1.

1.1.0 Methodology of Systems Design:

The methodology of systems design involves, in very broad terms, the following related steps:

a) Problem statement;

b) Problem formulation;

c) Design Realization.

(a) Problem Statement:

This involves in very general terms, identification of the objectives, which are given in rather imprecise terms. Further clarification of the internal structure of the problem is, therefore, required, before any solution can be contemplated.

For example, the task may be to improve the yield in a chemical plant; or to design a distribution network, an electrical machine or a communications system.

The statement at this stage is idealized in that it sets forth the goals to be reached without making reference to the problems encountered in the implementation of the corresponding system in the real world.

(b) Problem Formulation

During the transitional step, the objectives are clarified and given greater precision. System elements, together with their interaction between each other, are clearly defined.

Moreover, the boundary conditions of the systems and sub-systems are investigated: i.e., identified and their degree of importance assessed and indicated. For example, matters relating to physical capability of the system parts (thermal and / or stability limit of, say, electrical lines), reliability, system cost, etc., All this information is required in order to establish the internal structure of the problem.

For a general example of problem formulation, take a case where the task is to minimize design cost, or to maximize the reliability of a system: the formulation stage goes through the following steps:--

i) determination of the characteristics of the system variables and the relationship that exists between them;

ii) definition of the objective(s) or index of performance (in terms of the system variables);

iii) specification of certain (equality or inequality) constraints in the system variables.

The problem is then: Given (i), Optimize (ii) subject to (iii).

Another important procedure carried out during this stage is that of simulating (constructing a simplified representation of) the actual system. We examine the process of simulation in greater detail in section (1.1.1).

(c) Design Realization:

Having formulated the design problem in the form of, say, a mathematical model, the object is then to seek the design which represents the solution to the mathematical version of the design problem.

Simulation of the mathematical relationships on a computer (digital, analogue or hybrid) often plays a crucial role in the search for an acceptable solution.

After completing the mathematical design and evaluating it through simulation and experimentation, the engineer builds a prototype. The prototype is then tested to establish whether or not the requirements are met and the constraints satisfied. If the prototype operates satisfactorily, the work of the engineer is essentially complete.

1.1.1. Modelling

Problem Formulation is also concerned with the construction of a simplified representation of the system called a model. By working with a properly constructed model, the engineer is able to make useful inferences about the proposed real system from experiments conducted with the model.

Either physical or mathematical models may be employed.

1.1.1a. Physical Models:

Physical modelling involves establishing a system analogue of the one being studied. The essential point is that the behaviour of one should closely approximate that of the other (at least for the phenomena being investigated).

The alternative system may be a scale model, which is more convenient to experiment with than the actual system; e.g. the use of scale models in wind tunnels for the design of an aircraft.

Other types of physical models rely upon the analogy between the system being studied and some physical system of a different nature, but which is easier to build and manipulate. Elements of the actual system can be identified with those of the model. Moreover the relationships between the elements of the actual system, and between those of the model are governed by the same physical laws. For example, electrical networks (analogue computers) can be used to study problems of mechanical vibrations because of the similarity between the equations depicting the performance of electrical circuits and mechanical systems.

1.1.1b. Mathematical Models.

The actual system may also be represented in a more flexible manner in the form of mathematical equations. In setting up a mathematical model, the following points are considered:-

i) what are the mathematical relations between the relevant attributes of each of the system elements?

ii) what are the mathematical relations between the attributes of different elements in the system; i.e. what are the mathematical relations representing the interactions between the elements in the system?

Experienced judgement is needed in order to simplify the equations to a point where they are amenable to mathematical analysis without destroying some essential feature of the actual process. This requires that all the assumptions must be made explicit.

1.1.1c. Problems involved in setting up a Mathematical Model.

When formulating a mathematical model we are invariably faced with a number of very difficult problems, some of which are listed below:

i) we are required to have an accurate quantitative knowledge of how the system variables interact. This is a formidable (if not impossible) task. For, in many cases, the nature of the physical characteristics of the system's elements may not be fully understood. Furthermore, there may be a measure of uncertainty with regard to the external disturbances acting on the system. Consequently, some degree of idealization is inevitable. The question which immediately arises is: how much idealization can we allow and still obtain satisfactory results?

There is no cut and dried answer to this question; however, experience and skill in dealing with problems of the same nature may prove quite useful. It must be emphasized, though, that the nature of idealization permissible is, by and large, determined by the specific problem and depends both on the properties of the system considered and on just which questions we want answered.

ii) In principle, when formulating a model, we want to take into account only those factors which govern the sets of behaviour of the systems that are of interest to us. It is quite unnecessary (in fact, practically impossible) to consider all the properties without exception.

But even if we should succeed in accounting for only the relevant parts of the properties, the resulting system may be so complicated that its solution (computational) would be extremely tedious at best. Further approximations and / or reduction in the number of variables to be considered must then be undertaken. This may, however, give rise to a model that does not sufficiently represent the actual system.

iii) One important requirement is that of the determination of a measure of effectiveness (i.e. performance index, or objective function) that is expressible in terms of the system variables. This too, is an insuperable task. A realistic performance index (i.e. an index which represents most of the design requirements of the problem) is extremely difficult to define. For, in practice, any of the relevant criteria is rarely explicit enough to allow for a clear mathematical representation. And, quite often, it is very difficult to express the performance index in terms of some very important system variables; e.g. the reliability of a transformer in terms of the length of an electrical transmission line.

Even if a realistic criterion has been defined, it is frequently found that the basic concept of the particular performance index is too restrictive.

1.1.1d. Words of Caution on the Use of a Model.

When involved in any investigation that requires the use of a model, the following factors must always be kept in mind:

a) the model is, in fact, different from the prototype system that the former is designed to describe. Consequently, not all that is true of the model need be true of the prototype.

b) the model is usually formulated in such a way as to permit examination methods not applicable to the original system. Therefore, not all that is either logically necessary or inferable from the model need be a logical inference with regard to the actual system.

Failure to appreciate the above points may result in wrong conclusions being drawn about the actual system.

PART I

Mathematical Programming

CHAPTER 2

MATHEMATICAL PROGRAMMING PROBLEMS.

2.0. Optimization and Mathematical Programming Problems

Optimal use of available resources is an implicit goal of every human endeavour. Consequently, optimization problems have long been of interest to mathematicians, physical scientists and engineers.

Since the middle of the eighteenth century, methods of differential calculus and calculus of variations have been utilized to solve certain types of optimization problems in geometry and physics. But, except for very simple problems, the tedious computations that were required hindered a wide-spread application of these methods.

However, developments in high-speed, automatic computers over the past 20 years have facilitated the application of most of these older methods. Furthermore, much new research has been directed on the type of Optimization problems that are usually not amenable to solutions by the classical methods of Calculus. These new types of optimization problems take into account inequality constrains and are often referred to as Mathematical Programming problems.

2.1. The General Programming Problem:

The general programming problem can be formulated as follows: Select the values of a number of variables so that an objective function (which has already been defined) is minimized (maximized) among all choices of values that satisfy a set of inequality (and/or equality) constraints on the variables and their function(s).

Mathematically, the problem may be formulated as:

$$\text{Minimize} \quad F(\bar{x}) \; ; \qquad (2.1.a)$$

$$\text{subject to*} \quad \begin{array}{lcl} \bar{G}(\bar{x}) & \{ \leq , \geq \} & 0 \; , \\ \bar{x} & \geq & 0 \end{array} \qquad (2.1.b)$$

where $\bar{x}$ is an (nxl) column matrix of n components and $\bar{G}(\bar{x})$ represents a column vector of m functions;

i.e.

$$\bar{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

and

$$\bar{G}(\bar{x}) = \begin{bmatrix} G_1(\bar{x}) \\ G_2(\bar{x}) \\ \vdots \\ G_m(\bar{x}) \end{bmatrix}$$

* Some constraints may be of the strict equality type:

e.g. $G_i(\bar{x}) \geq 0 \qquad i = 1, \ldots, K$

$G_i(\bar{x}) = 0 \qquad i = K+1, \ldots, m$

2.2.1. The Linear Programming Problem:

This is a special case of the general programming problem in which $F(\bar{x})$ is a linear combination of the variables; and $\bar{G}(\bar{x})$ is a linear transformation. The problem thus becomes:

$$\text{Minimize} \qquad F(\bar{x}) = \bar{c}^T\bar{x} \qquad (2.2.a)$$

$$\text{with} \qquad \bar{G}(\bar{x}) = \bar{A}\bar{x} \geq \bar{b}$$

$$\bar{x} \geq 0 \qquad (2.2.b)$$

where $\bar{c}^T = [c_1, c_2, \ldots, c_n]$ is the transpose of an (nxl) column matrix $\bar{c}$ and $\bar{b}$ is an (mxl) column matrix - usually referred to as the requirement vector - denoted by:

$$\bar{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

$\bar{A}$ is an (mxn) transformation matrix given by

$$\begin{bmatrix} a_{11} & a_{12} & \vdots & a_{1n} \\ a_{21} & a_{22} & \vdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \vdots & a_{mn} \end{bmatrix}$$

and $m \leq n$.

2.2.2. Integer Linear Programming

An integer linear programming problem is a linear programming problem for which it is further required that the variables must take integer values. The field is divided into "all (pure) integer programming" when all the variables must be integers; and "mixed integer programming" if only certain specified variables must be integers.

In the most general terms, the problem may be represented:

$$\text{Minimize } F(\bar{x}) = \bar{c}^T\bar{x} \qquad (2.2.c)$$

$$\text{subject to } \bar{G}(\bar{x}) = \bar{A}\bar{x} \geq b, \quad \bar{x} \geq 0 \qquad (2.2.d)$$

$$x_i \text{ an integer } \quad i \in T \qquad (2.2.e)$$

If T contains all i , the problem is all integer; and if some of the i (i=1,...,n) then we have a mixed integer problem. Note that if T is empty, then the problem reduces to the normal linear programming problem.

2.2.3. Parametric Linear Programming Problem:

This may be considered as the most general form of the linear programming problem in that either the requirement vector or the coefficients of the objective function, (or both), is allowed to change.

A simple case of such change occurs when each of the coefficients of the objective function is a linear function of a parameter, λ_f ; or when each element of the requirement vector is a linear function of a parameter, λ_f. For example:

a) Parametric objective function

$$\text{Minimize} \quad F(\bar{x}) = (\bar{c} + \bar{\lambda}_f)^T \bar{x} \qquad (2.2.f)$$

$$\text{subject to} \quad \bar{A}\bar{x} \geq \bar{b}$$

$$\bar{x} \geq 0 \qquad (2.2.g)$$

b) Parametric Requirement vector:

$$\text{Minimize} \quad F(\bar{x}) = \bar{c}^T \bar{x} \qquad (2.2.h)$$

$$\text{with} \quad \bar{A}\bar{x} \geq \bar{b} + \bar{\lambda}_b{}^T \bar{d}$$

$$\bar{x} \geq 0 \qquad (2.2.i)$$

c) Parametric objective function and Requirement vector:

$$\text{Minimize} \quad F(\bar{x}) = (\bar{c} + \bar{\lambda}_f)^T \bar{x} \qquad (2.2.j)$$

$$\text{subject to} \quad \bar{A}\bar{x} \geq \bar{b} + \bar{\lambda}_b{}^T \bar{d}$$

$$\bar{x} \geq 0 \qquad (2.2.k)$$

2.3.1. Quadratic Programming Problem:

A quadratic programming problem is the simplest form of the non-linear programming problem: the objective function is a second degree form in the variables, while the constraints are linear. It is thus somewhat more general than the linear programming problem; and is denoted by:

$$\text{Minimize} \quad F(\bar{x}) = (\tfrac{1}{2})\, \bar{x}^T \bar{Q}\bar{x} + \bar{c}^T \bar{x} \qquad (2.3.a)$$

$$\text{subject to} \quad \bar{A}\bar{x} \geq \bar{b} \qquad (2.3.b)$$
$$\bar{x} \geq 0$$

where $\bar{Q}$ is an (nxn) matrix. For the general formulation given above, it is required that $\bar{Q}$ be positive semi-definite (i.e. ≥ 0). Clearly, if $Q = 0$, then the first term of equation (2.3.a) vanishes and we are left with a linear programming problem. For a strictly quadratic form, therefore, $\bar{Q}$ must be positive definite ($\bar{\bar{Q}} > 0$)·

2.3.2. Integer Quadratic Programming Problem:

As with the integer linear programming, it may be required that some or the whole of the solution variables take only integer values; e.g.

$$\text{Minimize} \quad F(\bar{x}) = (\tfrac{1}{2})\, \bar{x}^T\bar{Q}\bar{x} + \bar{c}^T\bar{x} \qquad (2.3.c)$$
$$\text{subject to} \quad \bar{A}\bar{x} \geq \bar{b}$$
$$\bar{x} \geq 0$$
$$\bar{x} \text{ has integral components} \qquad (2.3.d)$$

2.3.3. Parametric Quadratic Programming:

A parametric quadratic programming problem also concerns cases where the quadratic objective function, or the requirement vector or a combination of the two is allowed to vary; for example:

$$\text{a) Minimize} \quad F(\bar{x}) = (\tfrac{1}{2})\bar{x}^T\bar{Q}\bar{x} + (\bar{c} + \lambda_f)^T\bar{x} \qquad (2.3.e)$$

$$\text{subject to} \quad \begin{aligned} \bar{A}\bar{x} &\geq \bar{b} \\ \bar{x} &\geq 0 \end{aligned} \tag{2.3.f}$$

$$\text{b) Minimize} \quad F(\bar{x}) = (\tfrac{1}{2})\bar{x}^T\bar{Q}\bar{x} + \bar{c}^T\bar{x} \tag{2.3.g}$$

$$\text{subject to} \quad \begin{aligned} \bar{A}\bar{x} &\geq \bar{b} - \bar{\lambda}_b{}^T\bar{d} \\ \bar{x} &\geq 0 \end{aligned} \tag{2.3.h}$$

or

$$\text{c) Minimize} \quad F(\bar{x}) = (\tfrac{1}{2})\bar{x}^T\bar{Q}\bar{x} + (\bar{c} + \bar{\lambda}_f)^T\bar{x} \tag{2.3.i}$$

$$\text{subject to} \quad \begin{aligned} \bar{A}\bar{x} &\geq \bar{b} - \bar{\lambda}_b\bar{d} \\ \bar{x} &\geq 0 \end{aligned} \tag{2.3.j}$$

Comment:

A large number of problems can be formulated in terms of these two special cases (linear-and-quadratic) of mathematical programming by means of appropriate approximation of an otherwise complex function. Once a problem has been formulated in either of the two forms one can obtain its optimal solution in a finite number of iterative steps: there are now available many powerful computational techniques (e.g. the "Simplex method") for obtaining such a solution.

This is a very important factor; for with the general non-linear programming problem, an exact solution can never be obtained in a finite number of steps - although convergence may be quite rapid.

2.4. The Lagrangian Problem:

The Lagrangian problem is a general problem of the calculus of variations and has found extensive application in many fields of science and engineering.

Other general problems of calculus of variations include the problems of Bolza and Mayer. Theoretically, all the three are equivalent in that any one of them can be transformed to the other by a change of coordinates.

(a) Lagrange Multipliers:

Consider a mathematical programming problem in which all the constraints are in the form of strict equality, or whose inequality constraints have been transformed to equality by the introduction of appropriate slack or surplus variables

$$\text{Minimize} \quad F(\bar{x}) \tag{2.4.a}$$

$$\text{with} \quad G(\bar{x}) = 0 \tag{2.4.b}$$

Following the Lagrange multiplier rule, we obtain an augmented function by multiplying each constraint function, $G_i(\bar{x})$ by a factor, and adding the result to the objective function:

$$\psi(\bar{x},\bar{\lambda}) = F(\bar{x}) - \bar{\lambda}^T\bar{G}(\bar{x}) \tag{2.4.c}$$

where $\psi(\bar{x},\bar{\lambda})$ is the unconstrained augmented function, called the Lagrangian Function, whose minimum has to be found; and $\bar{\lambda}$ is a vector representing the Lagrange multipliers.

At the minimum of the Lagrangian function, the following conditions must hold:

$$\frac{\partial\psi(\bar{x},\bar{\lambda})}{\partial\lambda_j} = \bar{G}(\bar{x}) = 0 \qquad (2.4.d)$$

$$\frac{\partial\psi(\bar{x},\bar{x})}{\partial x_j} = \partial F(\bar{x}) - \bar{\lambda}^T \partial G(\bar{x}) = 0 \qquad (2.4.e)$$

where $\partial F(\bar{x})$ is the gradient of the objective function and $\partial\bar{G}(\bar{x})$ is the differential of $\bar{G}(\bar{x})$.

Note that in the above discussions it has been assumed that both $F(x)$ and $\bar{G}(x)$ have continuous derivatives.

An example:

A linear programming problem can also be formulated as a Lagrangian problem and solved by the method of Lagrange multipliers.

Consider a standard general linear programming problem:

$$\text{Minimize} \quad F(x) = \sum_{j}^{n} c_j x_j \qquad (2.4.f)$$

$$\text{subject to} \quad x_{j_n} \geq 0 \qquad (2.4.g)$$

$$\sum_{i=1}^{n} a_{ij} \, x_j = b_i \qquad i = 1,\ldots,m \qquad (2.4.h)$$

The first step is to replace the non-negative condition ($x_j \geq 0$) by

$$x_j - u_j^2 = 0 \qquad j = 1,\ldots,n \qquad (2.4.i)$$

Following the Lagrange multiplier rule, a new unconstrained minimum is determined:

$$\psi(\bar{x},\bar{\lambda}) = \sum_{j=1}^{n} c_j x_j - \left[\lambda_1(\sum_{j=1}^{n} a_{1j}x_j - b_1) + \ldots + \lambda_m(\sum_{j=1}^{n} a_{mj}x_j - b_m)\right] - \left[\lambda_{m+1}(x_1 - u_1{}^2) + \ldots + \lambda_{m+n}(x_n - u_n{}^2)\right] \quad (2.4.j)$$

or

$$\psi(x,\lambda) = (\sum_{i=1}^{m} \lambda_i b_i) + (c_1 - \sum_{i=1}^{m} \lambda_i a_{i1} - \lambda_{m+n})x_1 + \ldots + (c_n - \sum_{i=1}^{m} \lambda_i a_{in} - \lambda_{m+n})x_n + \sum_{i=1}^{n} \lambda_{m+i}\, u_i{}^2 \quad (2.4.k)$$

For $\psi(x,\lambda)$ to be a minimum, the following conditions must be met:

a) the coefficient of x_j vanish;

b) the coefficient of $u_j{}^2$ are non-negative;

c) the partial derivative with respect to u_j vanish.

2.5. Generalized Lagrangian Problem:

The generalization of the Lagrange problem to handle both equality and inequality constraints is largely due to H.W. Kuhn and A.W. Tucker.[51]

Consider a general programming problem:

Maximize $F(x)$

subject to $G_i(\bar{x}) \geq 0 \quad i = 1,\ldots,\ell$

$G_i(\bar{x}) = 0 \quad i = \ell+1,\ldots,m \quad (2.5.a)$

As in the previous section, an augmented function

$$\psi_G(\bar{x},\bar{\lambda}) \quad = \quad F(\bar{x}) \quad + \quad \sum_{i=1}^{m} \lambda_i G_i(\bar{x}) \qquad (2.5.b)$$

is formed.

If $\psi_G{}^*(\bar{x}^*\bar{\lambda}^*) \quad = \quad F(\bar{x}^*) + \sum_{i=1}^{m} \lambda_i^* \; G_i(\bar{x}^*)$

is an optimal solution to (2.5.b) the following relations must be satisfied:

$$\nabla_x\psi_G(\bar{x}^*, \bar{\lambda}^*) \quad \leq \quad 0 \qquad (2.5.c.1)$$

$$\bar{x}^{*T}\nabla_x\psi_G(\bar{x}^*,\bar{\lambda}^*) \quad = \quad 0 \qquad (2.5.c.2)$$

$$\nabla_\lambda\psi_G(\bar{x}^*, \bar{\lambda}^*) \quad \geq \quad 0 \qquad (2.5.c.3)$$

$$\bar{\lambda}^{oT}\nabla_\lambda\psi_G(\bar{x}^*, \bar{\lambda}^*) \quad = \quad 0 \qquad (2.5.c.4)$$

$$\bar{x}^* \quad \geq \quad 0 \qquad (2.5.c.5)$$

$$\bar{\lambda}^* \quad \geq \quad 0 \qquad (2.5.c.6)$$

where $\nabla_x\psi_G(\bar{x},\bar{\lambda})$ is the vector $(\partial\psi_G/\partial x_1,\ldots, \frac{\partial\psi_G}{\partial x_n})$

and $\nabla_\lambda\psi_G(\bar{x},\bar{\lambda})$ is the vector $(\partial\psi_G/\partial\lambda_1,\ldots, \frac{\partial\psi_G}{\partial\lambda_m})$

Note, however, that the above relations consitute the necessary conditions for optimality to problem (2.5.b) only if the following hypothesis (<u>known as constraint qualification property</u>)holds:

<u>Constraint qualification property</u>:

Let $x_i^* \;=\; 0 \qquad i = 1,2,\ldots,K$

$\quad\;\; x_i^* \;>\; 0 \qquad i = (K+1),\ldots,n$

(i.e. the values of some optimal variables are identically zero).

We also observe (from 2.5.a) that some of the constraints are strict equalities $(G_i(\bar{x}) = 0,\ i = \ell+1,\ldots,m)$.

The feasible region is such that if for any x on the boundary defined above the following conditions hold:

$$\text{i)}\quad \Sigma(\partial G_j(x)/_{\partial x_i})dx_i \ \geq\ 0 \qquad j = \ell+1,\ldots,m$$

$$\text{ii)}\quad dx_i \ \geq\ 0 \qquad\qquad i = 1,\ldots,K$$

then the direction $dx_1, dx_2,\ldots, dx_n$ is tangential to some arc from x into the interior.

If $\psi_G^o(x^o)$ is the optimum of $\psi_G(x)$ then

$$\Sigma(\partial F(\bar{x}^o)/\partial x_i)dx_i \ \leq\ 0$$

for all directions into the interior, and hence for all the directions described above. Consequently we have:

$$\frac{\partial F(\bar{x}^o)}{\partial x_i} + \sum_{j=1}^{m} \lambda_j^o \frac{\partial G_j(\bar{x}^o)}{\partial x_i} + z_i = 0 \qquad (2.5.d.1)$$

$$\lambda_j^o \geq 0 \qquad j = 1,2,\ldots,m \qquad (2.5.d.2)$$

$$z_i^o \geq 0$$

$$x_i^o \geq 0 \qquad i = 1,2,\ldots,n \qquad (2.5.d.3)$$

$$\text{with } z_i^o = 0 \quad \text{for } i = (k+1),\ldots,n \qquad (2.5.d.4)$$

$$\text{and } \lambda_j^o = 0 \quad \text{for } j = (\ell+1),\ldots,\ell \qquad (2.5.d.5)$$

We see, therefore, (from 2.5.c.2) that

$$\bar{x}^{oT}\nabla_x\, \psi_G^o(\bar{x}^o,\lambda^o) = \sum_{i=1}^{n} x_i^o z_i^o = 0$$

since either x_i^o or z_i^o is zero. Furthermore, relation (2.5.c.4) means that $\sum_{j=1}^{m} G_j(\bar{x}^o)\, \lambda_j^o = 0$.

In the above discussions, it has been assumed that the functions $F(\bar{x})$ and $\bar{G}(\bar{x})$ are concave (convex) and are differentiable. Furthermore, the objective function $F(\bar{x})$ is assumed to be bounded.

In an earlier work F.John[43] had also dealt with analogous generalizations. He considered a non-linear programming problem: Find $\bar{x}$ such that $F(\bar{x})$ is minimized (maximized) subject to $\bar{G}(\bar{x}) \geq 0$; and gave the following general theorem:

If the functions $F(x)$, $G_i(\bar{x})$, $(i=1,\ldots,m)$, are continuously differentiable, then a necessary condition that $\bar{x}$ must be a local minimum to the above problem is that there exist scalars: $\lambda_o,\lambda_1,\ldots,{}_m$ not all zero such that the following inequalities and equalities are satisfied:

$$G_i(\bar{x}^*) \geq 0 \qquad i = 1,\ldots,m \tag{2.5.e.2}$$

$$\lambda_o \nabla F(\bar{x}^*) - \sum_{i=1}^{m} \lambda_i \nabla G_i(\bar{x}^*) = 0 \tag{2.5.e.3}$$

$$\lambda_i G_i(\bar{x}^*) = 0 \qquad i = 1,\ldots,m \tag{2.5.e.4}$$

$$\lambda_i \geq 0 \qquad i = 1,\ldots,m \tag{2.5.e.5}$$

Note that conditions (2.5.e.2),(2.5.e.3),(2.5.e.4) and (2.5.e.5) are necessary for $\bar{x}^*$ to be an optimal solution without any additional hypotheses.

Since the original work of Kuhn and Tucker further theoretical research has been directed at the question of constraint qualification property. This has given rise to the regularity condition - by Arrow, Uzawa and Hurwicz[1]- which incorporates cases where $\bar{x}$ ranges over more general spaces and is subjected to more general constraints. The regularity condition is actually a sufficient criterion for a certain weaker form of the Kuhn-Tucker constraint qualification property.

2.6. Equivalent Formulations: Duality

Associated with any mathematical programming problem (usually called the primal) is another, called the dual. The objective of the dual is to maximize while that of the primal is to minimize - and vice versa.

For the linear programming problem the pair of (primal and dual) programs are represented:

Primal Problem:

$$\text{Minimize} \quad F(\bar{x}) = \bar{c}^T\bar{x} \tag{2.6.a}$$

$$\begin{aligned} \text{subject to} \quad \bar{A}\bar{x} &\geq \bar{b} \\ x &\geq 0 \end{aligned} \tag{2.6.b}$$

Dual Problem

$$\text{Maximize} \quad DF(\bar{x}) = \bar{b}^T\bar{y} \tag{2.6.c}$$

$$\begin{aligned} \text{subject to} \quad \bar{A}^T\bar{y} &\leq \bar{c} \\ \bar{y} &\geq 0 \end{aligned} \tag{2.6.d}$$

where $\bar{y} = (y_1, y_2, \ldots, y_m)$ is an mx1 matrix and $\bar{b}$, $\bar{A}$, and $\bar{c}$ are the same for the primal problem (see also section 2.2.1). Observe that the 'dual' of the dual problem is the original primal problem.

Corresponding programmes for the quadratic and the general convex programming problem can also be formulated:

Primal Quadratic Programme:[18]

Minimize $F(\bar{x}) = \frac{1}{2} \bar{x}_K^T Q \bar{x}_K + c_K^T \bar{x}_K + c_\ell^T \bar{x}_\ell$ (2.6.e)

subject to

$$\bar{A}_K \bar{x}_K + \bar{A}_\ell \bar{x}_\ell \geq \bar{b}$$
$$\bar{x}_K \geq 0, \quad \bar{x}_\ell \geq 0 \qquad (2.6.f)$$

Dual Quadratic Programme:

Maximize $DF(\bar{x}) = -\frac{1}{2}\bar{y}_K^T \bar{Q}^{-1} \bar{y}_K + \bar{b}^T \bar{y}_\ell$ (2.6.g)

subject to

$$\bar{A}_K^T \bar{y}_\ell - y_K \leq \bar{c}_K$$
$$\bar{A}_\ell^T \bar{y}_\ell \leq \bar{c}_\ell \qquad (2.6.h)$$
$$\bar{y}_\ell \geq 0; \quad \bar{y}_K \text{ unrestricted}$$

Primal Convex Programming Problem:[24]

Minimize $F(\bar{x})$

subject to $\bar{G}(\bar{x}) \geq 0$

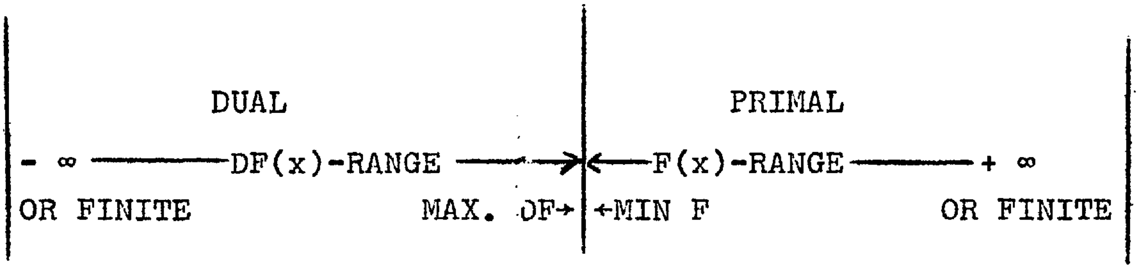
Dual Problem:

Maximize $DF(\bar{x},\bar{y}) = F(\bar{x}) - \bar{y}^T \bar{G}(\bar{x})$ (2.6.i)

subject to

$$\nabla_x F(\bar{x}) = \bar{y}^T \nabla_x \bar{G}(x) \qquad (2.6.j)$$
$$\bar{y} \geq 0$$

The range of possible values of $DF(\bar{x})$ and $F(\bar{x})$ (for a general convex problem, linear or non-linear) is illustrated in Fig. 4. It is assumed that both the primal and the dual are feasible.



<u>Fig. 4. Range of possible values of the Primal & Dual Programmes</u>

Fig. 4. can be summarized by the <u>Duality Theorem</u>:

> If solutions to the primal and the dual problems exist, the value, $F(\bar{x})$ of the objective function corresponding to any feasible solution of the primal is greater than or equal to the value $DF(\bar{x})$ of the objective corresponding to any feasible solution of the dual (i.e. $F(\bar{x}) \geq DF(\bar{x})$) Furthermore, an optimal solution exists for both problems and Max. $DF(\bar{x})$ = Min. $F(\bar{x})$.

The above theorem is utilized in a number of mathematical programming techniques in deciding when to stop a minimization (maximization) process.[24,55] The programmes solve the primal optimization problem in such a way that a set of points are dual feasible. The computational process stops when the difference between the primal feasible and the dual feasible values fall within a [illegible] specified limit.

The principle of duality has also led to the development of many new and useful programming techniques: for example, geometric programming (chapter 5), dual-simplex and primal dual methods, for handling parametric linear programming, dual quadratic programming etc.,

Moreover, the reciprocal nature of the primal-dual constraints can find use in the generation of a two level decomposition and coordination process, thus enabling the solution of several medium size convex programming problems, which together might comprise a prohibitively large integrated problem.[62]

Comment:

The principle of duality is also manifest in a number of physical, economic and mathematical problems:

e.g. a) physical; in electric circuit theory, duality exists between series and parallel circuits.

b) economics; the problem of determining the prices of foods produced by one economy may be considered as the dual of determining the quality to be produced.

CHAPTER 3

TECHNIQUES FOR SOLVING THE LINEAR PROGRAMMING PROBLEM.

3.0

As we have indicated in Chapter 2, linear programming is a special form of the general mathematical programming problem, in which both the objective function and the constraints are linear.

Methods for solving linear programming-type problems have been studied extensively as far back as 1947. Of these, the most successful and most widely used is the Simplex algorithm, developed by G. Dantzig.[15] Much of the chapter will be concerned with a brief discussion of the Simplex method and its variants. We shall also examine the logarithmic potential method developed by R. Frisch.[30]

3.1. Some definitions:

In the course of our discussions, a number of terms will be encountered on many occasions. Some of these are defined below.

Theorem 3.1.

If a given set of m simultaneous equations in n unknowns $(n \geq m)$

$$\bar{A}\bar{x} = \bar{b}$$

has a solution, and if the rank of A, $r(A) = m$; then $\bar{b}$ can be expressed as a linear combination of m linearly independent columns of $\bar{A}$.

Theorem 3.1 is based on the assumption that $r(\bar{A}) = r(\bar{A}_b)$ where $\bar{A}_b$ is the augmented matrix

$$\bar{A}_b = (\bar{A},\bar{b}) \tag{3.1.2}$$

Let $n > m$ then (3.1) contains at most m linearly independent equations. A solution can be obtained by assigning arbitrary values to the (n-m) variables, not associated with the non-singular (mxm) matrix, $\bar{A}$.

The remaining m variables are uniquely determined by the (n-m) variables. If all the (n-m) variables are set equal to zero, the solution to the resulting system of equations is called a basic solution.

The m variables (some of which may be zero) are called basic variables; while the other (n-m) are referred to as non-basic variables.

A basic solution that obeys the set of constraints (2.2.b) i.e. $x_i \geq 0$ is a basic feasible solution. Any basic feasible solution that minimizes (maximizes) an objective function, $F(\bar{x})$ is a basic optimal feasible solution.

A basic solution to $\bar{A}\bar{x} = \bar{b}$ is degenerate if one or more of the basic variables is equal to zero.

3.2. The Structure of the Solution:

A basic feature of a linear programming problem is that the feasible (admissible) region is convex;

i.e. if $(x_1', x_2', \ldots, x_m')$ and $(x_1'', x_2'', \ldots, x_m'')$ are any two points on the feasible region, any point $(x_1, x_2, \ldots, x_m)$ situated on the straight line joining the above two points will also belong to the feasible region. This fact is of great help for computational purposes.

The optimum point set, $\bar{x}^*$, (i.e. the set of points in the feasible region which minimizes the objective function, $F(\bar{x})$) must always lie on the boundary of the feasible region. This means that at least one of the n variables must be equal to zero at the minimum.

The number of degrees of freedom ($\equiv n-m$) - called the dimensionality - of the point set, may be one of the numbers $\delta g = 0,1,\ldots,(n-m-1)$. $\delta g = 0$. means that there is only one well-defined corner on the boundary of the feasible region where $F(\bar{x})$ is a minimum; while the case $\delta g = 1$ means that the minimum is reached along an edge that connects two corners, and so on. Fig. 3.1 illustrates the cases $(n-m) = 3$ and $\delta g = 0,1,2$.

Whatever the dimensionality of the optimum point set, there exists at least one corner with optimal properties i.e. at least one optimal point such that at this point at least $(n-m)$ of the variables are equal to zero.

If the simplex method (Section 3.3) is used, the procedure consists of essentially in first seeking some corner on the feasible region. This is followed by systematic moves along edges from corner to corner until an optimal corner is reached.

3.3. The Simplex Method

Consider a standard form of the linear programming problem (i.e. one for which the inequalities have been reduced to equalities by the introduction of appropriate slack variables):[41]

$$\text{Minimize} \quad F(\bar{x}) = \sum_{i=1}^{n} c_i x_i \qquad (3.3.1)$$

$$\text{with} \quad \begin{bmatrix} a_{11}x_1 & a_{12}x_2 & \cdots & a_{1n}x_n \\ a_{21}x_1 & a_{22}x_2 & \cdots & a_{2n}x_n \\ \vdots & \vdots & & \vdots \\ a_{m1}x_1 & a_{m2}x_2 & \cdots & a_{mn}x_n \end{bmatrix} \begin{matrix} = \\ = \\ \\ = \end{matrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \qquad (3.3.2)$$

$$\text{and} \quad x_1 \geq 0$$

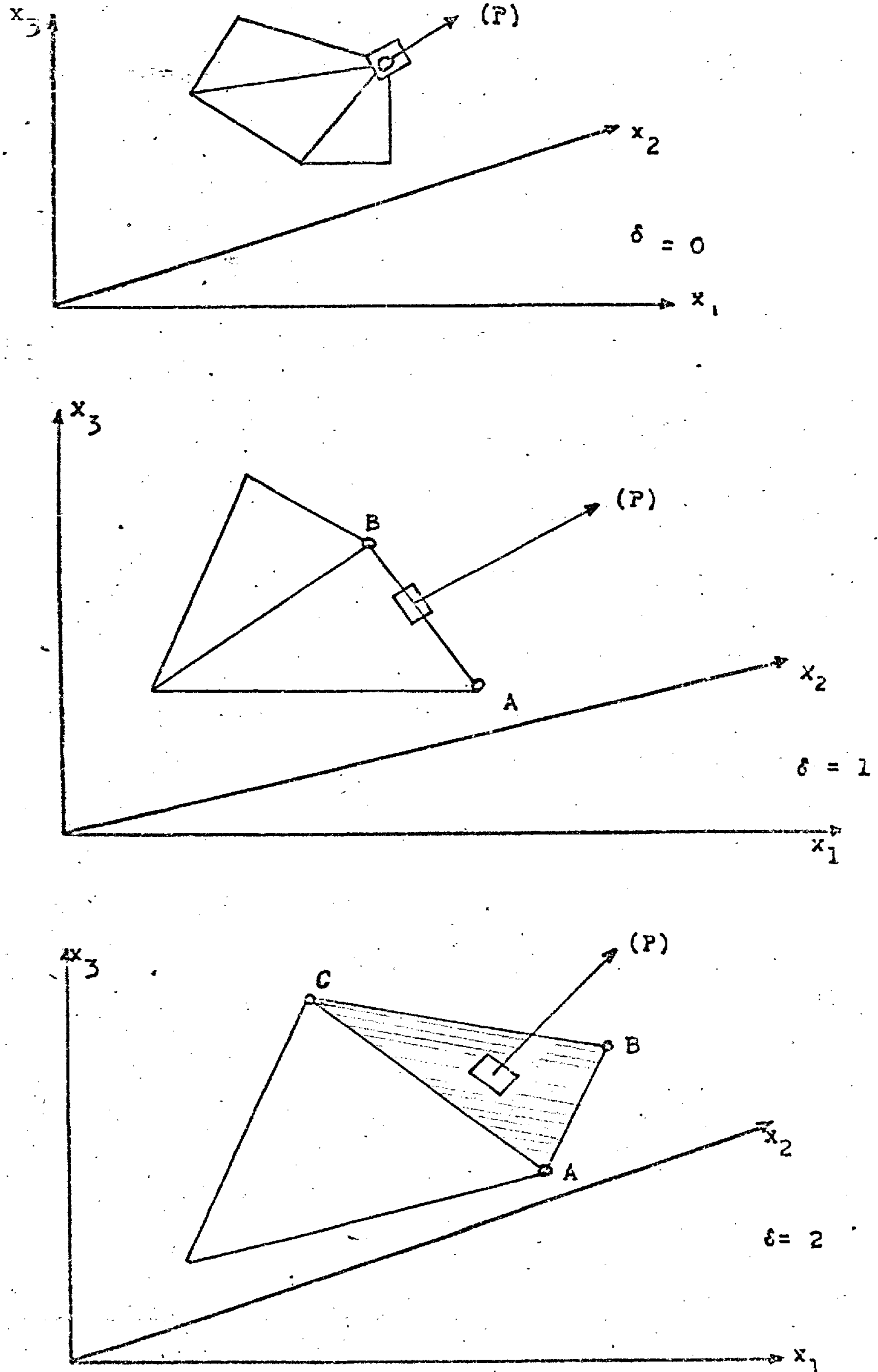
Solutions to (3.3.2) have the following important characteristics:

Theorem 3.2

a) Let the rank of (3.3.2) be equal to m, then if there exists a feasible solution to (3.3.2) - not necessarily optimal - there is at least one basic feasible solution to (3.3.2) (i.e. with at least (n-m) variables equal to zero).

b) If in addition a lower bound to the objective function, $F(\bar{x})$, (3.3.1) exists, then an optimal

Fig. 3.1. Number of Degrees of Freedom



basic feasible solution to (3.3.2) exists.

The simplex method makes use of the above characteristics.

Suppose that we have a basic feasible solution for which the objective function is not necessarily a minimum. The simplex method proceeds by eliminating each of the m basic variables from all but one of the (m) equations by choosing a pivot term in a manner similar to the ordinary elimination for solving m equations in m unknowns.

At the end of each elimination process, a simple test is applied in order to determine whether or not the solution is optimal. If not, a corrective procedure is applied which substitutes one of the non-basic variables for one of the basic variables. The procedure is repeated (in a finite number of times) until the optimum is reached - if one exists.

The net effect of an elimination process is that the original system of basic equations is replaced by an equivalent system of equations called the canonical system. The elimination procees is, therefore, sometimes referred to as a reduction to canonical form.[15]

If $x_1, x_2, \ldots, x_m$ are the variables selected for elimination, the canonical system takes the form:

$$\begin{array}{llllll}
x_1 & + \bar{a}_{1\ m+1}\, x_{m+1} & + \cdots & + \bar{a}_{1n}x_n & = & \bar{b}_1 \\
\quad x_2 & + \bar{a}_{2\ m+1}\, x_{m+1} & + \cdots & + \bar{a}_{2n}x_n & = & \bar{b}_2 \\
\cdot\ \cdot\ \cdot & \cdot\ \cdot\ \cdot\ \cdot\ \cdot & \cdot\ \cdot & \cdot\ \cdot\ \cdot & \cdot & \cdot\ \cdot \\
\qquad x_m & + \bar{a}_{m\ m+1}\, x_{m+1} & + \cdots & + \bar{a}_{mn}x_n & = & \bar{b}_m \\
\qquad -F(x) & + \bar{c}_{m+1\ m+1}x_{m+1} & + \cdots & + \bar{c}_n x_n & = & -\bar{F}_o
\end{array} \qquad (3.3.3)$$

where $\bar{a}_{ij}$, $\bar{b}_j$ and $\bar{F}_o$ are constants. Equation (3.3.3) is equivalent to (3.3.2). In the discussions that follow, we shall assume that the system of equations are in the canonical form.

3.3.1. Test for Optimal Basic Feasible Solution

The optimal solution is established by the following theorems:

Theorem 3.3

If in the canonical form, the values of $\bar{b}_i$ and $\bar{c}_j$ are non-negative, the basic solution is optimal.

Theorem 3.4

If in the canonical system, the basic solution is feasible and $\bar{c}_j > 0$ for all nonbasic variables, the solution is the unique feasible optimal solution.

Corollary.

If in the canonical system the basic solution is feasible and $\bar{c}_j \geq 0$ for all j, then no other feasible solutions can also be optimal unless the values of $x_j = 0$ whenever $\bar{c}_j > 0$.

3.3.2. Improving a Non-optimal Basic Feasible Solution.

An important property of the standard simplex method is that it works only with basic solutions which are feasible (i.e. $\bar{b}_i \geq 0$).

Consider the canonical form (3.3.3). If all the $\bar{b}_i > 0$ and at least one $\bar{c}_j < 0$, then a new basic feasible solution can be constructed by increasing the corresponding non-basic variables whose $\bar{c}_j < 0$ (while keeping the other non-basic variables at value zero) and adjusting the values of the basic variables accordingly. The new value of the objective function will generally be lower than $F(\bar{x}) = \bar{F}_o$ in (3.3.3).

An empirical rule for choosing one of the non-basic variables, x_s, is

$$c_s = \min \bar{c}_j < 0 \qquad (3.3.4)$$

This rule usually leads to fewer iterations than just choosing any $\bar{c}_j < 0$.

Using the canonical form (3.3.3) we construct a new solution in which x_s assumes some positive value. The values of the other non-basic variables are still equal to zero while the values of the basic variables, including F(x), are adjusted to account for the increase in x_s:

$$x_i = \bar{b}_i - a_{is}x_s \qquad i = 1,\dots,m \qquad (3.3.5)$$

$$F(x) = \bar{F}_o - \bar{c}_s x_s \qquad (3.3.6)$$

Since $\bar{c}_s$ has been chosen negative, it follows that the value of x_s should be made as large as possible in order to make F(x) as small as possible (for a minimization problem). In fact, if all $a_{is} \leq 0$, x_s can be made arbitrarily large, establishing the following:

Theorem 3.5.

If in the canonical system, for some x_s, all $a_{is} \leq 0$ and $\bar{c}_s \leq 0$, then a class of feasible solutions can be constructed whose $F(\bar{x})$ values have no lower bound.

If, on the other hand, at least one $\bar{a}_{is} > 0$, then there is a limit on the largest value that x_s can attain. For example, beyond the value of $x_s = (\bar{b}_i/\bar{a}_{is})$, the value of x_i (the basic variables in (3.3.5)) becomes negative.

If $\bar{a}_{is} > 0$ for several i , then the smallest of such ratios, denoted by i = r, will determine the largest value of x_s possible, such that all values of x_i in (3.3.5) remain non-negative.

Suppose $x_s^* = \max x_s$ possible; then

$$x_s^* = \frac{\bar{b}r}{\bar{a}_{rs}} = \min_{\bar{a}_{is}>0} \left\{\frac{\bar{b}i}{a_{is}}\right\} \geq 0 \qquad (3.3.7)$$

with $\bar{a}_{rs} > 0$. In case of a tie, r may be chosen arbitrarily; e.g. choose amongst the tied variables, the one with the smallest subscript; or choose one of the tied variables at random.

If the solution is degenerate (i.e. one or more $\bar{b}_i = 0$), then it is clear by (3.3.7) that for some $\bar{a}_{is}>0$, the corresponding value of $\bar{b}_i$ of the basic variable is zero; so that no increase in x_s is possible that will maintain the values of the basic variables non-negative. Hence there will be no decrease in $F(\bar{x})$. Special methods for handling degenerate cases are, therefore, required.

Several approaches have been proposed. One is the perturbation method of Charnes. The other scheme involves lexicographical ordering of vectors. A detailed discussion of those methods can be found in [41]

However, if the solution is non-degenerate, we have the following result:

Theorem 3.6

If for some s, $\bar{c}_s < 0$ and at least one $\bar{a}_{is} > 0$ then from a non-degenerate basic feasible solution a new basic feasible solution can be constructed with a lower value of $F(\bar{x})$.

The new basic feasible solution is then tested for optimality. If it fails, new variables (non-basic) are chosen by criterion (3.3.4). The process is repeated until (after a finite number of iterations) it terminates in either:

(a) a class of feasible solutions for which $F(\bar{x}) \rightarrow -\infty$: (Unbounded solutions);

or (b) an optimal basic feasible solution (all $\bar{c}_j > 0$).

3.3.3. First Feasible Solution:

In practice, a number of problems often have a starting feasible canonical form. For other problems, however, this is not the case; e.g. when the model does not have slack variables for some equations, or when the coefficients of some slack variables are negative. In the latter situation, a set of artificial variables are introduced into the standard form of the linear programming model:

$$\sum_{j=1}^{N} a_{ij}x_j \; + \sum_{i,j=N+1}^{N+M} \delta_{ij}x_j \qquad i = 1,\dots,m \qquad (3.3.8)$$

$x_j \geq 0$ $j = 1,\dots,N+M$; where $(x_{N+1}, x_{N+2},\dots,x_{N+M})$ are the artificial variables. The resulting auxilliary model (3.3.8) is in canonical form, and the simplex method can now be employed.

The method of providing an initial feasible solution is usually referred to as Phase I.

Note, however, that any solution to the new problem (3.3.8) is not a solution to the original problem (3.3.1) and (3.3.2), unless all the artificial variables are zero. Since all the artificial variables must be non-negative, it is sufficient to make their sum equal to zero.

The calculations are, therefore, started by minimizing the sum of the artificial variables:[55]

$$\sum_{j=N+1}^{N+M} x_j \quad = \quad w \qquad\qquad (3.3.9)$$

subject to $x_j \geq 0$

In practice, the equivalent form of (3.3.9):

$$w = \sum_{j=1}^{N} d_j x_j + \sum_{j=1}^{M} b_j \qquad x_j \geq 0. \qquad (3.3.10)$$

is minimized, where $d_j = - \sum_{j=1}^{N} a_{ij} \quad i = 1,\ldots,m$ (3.3.10b)

Equation (3.3.9) is called the sum of infeasibilities. If this sum is reduced to zero, then there is a genuine first feasible solution to the original problem; and the original objective function can be improved by applying the simplex steps. This then is the Phase II of the algorithm.

If at the end of Phase I, the sum of infeasibilities is positive, the original problem is infeasible.

A detailed procedure involved in both Phase I and Phase II is given in Ref[15]

3.4. Revised Simplex Method: [5,15]

In the simplex method, a large number of computations are done and recorded in the tableau. Some of this information is not required: in fact only the modified cost row, the modified requirement vector ($\bar{b}_i$) and the column corresponding to the variable entering the basic set play any role in the decision process.

The essence of the revised simplex method is to use simplex multipliers and the inverse of the basis to determine, directly from the original equations, all the necessary information for the decision at hand.

Definition:

Multipliers π_i, i = 1,...,m are called simplex multipliers relative to the F($\bar{x}$) equation, if multiplying the first equation of (3.3.8) by π_1, the second equation by π_2,..., the mth equation by π_m, and subtracting their sum from the F(x) equation eliminates the basic variables.

A set of multipliers relative to the auxilliary equation, w, is denoted by σ_i i = 1,...,m.

The multipliers display the following characteristics:

Theorem 3.7.

The simplex multipliers are unique and are equal to the negative of the coefficients of the artificial variables of the F($\bar{x}$) and w-equation of the canonical form (3.4.1).

Consider the following system of equations (≡4.3.8) after cycle 0.

$$\begin{array}{lccccccccccl}
 & a_{11}x_1 & + & a_{12}x_2 & + \dots + & a_{1n}x_n & + & x_{n+1} & = & & & = b_1 \\
 & a_{21}x_1 & + & a_{22}x_2 & + \dots + & a_{2n}x_n & & & +x_{n+2} & & & = b_2 \\
 & \cdot & \cdot & \cdot & \cdot\;\cdot\;\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
 & a_{m1}x_1 & + & a_{m2}x_2 & + \dots + & a_{mn}x_n & & & & +x_{n+m} & & = b_m \\
(3.4.1) & c_1x_1 & + & c_2x_2 & + \dots + & c_nx_n & & & & & -F & = 0 \\
 & d_1x_1 & + & c_2x_2 & + \dots + & d_nx_n & & & & & -w & = -w_o
\end{array}$$

where w is given by equation (3.3.9) and d_i by (3.3.10b)

$w_o = \sum_{i=1}^{m} b_i$.

The problem is to find $x_j \geq 0 \quad j = 1,\dots,n$ satisfying (3.4.1) such that $w = 0$ and F is minimized. For the standard simplex method, after the K^{th} cycle, the basic variables may be: $x_1', x_2', \dots, x_m', -F, -w$ with the basic feasible solution given by $x_1'=\bar{b}_1,\ x_2'=\bar{b}_2, \dots, x_m'=\bar{b}_m$; $x_j = 0$ otherwise :

basic var.	Admissible variables $x_1 \quad \dots \quad x_n$	Artificial variables $x_{n+1}, x_{n+2}, \dots x_{n+m}$		Constant terms
x_1'	$\bar{a}_{11} \dots \bar{a}_{1s} \dots \bar{a}_{1n}$	$\bar{a}_{1n+1}, \dots \bar{a}_{1n+m}$	$=$	$\bar{b}_1$
x_2'	$\vdots$	$\vdots$		$\vdots$
$\cdot$				
$\cdot$	$a_{r1} \dots \bar{a}_{rs} \dots \bar{a}_{rn}$	$\bar{a}_{rn+1} \dots \bar{a}_{rn+m}$	$=$	$\bar{b}_r$
$\cdot$	$\vdots$		$\vdots$	
x_n'	$\bar{a}_{m1} \dots \bar{a}_{ms} \dots \bar{a}_{mn}$	$\bar{a}_{mn+1} \dots \bar{a}_{mn+m}$	$=$	$\bar{b}_m$
$-F$	$\bar{c}_1 \dots \bar{c}_s \dots \bar{c}_n$	$\bar{c}_{n+1} \dots \bar{c}_{n+m}$ $-w$	$=$	$-\bar{F}_o$
$-w$	$\bar{d}_1 \dots \bar{d}_s \dots \bar{d}_n$	$\bar{d}_{n+1} \dots \bar{d}_{n+m}$ $-w$	$=$	$-w_o$

(3.4. 2)

For the revised simplex method, however the only recorded information from the tableau are the coefficients of artificial variables, the constant terms $\bar{b}_i$ and the names of the corresponding basic variables. During the cycle, part of the missing data from the tableau is generated as required directly from cycle 0 (3.4.1). These involve values of $\bar{c}_j$, $\bar{d}_j$ $j = 1,\dots,n$ and the values of the column $j = s$.

Since the first m equations of (3.4.1.) are in canonical form with respect to $x_{n+1}, x_{n+2}, \ldots, x_{n+m}, -F, -w$ and the system of equations (3.4. 2) is in canonical form with respect to $x_1', x_2', \ldots, x_m', -F, -w$ it follows that:

If the basis $\bar{B}$ is the coefficient matrix of $x_1', \ldots x_m', -F, -w$ in (3.4.1.) then its inverse, $\bar{B}'$ is the coefficient matrix of $x_{n+1}, x_{n+2}, \ldots x_{n+m}, -F, -w$ in (3.4. 2).

An element in a given row and column of (3.4. 2) can, therefore, be generated from (3.4.1) by forming a scalar product of the corresponding row in the inverse and corresponding column of (3.4.1). Thus $\bar{a}_{ij}$ can be generated for say j = s by forming the scalar product of the ith row of the inverse $\bar{B}^{-1}$ by the jth column of (3.4.1):

$$\bar{a}_{ij} = \beta_{i1}a_{1j} + \beta_{i2}a_{2j} + \ldots + \beta_{1m}a_{mj} \qquad (3.4.\ 3)$$

where β_{ij} are the elements of $\bar{B}'$ and are given by

$$\beta_{ij} = \bar{a}_{i,n+j} \qquad (i,j = 1,\ldots m) \qquad (3.4.\ 4)$$

The $\bar{c}_j$ and $\bar{d}_j$ are generated by the scalar product of the F and w row of the inverse, $\bar{B}'$ with the jth column of (3.4.1):

$$\bar{c}_j = c_j - (\pi_1 a_{1j} + \pi_2 a_{2j} + \ldots + \pi_m a_{mj}) \quad (3.4.\ 5)$$

$$\bar{d}_j = d_j - (\sigma_1 a_{1j} + \sigma_2 a_{2j} + \ldots + \sigma_m a_{mj}) \quad (3.4.\ 6)$$

where $\pi_K = -\bar{c}_{n+K} \qquad K = 1,\ldots m \qquad (3.4.\ 7)$

and $\sigma_K = -\bar{d}_{n+K} \qquad (3.4.\ 8)$

(i.e. 3.4. 7 π_K and σ_K are coefficients of artificial variables in the F and w equations of 3.4..2).

Finally, $\bar{b}_i$, $\bar{F}_o$ and $\bar{w}_o$ can be generated by the relation

$$\bar{b}_i = \beta_i b_1 + \beta_{i2} b_2 + \ldots + \beta_{im} b_m \quad (3.4.\ 9)$$

$$\bar{F}_o = \pi_1 b_1 + \pi_2 b_2 + \ldots + \pi_m b_m \quad (3.4.10)$$

$$\bar{w}_o = \sigma_1 b_1 + \sigma_2 b_2 + \ldots + \sigma_m b_m \quad (3.4.11)$$

3.4.1. Advantages of the Revised Simplex Method

1) In the standard method, the complete problem has to be up-dated. This involves (m+1)(n-m+1) operations. In the revised simplex method the up-dating involves (m+1)(m+1) operations. This can result in considerable savings in both time and storage capacity, especially for situations where n is much greater than m and $\bar{A}$ is a sparse matrix- which is usually the case in practice.

2) The revised simplex method is more flexible; e.g. one does not have to complete the pricing operation, for every iteration: instead one can use multiple-column-selection techniques in which one uses the pricing operation to select a number of the most promising columns, up-date them and then do a few steps of the standard method on the resulting small sub-problem.

3.4.2. Product Form of the Revised Simplex Method:

With the product form, the inverse of the original matrix, $\bar{\bar{B}}'$, is not recorded explicitly; rather, it is represented as a product of elementary matrices. Each of these matrices represents the effect of a single pivotal operation.

The effect of pivoting in the rth row is to premultiply $\bar{\bar{B}}^{-1}$ by a matrix that is unit except for the rth column, which is computed from the up-dated pivotal column in the tableau. Such an elementary matrix is generally referred to as an E-matrix.

Instead of recording the entire E matrix, one simply records the column number, r, and the non-zero elements in it; the remaining unit columns being understood. These unit matrices are so taken for granted that the elementary matrices are often referred to as vectors, specifically η-vectors.

The above operation is equivalent to storing the entire inverse of the matrix. It clearly reduces computer storage requirement, and is, in fact, the version that is largely used for digital computations.

3.5. Variants of the Simplex Method[15]

This section contains a summary of certain modifications to the simplex method that were developed in order to take advantage of situations where an infeasible basic solution to the primal problem is available. Examples include cases where a set of problems differ from one another only with regard to their requirement vectors or the cost factors. In such situations, it may be convenient to omit Phase I (if this were possible) and to use the optimal solution of one problem as the initial solution of the other. Some of these methods are discussed below.

3.5.1. Dual-Simplex Method

The method was developed to handle a class parametric linear programming problem; i.e. cases where a new problem differs from the original one in the values of the requirement vector ($\bar{b}_i$) only. The optimal basis of the original problem still prices-out optimally ($\bar{c}_j \geq 0$) for the second problem. However, the associated solution may not be feasible.

However, the said optimal pricing out implies that the solution to the dual problem is feasible. For this situation Lemke[15] developed the Dual-Simplex method, as a variant of the standard simplex algorithm. It operates with the same tableau as the primal method; however the $\bar{c}_j \geq 0$ from iteration to iteration (instead of $\bar{b}_i \geq 0$).

If all the $\bar{b}_i \geq 0$, then the associated problem is optimal as well as feasible. If not, a pivot row, r, is chosen such that

$$\bar{b}_r = \text{Min } \bar{b}_i > 0 \qquad (3.5.1)$$

and the pivot column, s, is chosen such that

$$\bar{c}_s/-\bar{a}_{rs} \quad = \quad \text{Min}\,\bar{c}_j/-\bar{a}_{rj} \quad (\bar{a}_{rj} < 0) \qquad (3.5.2)$$

Clearly if all $\bar{a}_{rj} \geq 0$, then the primal has no feasible solution.

When the Dual-Simplex method is viewed in terms of the primal variables, one decides first which basic variables to drop, and then decides which non-basic variables to introduce.

3.5.2. The Primal Dual Method

Apart from the Dual-Simplex method, several other computationally similar variants of the standard primal simplex algorithm have been developed; for example, the method of Leading Variables, Parametric Linear Programming and the Primal-Dual methods.

The Primal-Dual method, developed by Ford and Fulkerson for solving transportation problems and later extended to handle the general linear programming problem will be discussed in this section. Our discussion will concentrate on the generalized form of the method rather than the original algorithm of Ford and Fulkerson.

Any feasible solution to the original problem may be used to initiate the method, which is based on the fact that corresponding to any dual solution is an associated restricted primal requiring optimization.

When the solution to the restricted problem has been found, an improved solution to the dual system can be obtained. This in turn gives rise to a restricted primal problem which has to be optimized. The process is continued until, after a finite number of improvements, an optimal solution to the original unrestriced problem is obtained.

As with the Dual-Simplex method, the entire process may be considered as a way of starting an infeasible basic solution and using a feasible solution to the dual already at hand to decrease the infeasibility form of the primal in such a manner that when a feasible solution is found, it will already be optimal.

The initial canonical form of the Primal-Dual method is the same as for Phase I of the standard simplex (3.3.11). It is assumed that a feasible solution to the dual is available; and that by applying the associated multipliers and summing, the c_j have been adjusted before augmentation by artificial variables. So that now

$$c_j \geq 0 \qquad (j = 1,\ldots,n) \quad (3.5.3)$$

The problem is to find $x_j \geq 0$, $w = 0$ and Min. $F(x)$ satisfying (3.3.11). Suppose that after cycle t the tableau is as shown on page 57. Artificial variables not in the basic set are dropped from the system.

Theorem 3.8.

If $\bar{w}_o = 0$ then the basic solution is optimal. We note that when $\bar{w}_o = 0$ the artificial variables all have zero values in the basic solution. When these are dropped, the feasible solution has $\bar{c}_j = 0$ for $\bar{x}_j > 0$, which fulfils the condition of optimality.

Cycle t: Tableau of the Primal-Dual Algorithm

basis

x_1	…	x_q	x_{q+1}	…	x_m	x_{m+1}	…	x_{m+p}	x_{m+p+1}	…	x_{n+q}		π	σ
1						$\bar{a}_{1\ m+1}$	…	$\bar{a}_{1\ m+p}$	.	.	$.\bar{a}_{1\ n+q}$	$b_1 \geq 0$	π_1	σ_1
.		1												
			1	. . .										
					1	$\bar{a}_{m\ m+1}$	…	$\bar{a}_{m\ m+p}$	.	.	$.\bar{a}_{m\ n+q}$	$b_m \geq 0$	π_m	σ_m
0	…	0	0	…	0	d_{m+1}	…	d_{m+p}	d_{m+p+1}	…	d_{n+q}	$w=\bar{w}_o$		
*	…	*		…	*	0	…	0	$\bar{c}_{m+p+1}$	…	$\bar{c}_{n+q}$	$F-F_o$		

artifical $\quad$ $\bar{c}_j = 0$ $\quad$ $\bar{c}_j > 0$

restricted primal

Step 1. Minimizing Infeasibilities in the Restricted Primal

It is assumed that at cycle t there are one or more x_j with $\bar{c}_j = 0$. These x_j together with the basic variables constitute the restricted primal problem.

Using only these variables for first choice, the simplex method is applied to minimize w. The artificial variables which have become non-basic are usually dropped from the system.

Step 2. The process is terminated when either (a) $w = 0$ in which case the solution is optimal; or (b) if $w > 0$ and all $\bar{d}_j \geq 0$ $(j = 1,\ldots,n)$ thus implying the non-existence of a feasible solution. If, on the other hand, $w > 0$ and one or more $\bar{d}_j < 0$ then proceed to step 3.

Step 3. Improving the Dual Solution:

This is achieved by means of new multipliers:

$$\pi_i^* = \pi_i + K\sigma_i \qquad (3.5.4)$$

which generate non-negative cost factors

$$\bar{c}_j^* = \bar{c}_j + K\bar{d}_j \qquad (3.5.5)$$

where K is positive and is defined by:

$$K = \bar{c}_s/(-\bar{d}_s) = \min_{d_j<0} \bar{c}_j/(-\bar{d}_j) > 0 \qquad (3.5.6)$$

The new restricted primal is obtained using all the basic variables and all the non-basic variables whose cost factors $\bar{c}_j^* = 0$.

For further discussions on the Primal-Dual method see Ref. 15.

3.5.3. Other Variations

In contrast to the above algorithms Gass and Saaty have developed a method for solving problems with constant requirement vector but varying coefficients.[32] This method, too, relies heavily on the standard simplex algorithm. It has been used quite extensively for the solution of parametric linear problems with varying cost coefficients. Details of the algorithm are contained in reference[32]

In some cases the problems may differ by more than just the constant terms. Consequently, the old basis may not price out optimally. This may lead to neither the basic solution nor the dual solution generated by its multipliers remaining feasible. In such a situation a composite algorithm of which the Self-Dual algorithm is an example is employed.[15]

3.6. Logarithmic Potential Method:[30]

This was developed with special emphasis on macroeconomic planning. The main motivation was to develop a method that would involve less work than the simplex method*

As we have indicated (Section 4.2), the simplex method first seeks some corner on the feasible region, and then moves systematically along the edges from corner to corner, until the optimal corner is reached.

A major difficulty with the simplex procedure is that in the course of the elimination process, one

* Generally, with regard to desk computation. Most of this was done in early 1950's when the high speed large storage digital computer had not been firmly established.

may, from time to time, reach a multiply determined corner (i.e. obtain a degenerate solution). Possibly an optimal solution may itself be degenerate; still further laborious test would be required to ascertain whether it is actually optimal or not.

The logarithmic potential method, however, works systematically in the interior of the feasible region. It uses a logarithmic potential as a means of ensuring that the solution process is always within the feasible region.

The following is a brief summary of the method.

Define the potential

$$PF(x_1,x_2,\ldots,x_m) = \sum_{K=i,\ldots}^{m} \log_e x_K + \sum_{j=m+1}^{n} \log_e x_j \qquad (3.6.1)$$

In other words, the potential is the sum of the logarithms of all the variables (Basic and non-basic). The potential is continuous with continuous first derivatives anywhere in the feasible region; but as any point in the boundary is approached, the potential tends towards $-\infty$.

The potential can be viewed as a function of the basic variables with partial derivatives

$$\nabla PF_K = \frac{\partial PF}{\partial x_K} = \frac{1}{x_K} + \sum_{j=m+1}^{n} a_{jK}/x_j; \quad K = 1,\ldots,m \qquad (3.6.2)$$

where ∇PF is the gradient of the potential.

Consider also the gradient of the objective function ∇F_K where $F_K = c_1x_1 + c_2x_2 + \ldots + c_mx_m$ ($x_i = 0$ for $i = m+1, m+2, \ldots, n$).

These two gradients define two different directions along which one can move in the iteration process. In order to increase the objective function, it is desirable to move in the direction $+\nabla F_K$; but in order to steer away from the boundary, one should move along the direction ∇PF_K.

The optimal solution is obtained through a compromise between the two directions. A detailed procedure of obtaining the compromise is contained in[30] [illegible]

For desk machine computations, logarithmic potential method seems to have been quite successful. However, with the advent of high-speed, large-storage digital computers, the simplex method (in the inverse product form) has so far proved to be the most efficient method for solving linear programming problems.

The importance of the logarithmic potential method lies in its modification and generalization to handle nonlinear programming problems. This aspect has been considered in some detail in Chapter 5 .

CHAPTER 4

INTEGER PROGRAMMING METHODS

4.0. For many practical problems, the solution variables are required to be integers. In situations where the variables are sufficiently large, the resulting solution may very well be rounded off to the nearest integers satisfying the constraints. However, there is a host of problems for which the integer solutions must be very small numbers: often 0, 1 or 2. Any round off may, therefore, give solutions which are far removed from the optimum.

Integer programming is the name given to solving linear programming problems when the variables must take integer values. The methods are classified into pure integer programming when all the variables must be integer; and mixed integer programming when only certain specified variables must be integers.

4.1. The General Integer Programming Problem:

The general integer programming problem may be stated as follows: Find $x_1,\ldots,x_n$ with x_i integer valued for some specific set of indices $i \in J$ such that

$$F(\bar{x}) = \sum_{i=1}^{n} c_i x_i \qquad (4.1.1)$$

is minimized subject to

$$\sum_{j=1}^{n} a_{ij} x_j \geq b_i \qquad i = 1,\ldots,m \quad (4.1.2)$$

An equivalent formulation is: Find $x_1, \ldots, x_n$ with x_j integer valued for some specific set of indices i J such that

$$F(\bar{x}) = a_{oo} + \sum_{j=1}^{n} a_{oj}(-x_j) \qquad (4.1.3)$$

$$\sum_{j=1}^{n} a_{ij}x_j \leq a_{i,o} \qquad i = 1,\ldots,m \qquad (4.1.4)$$

We shall use the latter formulation in the discussions which follow.

It is clear that if J spans the entire x_j range $j = 1,\ldots,n$, (i.e. $J=J_T$) then we have a pure integer programming problem; while if J spans only part of the range (i.e. $J=J_m$) then the problem is of mixed integer programming type. If , on the other hand, the integer requirement is dropped (i.e. $J=J_o$) the problem reduces to a linear programming problem.

4.2. Methods of Solution

Several practical methods for solving integer programming problems have been developed. Some of these will be briefly discussed here. A more detailed exposition can be found in the survey paper by Balanski.[2]

The methods will be discussed along the following general classifications:

a) Cutting-plane methods

b) Primal methods.

c) Mutual Primal-Dual methods.

d) Branch and Bound methods.

e) Dynamic programming algorithms.

a) Cutting-Plane methods

Much of the work in this field has been done by R.E. Gomory. The underlying approach of the cutting-plane methods may be viewed as a process of convexification*. That is, the process of solving (4.1.3) and (4.1.4) isolates a feasible point with the required integer property by making it an extreme point of a new polyhedral convex constraint set at which the linear form (4.1.3) is minimized (maximized). This is achieved by devising new constraints in such a way that a finite number guarantees finding a linear programme whose solution has the required integer property.

The basic tool for the cutting-plane (or the new constraints generation) methods is a dual Simplex method (chapter 3). Many different cutting-plane methods are possible, depending on how the new constraints are generated. Several of these are discussed below.

(i) Pure integer Programming I.

Consider a linear programming problem whose solution variables are required to be all integers. Let

* For other possible points of view seeRef. 2.

the variables be expressed in terms of the non-basic (independent) variables. A typical equation of the linear programme then becomes:

$$x_i = a_{io} + \sum_{j=1}^{n} a_{ij}(-x_j) \qquad (4.2.1)$$

A new non-negative integer variable, x_j' can now be defined. Let $\bar{t} = [\bar{t}]^I + \xi$ where $[\bar{t}]^I$ is the integer part of $\bar{t}$ and $0 \leq \xi \leq 1$. Let $\pi > 0$. Define

$$a_{ij/\pi} = \left[a_{ij/\pi}\right]^I + \xi_{j/\pi} \quad 0 \leq \xi_j \leq \pi$$

$$j = 0,1,\ldots,n \qquad (4.2.2.)$$

Then, after dividing through by π (4.2.1) may be written

$$x_{i/\pi} + (1/_{\pi}) \; \Sigma\, \xi_j x_j = \{ \left[a_{io/\pi}\right]^I + \Sigma\left[a_{ij/\pi}\right]^I(-x_j)\}$$

$$+ \quad \xi_{oi/\pi} \qquad (4.2.3)$$

Now, the left-hand-side of (4.2.3) is non-negative; so does the right-hand-side. However, by definition (4.2.2) $(\xi_{o/\pi}) < 1$ and the expression inside {} in (4.2.3) is integer. Therefore the term inside {} must be non-negative as well as integer; i.e.,

$$x_j' = \left[a_{io/\pi}\right]^I + \Sigma \left[a_{ij/\pi}\right]^I(-x_j) \geq 0 \qquad (4.2.4)$$

is a new constraint. The above type of constraint was developed by R.E. Gomory and is applicable to pure integer programming problems.[2]

(ii) Pure Integer Programming II

Let $\pi = 1$. Then(4.2.3) can be rearranged to give

$$\Sigma\xi_j x_j = \{ \left[a_{io}\right]^I + \Sigma \left[a_{ij}\right]^I (-x_j) - x_i\} + \xi_o$$

(4.2.5)

Again the left-hand-side of (4.2.5) is non-negative and hence the right-hand-side; but $\xi_o < 1$ and the term inside {} is integer. Therefore

$$x_j' = -\xi_o + \Sigma\ \xi_j x_j$$

$$= -\xi_o + \Sigma\ (-\xi_j)(-x_j) \geq 0 \qquad (4.2.6)$$

is a new constraint. This is the constraint developed in Gomory's first cutting-plane method.[2]

(iii) Mixed integer programming

Suppose that in problem (4.1.3) and (4.1.4) only x_i for i J are constrained to be integers. Consider a topical equation (4.1.4) of the linear programme corresponding to a variable which is integer constrained. Then, making the substitution (4.2.2) for j=0 and j J_m and letting π=1, equation (4.2.1) can be rewritten:

$$\sum_{j\in J_m} \xi_j x_j + \sum_{j\in J_m} a_{ij} x_j = \{\left[a_{io}\right]^I + \sum_{j\in J_m} a_{ij}\ (-x_j) - x_i\}$$

$$+ \ \xi_o \qquad (4.2.7)$$

The term inside {} is integer and is either (a) greater than or equal to zero or (b) less than or equal to -1. Thus in case (a)

$$\sum_{j \in J_m} \xi_j x_j + \sum_{j \in J_m} a_{ij} x_j \geq \xi_o \qquad (4.2.8)$$

and hence

$$\sum_{j \in J_m} \xi_j x_j + \sum_{(j \in J_m, a_{ij} \geq 0)} a_{ij} x_j \geq \xi_o \qquad (4.2.9)$$

In case (b)

$$\sum_{j \in J_m} \xi_j x_j + \sum_{j \in J_m} a_{ij} x_j \leq -1 + \xi_o \qquad (4.2.10)$$

and hence

$$\sum_{(j \in J_m,\, a_{ij} < 0)} a_{ij} x_j \geq \xi_o - 1 \qquad (4.2.11)$$

or

$$\sum_{(j \in J_m,\, a_j < 0)} (\xi_o/(1-\xi_o)(-a_{ij})x_j \geq \xi_o \qquad (4.2.12)$$

Since the coefficients of all the variables in the last inequalities of (4.2.8) and (4.2.10) are non-negative, we have:

$$x_i' = -\xi_o + \sum_{j \in J_m} (-\xi_j)*(-x_j) + \sum_{(j \in J_m, a_{ij} < 0)} (-a_{ij})(-x_j)$$

$$+ \sum_{(j \in J_m, a_{ij} < 0)} (\xi_o \, a_{ij}/(1- {}_o))(-x_j) \geq 0$$

(4.2.13)

where x_i' is a new constraint; also developed by Gomory.[2]

Using the above new constraints Gomory proposed two algorithms, two for the pure problem and one for the mixed problem. Details of these may be found in and also in [2]

The algorithms converge to an optimal solution with the required integer property in a finite number of steps - if such a solution exists. Unfortunately, however, the codes have proved erratic in performance: sometimes solving large problems (e.g. a thousand variables in 100 constraints), while at others breaking down for no apparant reason.

Several new codes now perform much better; e.g. the one by Haldi and Isaacson. [40]

(b) Primal Methods

As we have mentioned above, cutting-plane methods solve the integer programming problem by the dual simplex method. Primal methods have also been developed for solving pure integer programming problems along the lines developed above.

The tableau is kept integral by pivoting only on the unit elements. The variable to be introduced in the basis is chosen as one with a negative reduced cost (Chapter 3). If however, the variables have a non-unit coefficient in the row in which one would naturally pivot, then the following transformation is introduced: if the natural pivotal row is:

$$x_i = a_{io} + \Sigma\, a_{ij}(-x_j) \qquad (4.2.14)$$

and if

$$\frac{a_{ij}}{\pi} = \left[\frac{a_{ij}}{\pi}\right]^I + \frac{\xi_{ij}}{\pi} \qquad 0 \leq \xi_{ij} < \pi > 1 \qquad (4.2.15)$$

where, as in the case above, $\left[\frac{a_{ij}}{\pi}\right]^I$ is the integral part of $a_{ij/\pi}$. The above transformation gives rise to a new integral and non-negative variable.

$$x_i' = \left[\frac{a_{io}}{\pi}\right]^I + \Sigma \left[\frac{a_{ij}}{\pi}\right]^I (-x_j) \qquad (4.2.16)$$

satisfying the constraint that $x_i' \geq 0$.

With the new conditions (4.1.2 and $x_i' \geq 0$) satisfied, it becomes possible to generate a suitable pivotal row for any π such that

$$\text{Max } \{ \ a_{is/2}, \ a_{io/} \left[(a_{io}/a_{is}) + 1\right] \} < \pi \leq a_{is} \qquad (4.2.17)$$

where x_s is the variable being made basic and (a_{io}/a_{is}) is the value of x_s at the nezt trial solution.

Note that the above approach is essentially similar to the "cutting-plane" methods, except that the computation algorithm is the primal simplex method rather than the dual simplex method. L.M. Isaacson[42] has written a computer code for the special case when $\pi = 1$. The code uses the "revised Simplex" method or the inverse method for carrying out computations. The code can handle up to 300 variables in 120 constraints. It has achieved successes on a number of test problems; and is, in fact, the code used for solving a network design problem in Chapter 10. The author's computation experience with this method is that its performance is also erratic.

(c) Mutual Primal-Dual Methods:

These methods were developed by M.L. Balanski and R.E. Gomory[3] and independently by G.W. Graves.[38] Grave's version was specifically developed for the application to the general mixed integer programming problems.

The essential features of both methods are the elimination of "artificial variables" for the initial feasible solutions and the use of a nested sequence of contracting alternate primal and dual problems to cope with degeneracy.

In addition to the above common features, Graves algorithm provides explicitly a unified treatment of mixed systems containing both positive and free (non-restricted) variables as well as both inequalities and equations. Furthermore, the algorithm tends quickly to dispose of free variables and of equations thus effectively reducing the size of the problem under consideration.

The author has only recently learned of the algorithm and has, therefore, been unable to programme and apply it to any specific problems. But its computational efficacy seems to be very promising .

(d) Branch and Bound Methods.

In 1963 Little et al.[54] developed a method which they successfully applied to solve travelling salesman-type problems. They called it a 'branch and bound' algorithm.

Three years earlier, a similar method had been proposed by Lang and Doing [19] as suitable for solving both pure and mixed integer programming problems.

The procedure involves the solution of a series of linear programming problems in which upper and lower bounds are imposed on all integer variables. Initially the bounds are placed far enough apart to be sure to include the optimal solution (with integer variables). In the course of the enumeration of each sub-problem, a current best known solution is stored. The process is continued until all the possibilities have been exhausted. One of these solutions is an optimal solution.

A detailed account of the general procedure, together with Lang and Doing's version may be found in reference [2] So far no computational experience with Lang and Doing's method has been reported.

Several variations of the general branch and bound procedures have since been proposed and successfully used to solve a variety of problems.[2,5] In this section, we shall consider the special versions due to Glover [34] and Balas [4]. Their version falls under the general sub-classification of <u>Partial Enumeration methods</u>. By this method one considers only trial solutions where all variables must be integers, but in which the constraints may not all be satisfied.

A major advantage of the method is that not all solutions are explicitly enumerated; rather they are implicitly enumerated by considering groups of solutions together. Consequently, substantial saving in storage requirements is assured.

Definition 4.1.: A partial solution, S, is an assignment of binary values (0 or 1) to a subset of n variables.

Any variables not assigned a value of S is called free. In the discussions that ensue the following notational convention will be adopted: the symbol j denotes that $x_j = 1$ and -j denotes that $x_j = 0$. For example, if n = 5 and S = {3,5-2} then $x_3 = 1$, $x_5 = 1$, $x_2 = 0$ while x_1 and x_4 are free. The order in which the elements of S are written represents the order in which the elements are generated.

Definition 4.2: A completion of a partial solution, S is a solution that is determined by S together with the binary specification of the values of the free variables.

Thus in the above example there are four possible completions of S: (0,0,1,0,1),(0,0,1,1,1)(1,0,1,0,1) and (1,0,1,1,1). It is clear from the above definition that a partial solution S with $\bar{s}$ elements, say, determines a group of $2^{n-\bar{s}}$ different completions or solutions. For the special case when there are no free variables, there is only one completion of S: the trivial one determined by S itself.

Implicit enumeration involves generating a sequence of partial solutions and simultaneously considering all completions of each.

As the calculations proceed, feasible solutions are discovered from time to time. The best one yet is stored as an incumbent.

It may happen that for a given partial solution S, a best feasible completion of S can be determined. If such a solution is better than the best known feasible solution, then it replaces the latter in store.

On the other hand, it may be established that S has no feasible completion which is better than the incumbent.

In either case, we say that we can fathom S. All completions of a fathomed S have been implicitly enumberated in the sense that they can be excluded from further consideration, the only exception being a best feasible completion of S that unseats the incumbent.

Definition 4.3: A partial solution for which no completion in the sequence ever duplicates a completion of the previous partial solution that was fathomed is called a non-redundant partial solution.

The main steps for generating a non-redundant sequence $\underline{s}^v$ are illustrated in Fig. 4.1.[33] The scheme terminates only after all 2^n solutions have been (implicitly) enumerated. The question of how to fathom a given S is answered by the illustration in Fig. 4.2. This is the version due to Balas.[4]

For Fig. 4,1. we start with $S^o = \bar{0}$, indicating an empty set. If S^o can be fathomed, the process is terminated: Either there is no feasible solution or there is one and the best feasible solution can be found.

If S^o cannot be fathomed, it is augmented by specifying a binary value for the additional free variables at a time, each trying to fathom the resulting partial solution until, at some trial K_1, S^{K1} is fathomed.

In order to be sure of having enough information in the future so as to enable us to know when all 2^n solutions have been accounted for, S^{K_1} is stored. Furthermore, to be sure of having non-redundant sequence S^v starting from $v = K_1 + 1$ on, we must have in all future S^v at least one element complementary to S^{K_1}. Once S^{K1} has been stored, the condition for non-redundancy of S^{K1+1} may be accounted for by taking S^{K_1+1} to exactly S^{K_1} with the latter's element multiplied by -1 and underlined.

If S^{K_1+1} can be fathomed, then all completions of S^{K_1}, without its last element, can be enumerated; so that we can 'forget' the fathoming of S^{K_1} and S^{K_1+1} and 'remember' that only S^{K_1} with its last element has been fathomed. For example if $K_1 = 3$ and $S^3 = \{3,5,-2\}$ is fathomed and then $S^4 = \{3,5,\underline{2}\}$ is fathomed, then all completions of $\{3,5\}$ have been accounted for. So that fathoming S^3 and S^4 is equivalent to fathoming $\{3,5\}$. Such procedures lead to economies in the storage requirements.

By the same token, S^{K_1+2} is chosen as S^{K_1} less the latter's last element with its next to last element multiplied by -1 and underlined. In the above example $S^5 = \{3,-\underline{5}\}$.

If on the other hand S^{K_1+1} cannot be fathomed, then it is augmented by a binary value for one additional free variable at a time, each time trying to fathom the resulting partial solution until at some latter trial K_2, S^{K_2} is fathomed.

When S^{K_2} has been fathomed it too (in addition to S^{K_1}) must be stored. Consequently, every succeeding S^v contains elements that are complements to S^{K_1} and S^{K_2} respectively.

Thereafter S^{K_2+1} is taken as S^{K_2} with the latter's last element complement and underlined; e.g. if, say, S^4 could not be fathomed and S^5 were taken as $\{3,5,\underline{2},1\}$; i.e. $K_2 = 5$; then $S^6 = \{3,5,\underline{2},-\underline{1}\}$. The procedure is repeated until an optimal solution is found - if one exists.

A particularization of the above procedure, based on Bala's algorithm is illustrated in Fig. 4.2, where the details of the mechanisms of Steps 1 and 2 of Fig. 4.1. are also derived.

Beginning with Step 1, the task is to fathom the current partial solution, S. This can be done by doing either of the following:

a) finding the best feasible completion of S.

b) determining that no feasible completion of S has a lower value of the objective function than the incumbent.

<u>Definition 4.4</u>: A bounded solution is one for which an upper bound v_j is available for each variable.

The general strategy of fathoming S involves quite simple computations. Associated with S is a best completion $x^{\bar{s}}$ of S. Such a completion is arrived at by selecting $x^{\bar{s}}$ equal to either 0 or 1, depending on whether $\bar{c}_j \geq 0$ or $\bar{c}_j < 0$.

Let $\bar{c}_j < 0$; then $x^{\bar{s}} = 0$. If $x^{\bar{s}}$ is feasible then $x^{\bar{s}}$ is a best feasible completion of S and S is thereby fathomed.

As a first substep of Step 1 (Fig. 4.1) the feasibility of $x^{\bar{s}}$ is tested. As the computations proceed, the incumbent feasible solution gives an upper bound $\bar{z}^*$ on the optimal value of the objective function (4.1.3).

Initially $\bar{z}^*$ is taken equal to ∞.

If the best completion $x^{\bar{s}}$ is not feasible, an attempt is made to establish that no feasible completion of S is better than the incumbent. If this is the case, then it must be impossible to complete S so as to eliminate all the infeasibilities of $x^{\bar{s}}$ and at the same time improve on $\bar{z}^*$.

The said impossibility may be demonstrated along the following lines: consider non-zero binary values only for the variables in

$$T^{s} \equiv \{j \text{ free: } cx^{\bar{s}} + \bar{c}_j < \bar{z}^* . \text{ and } a_{ij} > 0$$

for some i $y_i^{\bar{s}} < 0$ } (4.2.19)

where $y_i^{\bar{s}} = A\bar{x}^{\bar{s}} + \bar{b}$. So that if $T^{\bar{s}}$ is empty (i.e. $\bar{T}^{\bar{s}} \equiv \bar{0}$) then there is no feasible completion of S that is better than the incumbent, and S is fathomed. The same conclusion holds if

$$y_i^{\bar{s}} + \sum_{j\ T^{\bar{s}}} \mathrm{Max}\ \{0,a_{ij}\} < 0 \qquad (4.2.20)$$

for some i $\ni$ $y_i^{\bar{s}} < 0$.

For the augmentation of Step 2. (Fig. 4.1), one choice is to augment S by the variable from $T^{\bar{s}}$ which leaves the least amount of total infeasiblity in the next $x^{\bar{s}}$ in the sense of making

$$\sum_{i=1}^{m} \min\ \{\ y_i^{\bar{s}} + a_{ij},0\} \qquad (4.2.21)$$

an algebraic maximum.

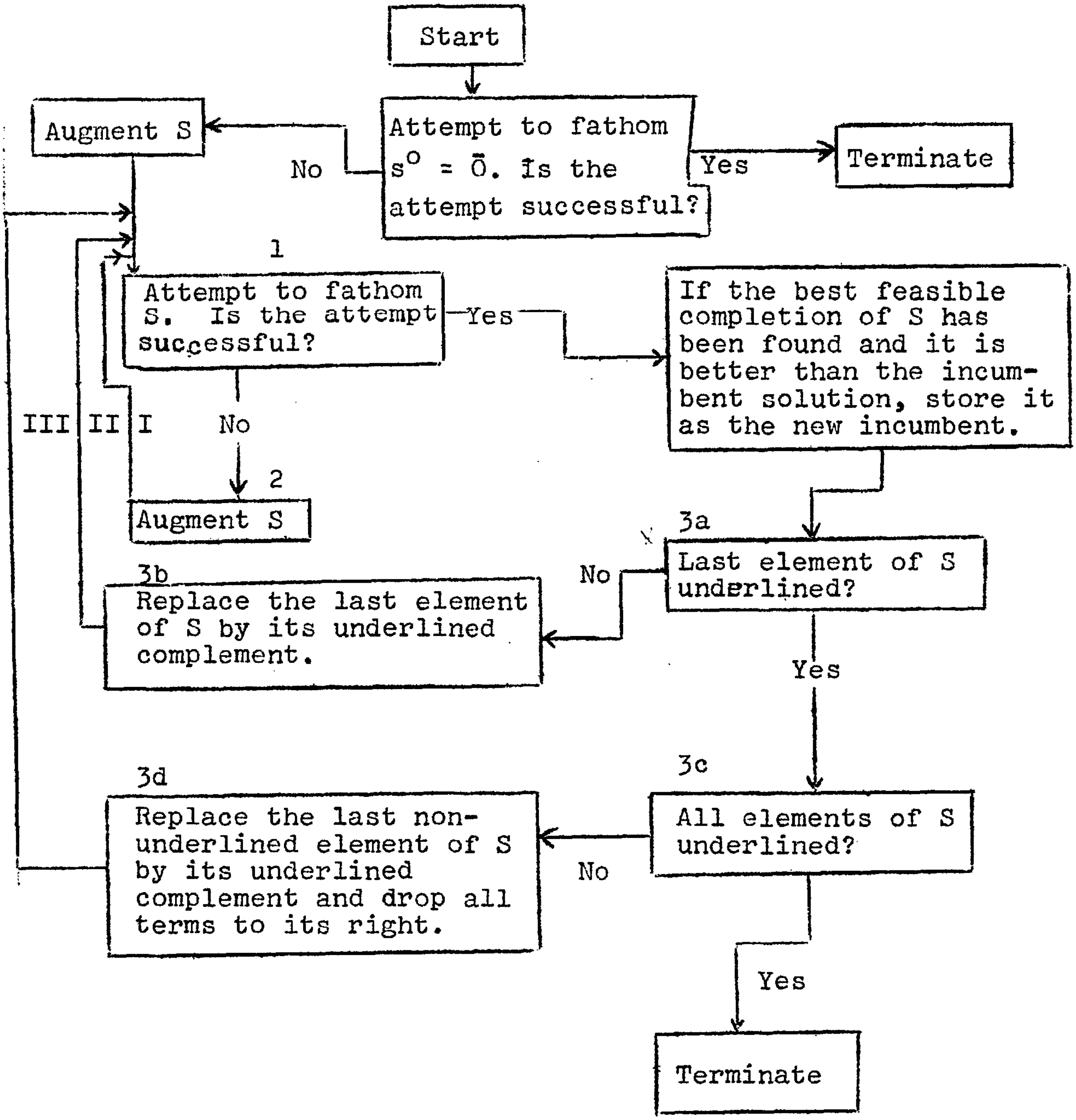

Start
Attempt to fathom $s^o = \bar{0}$. Is the attempt successful?
Yes
Terminate
No
Augment S
1
Attempt to fathom S. Is the attempt successful?
Yes
If the best feasible completion of S has been found and it is better than the incumbent solution, store it as the new incumbent.
III
II
I
No
2
Augment S
3a
Last element of S underlined?
No
3b
Replace the last element of S by its underlined complement.
Yes
3c
All elements of S underlined?
No
3d
Replace the last non-underlined element of S by its underlined complement and drop all terms to its right.
Yes
Terminate

Fig. 4.1. Flow Chart of a General Enumerative Procedure

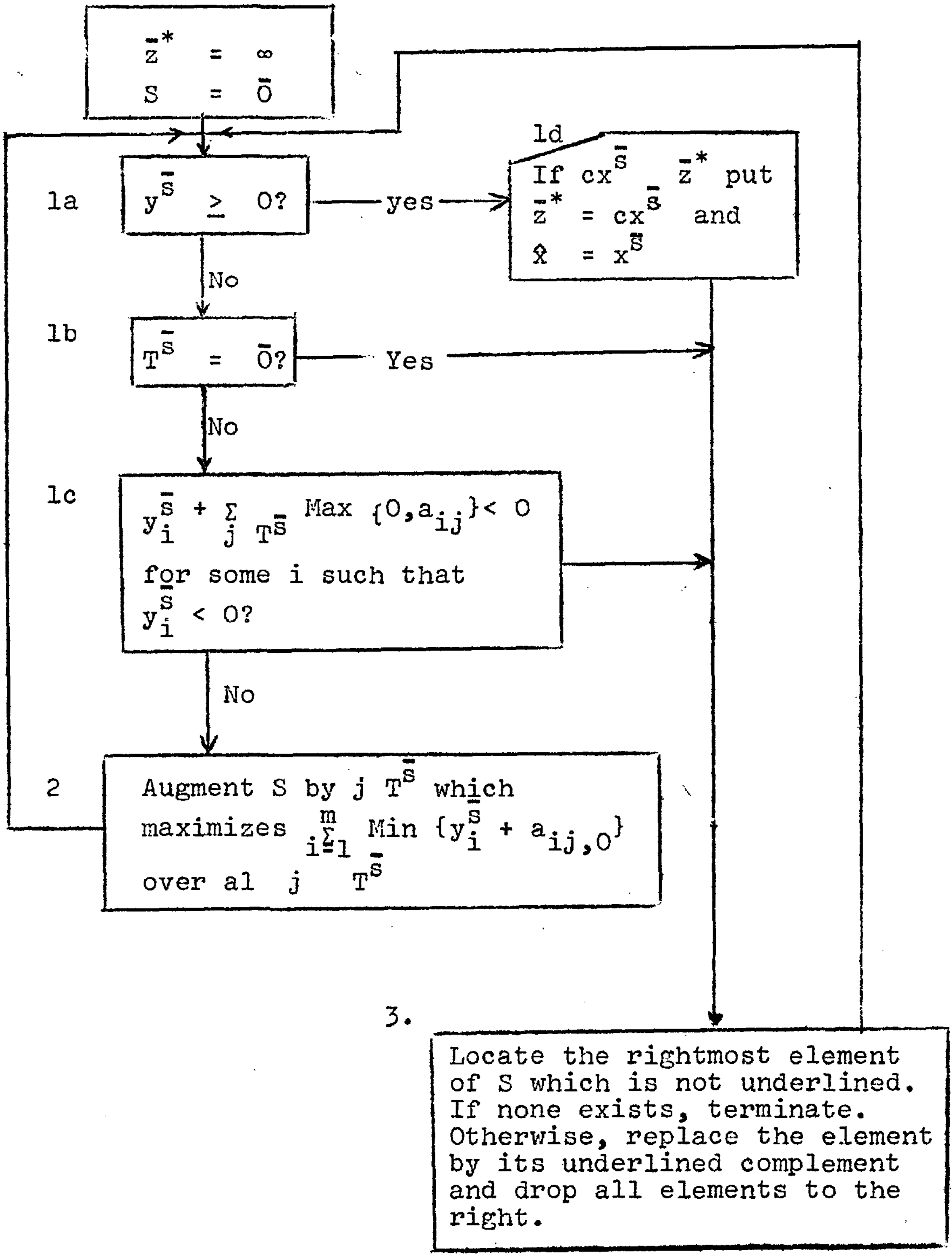

$\bar{z}^* = \infty$
$S = \bar{0}$
1a
$y^{\bar{s}} \geq 0?$
yes
1d
If $cx^{\bar{s}}$ _ $\bar{z}^*$ put
$\bar{z}^* = cx^{\bar{s}}$ and
$\hat{x} = x^{\bar{s}}$
No
1b
$T^{\bar{s}} = \bar{0}?$
Yes
No
1c
$y_i^{\bar{s}} + \sum_{j \, T^{\bar{s}}} \text{Max} \{0, a_{ij}\} < 0$
for some i such that
$y_i^{\bar{s}} < 0?$
No
2
Augment S by j $T^{\bar{s}}$ which
maximizes $\sum_{i=1}^{m} \text{Min} \{y_i^{\bar{s}} + a_{ij}, 0\}$
over al j $T^{\bar{s}}$
3.
Locate the rightmost element
of S which is not underlined.
If none exists, terminate.
Otherwise, replace the element
by its underlined complement
and drop all elements to the
right.

Fig. 4.2. Flow Chart of a Particularization of Fig. 4.1. Based on Bala's Algorithm.

The above details have been incorporated into the procedure of Fig. 4.1. and Fig. 4.2. although the logic of Fig.41. has been arranged so as to give a more compact presentation, the logics of Figs. 4.1. and 4.2. are in fact equivalent.

e. Dynamic Programming Algorithms:

The work in this field was initiated by R.E. Gomory.[36] He showed that if a linear programming problem has been solved, it is relatively easy to solve the corresponding pure integer programming problem, so long as the requirement that the basic variables be non-negative is ignored.

This is done by dynamic programming (Chapter 6). The non-basic variables are considered in turn and the state-space (Chapter 6) consists of the finite group of possible combinations of non-integral parts in the values of the basic variables.

The computational experience so far indicates that the method will not necessarily give the answer to the integer programming problem unless the original values of the basic variables were fairly large

The remaining part of the section is devoted to a very brief discussion of Gomory's method of transforming a given linear integer programming problem into a related group of optimization problems, which can then be solved by dynamic programming. The concept behind this procedure is that an optimal solution to the group problem from which it is derived is optimal for the integer programming problem if the solution is feasible.

Consider an integer programming problem in canonical form: (See Chapter 3)

$$\text{minimize (maximize)} \quad F(\bar{x}) = c^T\bar{x} \qquad (4.2.22)$$

$$\text{subject to} \quad A\bar{x} = \bar{b} \qquad (4.2.23)$$

$$x \geq 0, \ x \text{ integers}$$

Gomory's transformation is as follows: relations (4.2.22) and (4.2.23) are written as

$$\text{minimize} \quad F(\bar{x}) = c_R^T\bar{x}_R + c_{\beta}^T\bar{x}_B \qquad (4.2.24)$$

$$\text{subject to} \quad R\bar{x}_R + B\bar{x}_{\beta} = \bar{b} \qquad (4.2.25)$$

$$\bar{x}_R, \bar{x}_B \ \text{are non-negative integers.}$$

and $\bar{B}$ is the optimal basis for the Linear Programming Problem (4.2.22) in which some of the solution variables are non-integer i.e:

a) B is an mxm matrix

b) $B^{-1}\bar{b} \geq 0$

c) $c_j - c_B B^{-1}a_{ij} \geq 0 \quad (j = 1,2,\dots,m+n)$

(4.2.26)

Condition (c) implies that $c_B^T \bar{x}_B \leq \bar{c}^T x$ for any non-negative vector $\bar{x}$. such that $A\bar{x} = \bar{b}$. B is found by the Simplex Method.

Solving for $\bar{x}$ and substituting into the objective function gives

$$\text{Minimize} \quad F(\bar{x}) = \bar{c}_R^T \bar{x}_R + \bar{c}_{\beta}^T\bar{B}^{-1}(\bar{b}-\bar{R}\bar{x}_R) \qquad (4.2.27)$$

$$\text{subject to} \quad \bar{x}_B = \bar{\bar{B}}^{-1}(\bar{b} - R\bar{x}_R) \qquad (4.2.28)$$

$$\bar{x}_R,\ \bar{x}_B \quad \text{integer.}$$

For the sake of simplicity in mathematical argument (and without loss of generality), let the constant $c_B\bar{B}^{-1}\bar{b}$ be deleted from the objective function (4.2.27). Let us define

$$\hat{c}_R = \bar{c}_R - \bar{c}_B - \bar{c}_B\bar{\bar{B}}^{-1}\bar{b} \qquad (4.2.29)$$

then (4.2.27) and (4.2.28) become

$$\text{Minimise} \quad \hat{F}(\bar{x}) = \hat{c}_R\bar{x}_R \qquad (4.2.30)$$

$$\text{subject to} \quad \bar{x}_B = B^{-1}\bar{b} - B^{-1}\bar{R}\bar{x}_R \qquad (4.2.31)$$

$\bar{x}_R$ and $\bar{x}_B$ have non-negative integer elements

It is now possible to eliminate $\bar{x}_B$ from (4.2.31). First, the explicit non-negative condition on $\bar{x}_B = B^{-1}\bar{b} - B^{-1}\bar{\bar{R}}\bar{x}_R$ is removed. Second, $\bar{x}_B$ is an integer vector in (4.2.31) if and only if $\bar{\bar{B}}^{-1}\bar{b}$ and $B^{-1}\bar{\bar{R}}\bar{x}_R$ differ by an integer vector. Hence the constraint equation $\bar{x}_B = \bar{\bar{B}}^{-1}\bar{b} - \bar{\bar{B}}^{-1}\bar{\bar{R}}\bar{x}_R$ can be replaced by $\bar{\bar{B}}^{-1}\bar{b} = \bar{\bar{B}}^{-1}\bar{\bar{R}}\bar{x}_R$ (modulo 1)

The vector $\bar{\bar{B}}^{-1}\bar{b}$ and the columns of $\bar{\bar{B}}^{-1}\bar{\bar{R}}$ can be replaced by their fractional parts - without loss of generality - since the contribution of their integer parts is 0 (modulo 1).

All the fractions are cleared by multiplying both sides of the constraint equation by $\underline{D} = |\det \bar{B}|$ and at the same time replacing modulo 1 by modulo D. The net result is

$$\text{Minimise} \quad \hat{F}(\bar{x}) \quad = \quad \Sigma_{j=1}^{j=n} \quad c_j x_j \qquad (4.2.32)$$

$$\text{subject to} \quad \Sigma_{j=1}^{n} \quad \alpha_j x_j \quad = \quad \alpha_b \text{ (modulo D)} (4.2.33)$$

with x_j a non-negative integer $j = 1,\ldots,n$, where

$$\alpha_b \quad = \quad \underline{D} \quad \{\bar{\bar{B}}^{-1} b - \left[\bar{\bar{B}}^{-1}\bar{b}\right]^I\} \qquad (4.3.34)$$

$$\alpha_j \quad = \quad \underline{D} \quad \{\bar{\bar{B}}^{-1} a_j - \left[\bar{\bar{B}}^{-1} a_j\right]^I\} \qquad (4.2.35)$$

$$j = 1,\ldots,n.$$

and $[a]^I$ is an integer of part a.

Since $\hat{c}_j \geq 0$, (4.2.32) and (4.2.33) have a solution denoted by

$$\bar{y}_R \quad = \quad (y_1,\ldots,y_n) \qquad (4.2.36)$$

Observe that $(\bar{y}_R, \bar{\bar{B}}^{-1}(\bar{b} - \bar{\bar{R}}\bar{y}_R)$ is an optimal solution to (4.2.22) and (4.2.23) if it is feasible.

The above transformatiors were used by Gomory to obtain a solution to the original integer programming Problem. But as we have already indicated, although the optimal solution $\bar{y}_R$ to the modified problem does have integer values, there is no way of systematically guaranteeing that all the elements of $\bar{y}_R$ will be non-negative (thus satisfying the feasibility conditions of the original problems). Here is an area open for further research work.

J.F. Shapiro[71] has also derived sufficient conditions that can be incorporated into an efficient algorithm for solving the group optimization problem. His algorithm is based on the renewal and knapsack algorithms of reference.[72]74

In a subsequent paper Shapiro extends the method to an algorithm for solving pure integer programming problems.

CHAPTER 5

NON-LINEAR PROGRAMMING METHODS

5.0. In practice, we are quite often faced with optimization problems for which the objective function and/or the set of constraints is non-linear. Computational developments for such types of problems have not reached the degree of efficiency enjoyed by such methods as the "Simplex method" for handling linear and quadratic programming problems.

However, considerable progress in tackling non-linear programming problems has been made: many new techniques have been developed and several old ones modified and perfected.

This chapter discusses the theoretical foundations of some of the major mathematical programming techniques that have been developed to solve a large variety of non-linear optimization problems.

Emphasis is placed on the general concepts underlying each method; and, wherever possible, the unity that exists amongst the seemingly completely different approaches is highlighted. Relative merits and limitations of some of the methods are discussed at length.

5.1. Direct and Indirect Methods:

Broadly speaking, optimization methods can be divided into two classes: direct and indirect methods.

Direct methods start at an arbitrary point and proceed stepwise towards the optimum through direct comparison of the values of the function at two or more points. The point which gives an improved value of the function is chosen; and the search is continued until there is little or no further improvement in the value of the function, indicating that the optimum point has been reached. Most of this chapter is devoted to direct methods.

Indirect methods, on the other hand, are concerned with the knowledge of the function characteristics at or near the optimum. The necessary and sufficient conditions for optimality are first established; if the conditions are satisfied, the optimal policy (values of the independent variables at the optimum) is then determined. This ultimately involves solving a set of (linear or non-linear) equations rather than searching for an optimum.

For analytical solutions, the indirect methods are generally used; but when numerical results are sought, the direct methods are often preferred, T.N. Edelbaum[53] has presented a very good summary of the advantages and disadvantages of both the direct and indirect methods.

5.2. Indirect Methods:

As we have indicated above, most of this Chapter will be devoted to direct methods; and the indirect methods will be considered very, very briefly. These include the 'differential method' and geometric programming.

A. Differential

This is one of the oldest optimization techniques. It is based on the fact that for $\bar{x}^*$ to be saddle point of $F(\bar{x})$, the gradient of the latter must vanish at that point; i.e.

$$\nabla F(\bar{x}^*) = 0 \qquad (5.2.1.)$$

Relation (5.2.1.) is a necessary condition.

Condition (5.2.1) results in a set of non-linear (or linear) simultaneous partial differential equations which can then be solved by methods such as Newton-Raphson or Quasi-Newton methods.

At the saddle point, x*, the following expression is obtained by means of Taylor series expansion:

$$\partial F(\bar{x})^* = \tfrac{1}{2}(\partial \bar{x})^T H^* \partial \bar{x} \qquad (5.2.2)$$

where H is a Hessian matrix (i.e. the matrix of second order derivatives).

A sufficient condition for $\bar{x}^*$ to be a local minimum is that H^* be positive definite ($\equiv H > 0$).

Constraints:

The method adopted for handling constraints will depend on the type of constraints in question. For example, strict equality constraints give rise to Langrange-type problems considered in Chapter 2. Similarly, if there are both equality and inequality constraints, we have the generalized Lagrange Problem (Chapter 2).

Another important indirect method for handling mixed constraints - called the Maximum Principle - will be discussed briefly in Chapter 6.

B. Geometric Programming:

This is a relatively new method: it was first proposed in 1961.[83] The essential concept of the technique is that instead of seeking the optimal values of the independent variables first, geometric programming determines the optimal way to distribute the total cost amongst the various terms of the objective function.

Once these optimal allocations are obtained, the optimal cost can be found by routine calculations, Which in some cases may involve solving a set of non-linear equations.

The name geometric programming is derived from the fact that the development of the technique relies on the dual relationship which exists between the arithmetic and geometric means of a certain type of functions considered.

A detailed exposition and discussion of the method may be found in 77 and 83 .

Remarks:

Geometric programming has received considerably little attention but its potentialities are apparent, especially for problems or functions that can be formulated in the general posynomial* form, i.e.,

$$F(x) = \sum_{j=1}^{J} c_j \prod_{i}^{N} x_i^{a_{ji}} \quad ;$$

e.g:

a) optimal reliability problems,

b) efficiency of cascaded governor, turbine, and generator combinations,

c) optimal design of electrical equipment whose cost functions can be expressed as products of the design variables.

A major advantage is that the method can handle very highly non-linear functions (both in the objective and constraints). And for certain small size problems the optimal value of the function can be obtained by inspection

The author has only recently heard of the method, and has, therefore been unable to present solutions to specific power system problems.

5.3. DIRECT METHODS

5.3.1. Unconstrained Problems

Although practically all mathematical programming problems involve a variety of constraints it is essential to study problems with no constraints. For a large number of techniques that handle constrained problems rely on the

* See for example, reference 77 for a definition

procedures for solving the unconstrained problems; all that is needed is to make appropriate modifications in the function to take into account the constraints.

As has been indicated (section 5.1) the direct methods for function optimization are iterative in nature: thus, starting with arbitrary values of the variables, a direction for the next step; and the step-length are chosen. The process is repeated until the desired degree of accuracy has been attained.

Mathematically, this is represented as:

$$\bar{x}^{K+1} = \bar{x}^{K} + t^{K}.\Delta\bar{x}^{K} \qquad (5.3.1)$$

where $\Delta\bar{x}^{K}$ is the direction, and t^{K} the step-length to be determined; and K is the number of iterations.

5.3.1.A. Choice of Step:

The step length t^{K} is usually chosen as the value of $t \geq 0$ which minimizes the function

$$F(\bar{x}^{K} + t\ \Delta\bar{x}^{K}) \qquad (5.3.2)$$

The "best" value of t^{K} is obtained by means of either cubic interpolation or by quadratic interpolation.*

However, in many instances, t is chosen to be identically equal to one, or to any other arbitrary value. This value can then be arbitrarily changed in the course of the computational steps in such a manner as to increase the rate of convergence.

* See Appendix A5

5.3.1.B. Choice of Direction ($\bar{x}^K$)

Two general methods are widely used for this purpose:

a) In the direction of the first partial derivatives of the function $\equiv \nabla F(\bar{x})$. These are the "grandient methods".

b) In a direction dependent on $\nabla F(\bar{x})$, but intended to improve on that direction - often referred to as "modified gradient" techniques.

(i) Gradient Method:

This is alternatively called the "method of steepest ∂scent (descent). It chooses

$$\Delta\bar{x}^K = -t^K(\nabla F(\bar{x}^K)) \qquad (5.3.2)$$

i.e. in the direction in which $F(\bar{x})$ has the steepest slope. The method is one of the most widely used, and has received a great deal of study. A major disadvantage of the method is that of slow convergence; furthermore, the method is susceptible to oscillations in $\bar{x}$. [26]

(ii) Modified Gradient Techniques

A large number of methods fall into this category; notably: Newton's method, conjugate directions, conjugate gradients, projection methods, and partan' (method of parallel tangents).

The essential feature of the methods is that the gradient of the function is modified so as to give a better direction of motion in the iterative step:

$$\Delta \bar{x}^K = -t^{\underline{K}} \left[GH(\bar{x}^K) \right]^{-1} \nabla F(\bar{x}^K) \qquad (5.3.3)$$

where the function $GH(\bar{x})$ differs according to which method is used.

Note that the gradient method can be viewed as a special form of (5.3.3) for which $GH(\bar{x})^{-1} = 1$.

Necessary and Sufficient Conditions for a Local Optimum.

The process (5.3.3) continues until a stationary point, $\bar{x}^o$ has been found (if one exists) i.e., until the point for which the following relation holds:

$$\nabla F(\bar{x}^o) = 0 \qquad (5.3.4)$$

has been reached.

Necessary conditions that a point $\bar{x}^o$ be a local minimum (maximum) to $F(\bar{x})$ are that

$$GH(\bar{x}^o) \geq 0 \qquad (5.3.5)$$

$$\nabla F(\bar{x}^o) = 0 \qquad (5.3.6)$$

Sufficient conditions that a point $\bar{x}^o$ be a local minimum (maximum) to $F(\bar{x})$ are that

$$GH(\bar{x}^o) > 0 \qquad (5.3.7)$$

$$\nabla F(\bar{x}^o) = 0 \qquad (5.3.8)$$

Note that for a convex function: (i.e. a function for which the following relation holds

$$F(y) \geq F(x) + (y-x)^T \nabla F(x) \qquad (5.3.9)$$

for all y and x) the local minimum is also the global minimum.

Methods of Conjugate Directions:

Consider a general quadratic function

$$F(\bar{x}) = a + \bar{c}^T\bar{x} + \tfrac{1}{2}\bar{x}^TQ\bar{x} \qquad (5.3.10)$$

where Q is an nxn positive definite symmetric matrix: a and c_i are constants.

A set of vectors s_i (i=1,...,n), $s_i \neq 0$ with the property that

$$s_iQ\,s_j = 0 \qquad i \neq j \qquad (5.3.11)$$

is said to be orthogonal (conjugate) with respect to Q.

Any procedure for obtaining the minimum value of a function $F(\bar{x})$ - starting from an arbitrary point, $\bar{x}^o$ - by generating a sequence of steps ($t^K\bar{s}^K = x^{K+1}-x^K$) that are Q conjugate is called a conjugate direction algorithm.

The vectors $t^K\bar{s}^K$ K = 1,...,n are linearly in.dependent and form a basis in the n-space.

So far, methods of conjugate directions have proved to be the most efficient for function optimization. For quadratic functions, covergence to an optimum in at most n(≡ number of variables) is assured.

However, the methods can handle any convex non-quadrati functions. This is so because in the neighbourhood of the optimum point $\bar{x}^*$, the function is nearly quadratic and can thus be approximated:

$$F(\bar{x}) = F(\bar{x}^*) + \tfrac{1}{2}(\bar{x}-\bar{x}^*)^T\,H(\bar{x}-\bar{x}^*) + \text{higher terms} \qquad (5.3.12)$$

where H is a Hessian matrix; is symmetric and positive definite. The higher orders are negligible.

Several conjugate direction methods are briefly discussed below.

Method 1

a) Given an initial point $\bar{x}^1$, the subsequent direction is given by

$$\Delta\bar{x}_i = t_i\bar{s}_i \qquad (5.3.13)$$

where the initial direction is given by

$$s_1 = -\nabla F(\bar{x}^1) \qquad (5.3.14)$$

and

$$s_i = -\nabla F_i + \left(\frac{\nabla F_i^T \cdot \nabla F_i}{\nabla F_{i-1}^T \cdot \nabla F_{i-1}}\right) \cdot s_{i-1}$$

$$i = 2,3,\ldots,n \qquad (5.3.15)$$

Where $t_i \geq 0$ is selected by a one-dimensional search for the minimum of the function $F(\bar{x})$ versus t along the line determined by the direction of the vector s_i - (5.3.1).

The process is continued until the optimum has been obtained - within the desired degree of accuracy.

(b) This is essentially the same as (a) except for the fact that, the approximation of the quadratic function is reassessed after every n straight-line minimization searches.

Thus after the (n)th iteration, $\bar{x}^1$ is replaced by $\bar{x}^n$ and the process continued until no further improvement in the objective function is observed. With this modification, it may be possible to obtain a rapid convergence.

Method 2

In the previous method, $F(\bar{x})$ was assumed to be quadratic. In the following method, a minimum amount of knowledge about the quadratic nature of the function is assumed.

Given as initial starting point $\bar{x}^1$, subsequent directions are pursued according to

$$\Delta \bar{x}_i = t_i \bar{s}_i \qquad (5.3.16)$$

$$s_1 = -\nabla F_1(\bar{x}) \qquad (5.3.17)$$

$$s_i = -\nabla F_i(\bar{x}) + \left[\sum_{j=1}^{i-1} \frac{\Delta \bar{x}_j \; \Delta(\nabla F_j(\bar{x}))^T}{\Delta \bar{x}_j^T \; \Delta(\nabla F_j(\bar{x}))} \right] . \nabla F_i(x)$$

$$i = 2,3,\ldots,n \quad (5.3.18)$$

Method 3

The first two steps are identical to those of Method 2 (5.3.16) and (5.3.17). However, s_i is chosen according to the following rule:

$$s_i = -\nabla F_i(x) + \left[\frac{\Delta x_{i-1} \; \Delta(\nabla F_{i-1}(\bar{x}))^T}{\Delta x_{i-1}{}^T \Delta(\nabla F_{i-1}(\bar{x}))} \right] . \nabla F_i(\bar{x})$$

$$i = 2,3,\ldots,n \quad (5.3.19)$$

Method 4 [17]

Starting at point $\bar{x}^1$, subsequent directions are obtained by applying the following rules:

$$\Delta\bar{x}_i = t_i\bar{s}_i \qquad (5.3.20)$$

$$\bar{s}_1 = -\nabla F(\bar{x}^1) \qquad (5.3.21)$$

$$s_i = -\nabla F_i(\bar{x}) + \left[\sum_{j=1}^{i-1} \left(\frac{\nabla F_i^T(\bar{x})\nabla F_{j+1}(\bar{x}) - \nabla F_i^T(\bar{x})\nabla F_j(\bar{x})}{-\bar{s}_i^T\,\nabla F_i(\bar{x})}\right)\right] \cdot \nabla F_i \qquad i = 2,3\ldots$$

$$(5.3.22)$$

At the expense of a certain amount of complexity in the computed programme, especially for higher order systems methods 2 through to 4 inclusive, may increase the rate of convergence. The device of restarting the conjugate gradient process for methods 2 and 3 (as with method 1) further increases the rate of convergence.

Methods 1 through to 4 are different versions of conjugate gradients. If F(x) is quadratic and there are no round-off errors, the four methods become identical.[46]

Method 5

The conjugate directions are given by

$$\Delta\bar{x}_i = t_i\,\bar{H}_i\,\nabla F_i(\bar{x}) \qquad (5.3.23)$$

where $\bar{H}_i$ is defined by

$$\bar{H}_i = \bar{H}_{i-1} + \frac{\Delta\bar{x}_{i-1}\Delta\bar{x}_{i-1}^{\,T}}{\Delta\bar{x}_{i-1}^{\,T}\Delta(\nabla F_{i-1}(\bar{x}))} - \frac{\bar{H}_{i-1}\Delta(\nabla F_{i-1}(\bar{x}))\Delta(\nabla F_{i-1}^{(\bar{x})T})\bar{H}_{i}}{\Delta(\nabla F_{i-1}(\bar{x}))^T\bar{H}_{i-1}\Delta(\nabla F_{i-1}(\bar{x}))}$$

$$i = 2,3,\ldots \qquad (5.3.24)$$

H_1 is an arbitrarily specified positive definite matrix, and is usually given as an identity matrix.

Method 5 is the variable matrix algorithm developed by Davidson and a variation of which was examined by Fletcher and Powell.[29]

The updating process of $\bar{H}_i$ is such that its value approaches that of $\nabla^2 F(x)^{-1}$ at the optimum point.

Recently Kelley and Myers have suggested a modification to Davidson's method along the following lines:[46]

$$\bar{H}_i = H_{i-1} - \frac{\bar{H}_{i-1}\,\Delta(\nabla F_{i-1}(\bar{x}))\Delta(\nabla F_{i-1}(\bar{x}))^T\,\bar{H}_{i-1}}{\Delta(\nabla F_{i-1}(\bar{x}))^T\,\bar{H}_{i-1}\,\Delta(\nabla F_{i-1}(\bar{x}))}$$

$$i \geq 2,\ i \neq mn + 1 \qquad (5.3.25)$$

$$\bar{H}_{mn+1} = \sum_{j=mn-n+1}^{mn} \frac{\Delta\bar{x}_j \Delta\bar{x}_j^{\,T}}{\Delta\bar{x}_j^{\,T}\Delta(\nabla F_j(\bar{x}))} \qquad (5.3.26)$$

$\bar{H}_1$ is the same as above.

Here the matrix $\bar{H}$ is reduced in rank by one at each up-date and, after the rank has been reduced to zero, is replaced by the estimate of $\Delta^2 F(\bar{x})$ given by (5.3.26). The estimate is exact for a quadratic function with no round-off errors and for which conjugate steps are taken. The authors report quite reasonable convergence for the modified version.

The above conjugate direction methods represent iterative processes requiring a linear search, for a one-dimensional minimum at each iteration. All exhibit theoretical n-step convergence for a quadratic function. The choice of which method to use will depend upon the complexity of the problem, computer storage requirements, departure from the quadratic character of the function being optimized and acceptable maximum round-off errors.

Methods 1,3,4 and 5 retain their descent properties for the general non-quadratic functions. Method 2 does not. For its descent properties depend on the conjugacy of the previous steps in the sense of equation (5.3.11) i.e. conjugacy with respect to some positive definite matrix for all $i \neq j$. Such a restriction is realized only for a quadratic function, in which the extra number of the sum (5.3.18) vanish so that the method reduces to method 3.

Computational experience seems to indicate that Davidson's variable metric minimization technique has better convergence then any of the other methods of conjugate directions. The variable metric method, especially in double precision format, is also much less susceptible to round-off errors than any of the other methods. This property is largely due to the fact that there is continuous compensation for errors from the one-dimensional minima in the directions previously searched, whether caused by propagation of round-off errors or by departure from quadratic functional form.

However, for certain ill-conditioned problems; or for situations where the computations are performed with eight or less significant figures (single precision); or if there is a large number of truncation or round-off in the particular problem being tackled; e.g. if the gradient is being obtained by approximation, then the H matrix and/or $\nabla F(\bar{x})$ may be singular thus causing a break-down of Davidson's method. The other conjugate gradient methods do not share this limitation.

Another advantage of the methods of conjugate gradients is that they require less computer storage. With the variable metric method, the entire matrix H has to be stored and updated at each iteration. This limits the size of the problems that can be handled, especially in the case of double precision arithmetic which is quite often necessary in the variable metric scheme.

In addition, the sheer simplicity of the conjugate gradient algorithms is itself quite attractive.

Other versions of Conjugate Direction Algorithms

There are many other versions of the methods of conjugate directions. A detailed discussion of these may be found in the review paper by R. Fletcher.[26] These include the generalized Newton's method; the method of parallel subspaces (which can be used with or without the knowledge of the gradient of the function; the method of parallel tangents ("Partan"), which may be considered as a modified form of the method of parallel subspaces; and several projection methods; etc.

Another projection method which was proposed by Zontendijk has been applied by Pearson and McCormick[63] The method seems to have reasonably good convergence.

Non-gradient Methods

Minimization methods which require no evaluation of derivatives have been studied by a number of authors. The efficiency of some of these methods was evaluated by R. Fletcher in a review paper.[27] These included the method of Davies, Swann and Campey, the method of Powell and a modification of the method by Smith. The first one of these (Davies et al) is in fact a modification of Rosenbrock's method so as to include linear minimizations.

Fletcher's conclusions were as follows: the modified Smith's method is not as good as the others both in terms of the number of linear minimizations and the number of function evaluations required. Compared to the method of Davies et al in terms of function evaluations, Powell's method is the more superior; however both methods stand on about equal footing in terms of the number of linear minimizations required, with the former method somewhat more efficient at point removed from the optimum. Powell's method has quadratic convergence properties; the method of Davies et al does not. Consequently, in the neighbourhood of the minimum the quadratic convergence of Powell's method asserts itself and the final convergence to the minimum is more rapid. With the increase in the number of variables, however, the advantages enjoyed by Powell's method vanish.

In a recent paper, Zangwill[81] claims to have found a flaw in the theory underlying Powell's method. This flaw seems to be the major cause of the convergence difficulties encountered by Powell's method for cases when a function has many variables (usually 5 or more). Zangwill has suggested simplifications and improvements to overcome the said difficulties. However, to the best of the author's knowledge, no results are available to confirm this. Zangwill has also proposed a method, based on Powell's theorems, which has theoretical convergence for a strictly convex differentiable function. The method has since been programmed* and has shown a reasonable convergence rate - at least for the simple problems tested.

In another recent paper, Stewart III[74] has proposed a modification of Davidson's variable metric algorithm

* by Mr. J.N. Ray of Imperial College.

that would enable one to approximate the gradient vector by differences. He reports adequate convergence for a number of test problems, and shows that for some of the problems considered, his method does better than Powell's. However, the steps of the proposed methods are rather complicated and time-consuming, thus counter-balancing any of the advantages that it might enjoy.

5.3.2. Constrained Problems:

These fall into three general classifications; namely:

(a) Those which are direct extensions of the simplex method (the simplex method is discussed in Chapter 3).

(b) Methods of feasible directions, which work with linear sub-problems while at the same time making use of techniques originally developed for unconstrained problems;

(c) Penalty-function techniques, which involve a sequence of unconstrained optimization procedures.

Extensions of the Simplex Method

Many of these have been amply discussed by a number of authors.[84] They include reduced gradient, cutting-plane, method of approximation programming, and separable programming.

The reduced-gradient method uses the gradient of the objective function to determine the desired direction of motion. It works only with linear constraints (and non-linear or linear objective functions). Its computational basis is that of the simplex method (Chapter 3).

The method has been shown to converge to a solution for a non-linear objective function; and to terminate for the linear objective function for the case where the objective is bounded and the constraints of the problem are non-degenerate. In situations where the rate of convergence is slow, acceleration methods: e.g. modifying the direction of motion may result in some improvement.

Separable programming method was first formulated by Miller.[58] It provides a simple technique for handling arbitrary non-linear functions of single arguments in either constraints or objective functions of an otherwise linear programming problem. Furthermore, the method can readily be adapted to handle product terms.

It is called separable programming because it assumes that all the non-linear expressions in the given problem can be separated into sums and differences of non-linear functions of single arguments. A detailed discussion of the method may be obtained in.[58]

A major disadvantage of the separable programming method is that it imposes a severe restriction on the type of non-linearities that can be handled. Moreover, although it can handle product terms, this is only applicable to small problem (several variables).

The <u>cutting-plane method</u> was developed by Kelly[46] and independently by Hartley and Hocking.[84] The method is based on the idea that the constraint set can be represented as the intersection of a sufficiently numerous set of half-spaces which contains it.

An essential point of the procedure is that the non-linear function (or constraint) is replaced by a first-order Taylor series approximation; e.g. for the constraint functions:

$$G_i(\bar{x}) = G_i(\bar{x}^K) + \nabla G_i(\bar{x}^K)(\bar{x} - \bar{x}^K) \leq 0 \qquad (5.3.27)$$

where the expansion is carried out about the point $\bar{x}^K$. Note that if $G_i(\bar{x})$ is convex then the approximation (5.3.27) will never be greater than $G_i(\bar{x})$.

Once the linearization has been accomplished the linear programming problem is solved along the lines described in [78].

Convergence is assured if both the objective function and the constraints are convex. However, the method does not work for non-convex problems. This is a serious limitation since most of the problems encountered in practice are non-convex.

It also suffers from other drawbacks: convergence is ratherslow, especially if the optimum is not in a vertex and the linear approximations are subject to serious round-off errors, especially if the optimum is not in a vertex.

Wolfe has proposed an accelerating method suitable for problems with linear constraints but there is, as yet, no computational result to confirm the efficacy of the acceleration procedure.

A major advantage of the cutting-plane methods is that they are efficient for convex problems which are nearly linear. Furthermore, the algorithms involve relatively little work per step and the computer programmes are quite simple.

The Method of Approximation Programming has some relation to the cutting-plane methods. The only differences are: [37]

(a) for this method, the initial point, $\bar{x}^o$. has to be feasible.

(b) a complete relinearization takes place at each step;

(c) the gradient step-size is a predetermined small value.

The computational procedure is as follows: For a given initial point $\bar{x}^o$

$$\text{Maximize} \quad \hat{F}(\bar{x}) = F(\bar{x}^K) + \nabla F(\bar{x} - \bar{x}^K) \qquad (5.3.28)$$

$$\text{subject to} \quad G_i(\bar{x}^K) + \nabla G_i(\bar{x}^K)(\bar{x} - \bar{x}^K) \le 0$$

$$K = 1,\ i = 1,\ldots,m \qquad (5.3.29)$$

$$|\bar{x}_i - \bar{x}_i^K| \le \epsilon_K \qquad (5.3.30)$$

The process is repeated for K = 2,... with decreasing values of step-size, $\epsilon_K > 0$, until the improvement in the value of the objective function becomes sufficiently small and the infeasibility in $\bar{x}^K$ is acceptable.

The method has been successfully applied to solve a large number of problems, both convex and non-convex, with a reasonably high degree of accuracy.

However, because many small steps are needed, and because linearization is undertaken at each step, convergence is quite slow.

Methods of Feasible Directions:

These use the same general approach as the methods of unconstrained optimization. However, they have been modified to deal with inequality constraints. A great number of techniques described in [85] belong to this class.

The guiding concepts are as follows: an initial feasible point is determined. Thereafter, the solution process moves along a direction in such a way that no constraint is violated while at the same time the objective function is improved. The process is repeated, until a point is reached from which no improvement of the objective function is possible without violating at least one of the constraints. In general, such a point is a constrained local optimum and not necessarily a global optimum for the entire region of interest.

A direction along which a small move can be made without violating any constraints is called a feasible direction; while a feasible direction which improves the objective function is called a usuable feasible direction. Because there are many ways of choosing such directions, there are many different methods of feasible directions.

In this section we shall discuss a method due to Zoutendijk and Rosen's Gradient-Projection method.[68] Modifications and extensions of Rosen's method by Goldfard-Lapidus[35] and Murtagh-Sargent[60] will also be discussed.

(i) Zoutendijk's Method of Feasible Directions:[85]

Consider the optimiaation problem given by equations (5.3.43) and (5.3.44). A typical method of feasible direction proceeds according to the following rules.

(a) We start with an initial feasible point, $\bar{x}^o$. Suppose that $\bar{x}^o, \bar{x}^1, \ldots, \bar{x}^{K-1}$ have already been calculated.

(b) at current point, $\bar{x}^{K-1}$, a usable feasible direction is determined: i.e. a direction, p^{K-1} with the property that a $\bar{t} > 0$ exists such that for all t, $0 < t \leq \bar{t}$.

$$\bar{x}^{K-1} + t\,p^{K-1} \in \Omega \tag{5.3.31}$$

and

$$F(\bar{x}^{K-1} + t\,p^{K-1}) > F(\bar{x}^{K-1}) \tag{5.3.32}$$

(c) the step-length t^{K-1} is determined by solving the one dimensional maximum problem in t

$$\text{Max} \quad F(\bar{x}^{K-1} + t\,p^{K-1}) \tag{5.3.33}$$

$$\text{subject to} \quad \bar{x}^{K-1} + t\,p^{K-1} \in \Omega \tag{5.3.34}$$

(d) the new usable feasible direction is then computed. The direction finding problem is easy to formulate in the case of linear constraints ($G_i(x) \leq 0$ $\equiv A\bar{x} \leq \bar{b}$). Suppose the present solution is $\bar{x}$, then the following problem is solved:

$$\text{Maximize} \quad \nabla F(\hat{\bar{x}})^T.p \tag{5.3.35}$$

$$\text{subject to} \quad Ap \leq 0 \; (\equiv A\hat{\bar{x}} = \bar{b}) \tag{5.3.36}$$

$$p^T p \leq 1* \tag{5.3.37}$$

In the case of non-linear constraints for which $G_i(\hat{\bar{x}}) \nleq 0$ we have the problem:

* other alternative normalization constraints include: $-1 \leq p_j \leq 1$ for all j or $\Sigma|p_j| \leq 1$ or $|x_j^{K-1} - p_j| \leq 1$

$$\text{Maximize:} \quad \xi \qquad (5.3.38)$$

$$\text{subject to} \quad \nabla G_i(\hat{\bar{x}})^T p + \tau_i \xi \leq 0 \qquad (5.3.39)$$

$$-\nabla F(\hat{\bar{x}})^T \bar{p} + \xi \leq 0 \qquad (5.3.40)$$

$$p^T p \leq 1 \qquad (5.3.41)$$

where $\tau_i = 0$ if $G_i(x)$ is linear and $\tau_i > 0$ if $G_i(x)$ is non-linear.

In either linear or non-linear case, the process is repeated until either (a) $p_j \equiv 0$ or (b) the decrease in the objecfive function is sufficiently small. Zoutendijk has shown that this process will converge in a finite number of iterations. A general procedure is illustrated in Fig. 5.3.1.

The methods are applicable to non-convex problems, and fast convergence can be expected, especially if τ_i are properly chosen. In the linear case, if a linear normalization procedure is used, the technique reduces to an efficient linear programming method. Generally, quite accurate results can be expected, especially if the maximum does not lie on the vertex.

The main drawbacks are that the determination of step-length results in more work required per iteration; and the entire computer programme is rather complicated.

(ii) Rosen's Gradient Projection Method:[68]

Another drawback with Zoutendijk's method of feasible directions is that an optimization problem

(Max ξ or Max $\nabla F(x).p$) must be solved to find a direction in which to move. This procedure can be quite time consuming. Rosen has developed a method that gets over this difficulty: the gradient projection method. According to Rosen's procedure, a usable feasible direction is found without solving the optimization sub-problem - although the ensuing direction may not be locally 'best'. It utilizes the Kuhn-Tucker conditions, both to generate new directions, and to stop the solution process.

The procedure is illustrated in Fig. 5.3.3. for the case of linear constraints; (i.e. $A\bar{x} \geq \bar{b}$) the constraint set is a convex polyhedron, with the boundaries determined by $A\bar{x} = \bar{b}$. A typical example with linear constraints is illustrated in Fig. 5.3.2. The points $\bar{x}^3$ and $\bar{x}^4$ in the diagram have been obtained by minimizing along the directions $\bar{x}^2$, $\bar{x}^3$ and $\bar{x}_3$, $\bar{x}_4$ respectively.

In the gradient projection procedure, a lot of effort goes into the computation of the various projections. The projection of a vector $\bar{a}$ into a given vector in space Ω is another vector $\bar{b}$, the latter being obtained by multiplying $\bar{a}$ by a projection matrix $\hat{p}$.

For the linear constraint case, the appropriate projection matrix is

$$\hat{p} = I - \hat{M}(\hat{M}^T\hat{M})^{-1}M^T \qquad (5.3.42)$$

where $\hat{M}$ is a $\hat{p}$xn matrix corresponding to $\hat{p}$ rows of $\bar{A}$;

p is the intersection of the boundaries for which $A\bar{x} = \bar{b}$. P is recomputed each time the set of constraint changes, thus making the procedure quite cumbersome when one constraint set differs from the next radically.

(iii) Modifications and Extensions of the Gradient Projection Method.

As we have seen above, Rosen's gradient projection method, is based on the steepest descent(ascent) method for optimization coupled with orthogonal projection of the gradient into a linear manifold, which approximates the original constraints.

Goldfarb and Lapidus [35] have developed a method that is based on the use of conjugate direction with special modifications to handle constraints. Specifically, they have combined Rosen's orthogonal projection procedure with Fletcher and Powell's (modification of Davidson's) method, in such a manner as to take into account linear constraints. They report that their method requires much fewer functional evaluations than Rosen's - (at least for the test programme considered) and that it is more efficient with regards to highly non-linear problems.

However, the Goldgarb-Lapidus method suffers from several computational difficulties. The Fletcher-Powell method generates successive approximations to the Hessian matrix, H, of the function to be optimized, by

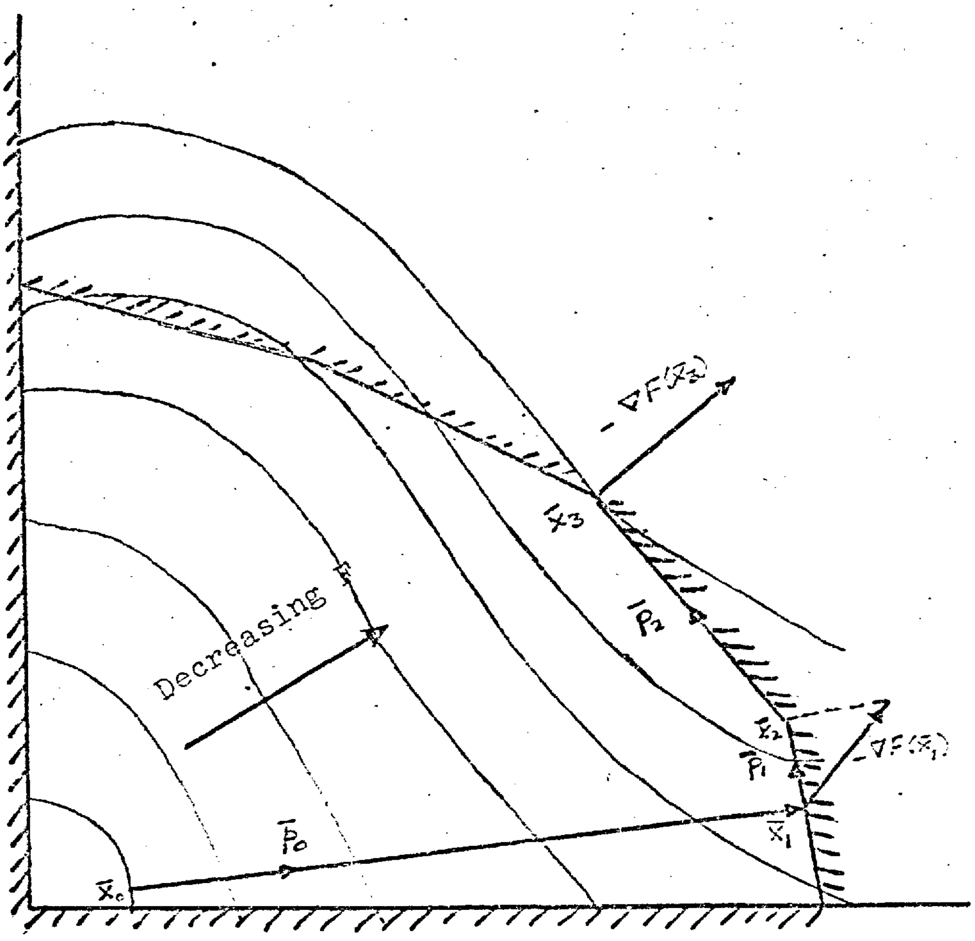


Fig. 5.3.1. Constrained Minimization with usable Feasible Directions. The starting point is $\bar{x}_o$. The desired minimum is at $\bar{x}_3'$.

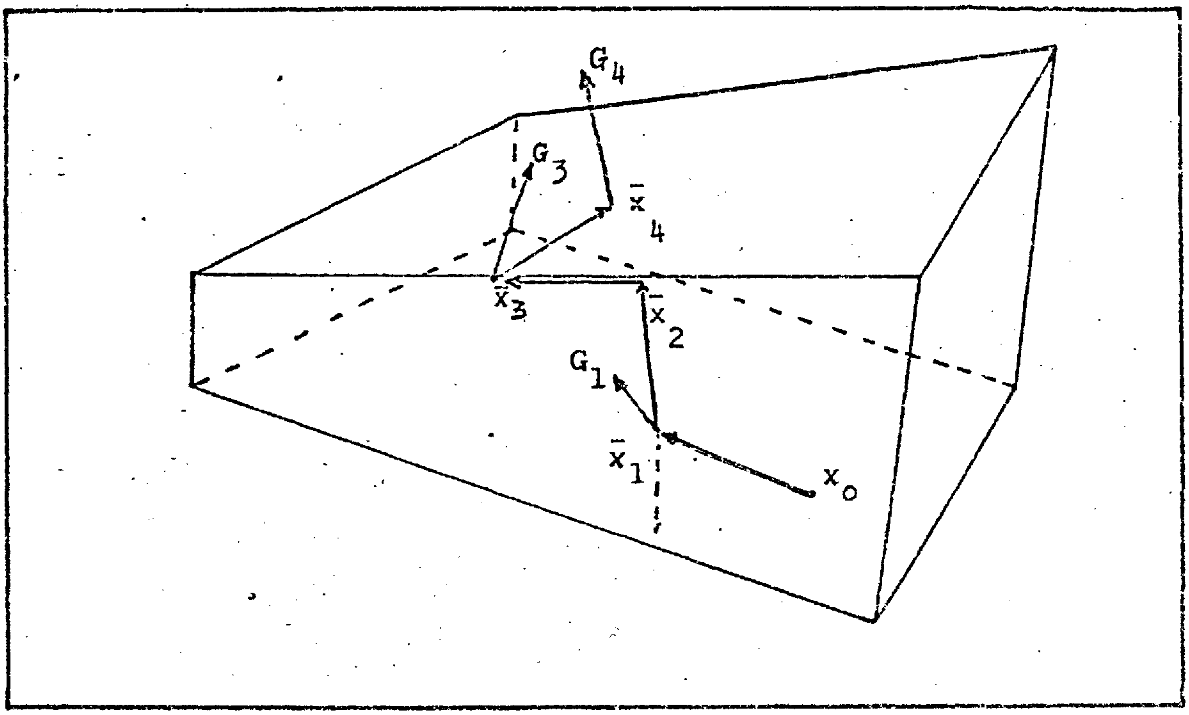


Fig. 5.3.2. Gradient Projection Search Procedure.

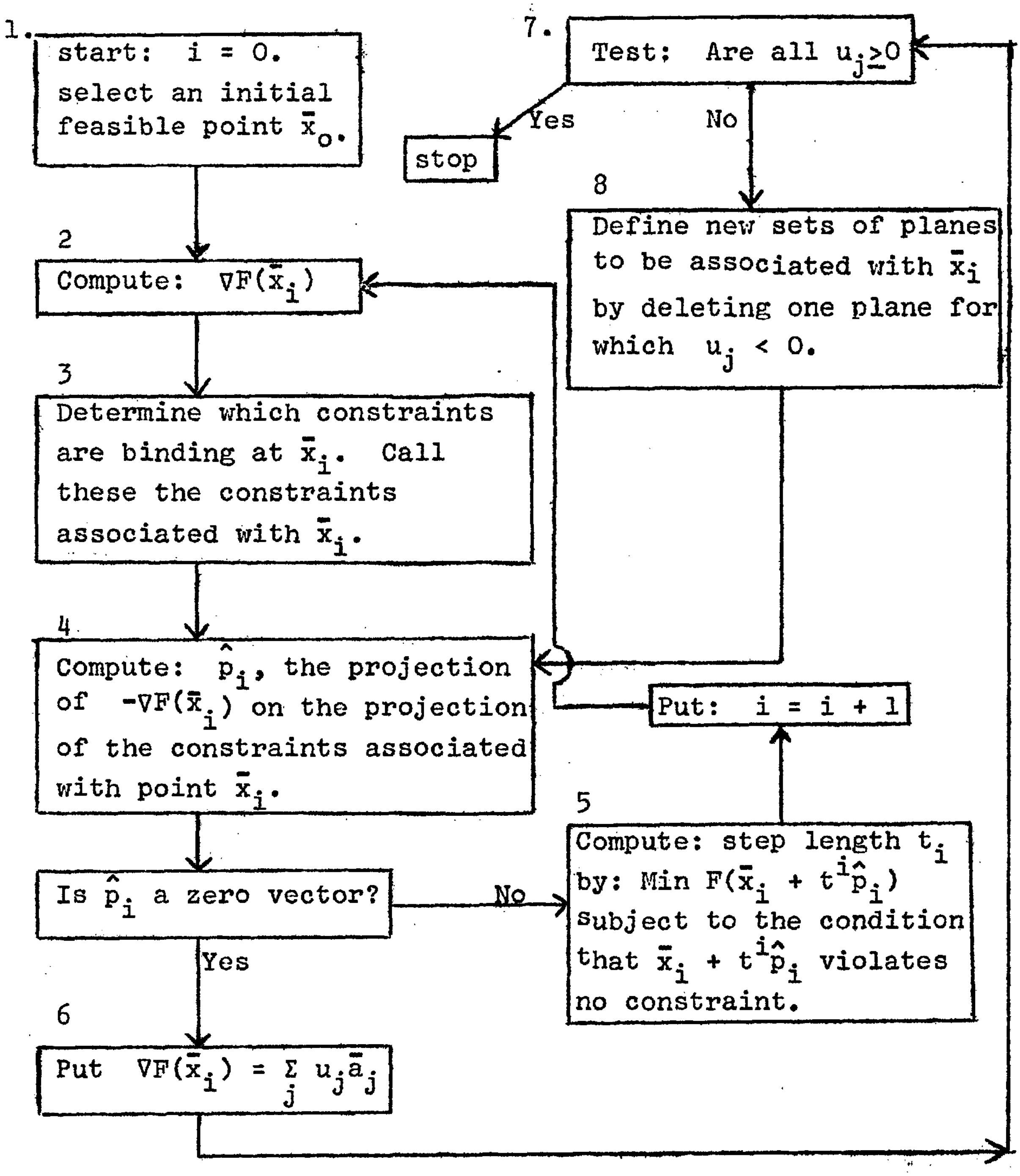

1.
start: $i = 0$.
select an initial feasible point $\bar{x}_o$.
2
Compute: $\nabla F(\bar{x}_i)$
3
Determine which constraints are binding at $\bar{x}_i$. Call these the constraints associated with $\bar{x}_i$.
4
Compute: $\hat{p}_i$, the projection of $-\nabla F(\bar{x}_i)$ on the projection of the constraints associated with point $\bar{x}_i$.
Is $\hat{p}_i$ a zero vector?
Yes
No
6
Put $\nabla F(\bar{x}_i) = \sum_j u_j \bar{a}_j$
7.
Test: Are all $u_j \geq 0$
Yes
No
stop
8
Define new sets of planes to be associated with $\bar{x}_i$ by deleting one plane for which $u_j < 0$.
Put: $i = i + 1$
5
Compute: step length t_i by: Min $F(\bar{x}_i + t^i\hat{p}_i)$ subject to the condition that $\bar{x}_i + t^i\hat{p}_i$ violates no constraint.

Fig. 5.3.3. Rosen's Gradient Projection Method:

seeking the optimum of the function along successive search directions. The procedure is based on the fact that at the end of each step, the function gradient is orthogonal to the direction of search; and that H is symmetric so that the search directions are mutually conjugate with respect to it.

The above procedure is inconveient for use with the gradient-projection method. For, in general, the gradient is not orthogonal to the search direction at a point where a constraint is encountered. Consequently, a new conjugate direction cannot be set up. Moreover, the inverse of the Hessian Matrix cannot be up-dated at such a point. Hence a new sequence of conjugate directions must be started each time an active constraint is changed. Furthermore, the Goldfarb-Lapidus method involves orthogonal projection of H^{-1} into the current constraint set, with the result that all accumulated information orthogonal to this set is lost.

In a recent paper Murtagh and Sargent have proposed a class of methods which makes it possible to up-date H^{-1} for steps of arbitrary length and direction.[60] This makes them particularly useful for use with gradient projection. They give examples with two methods for the test programs, one apparently has better convergence than the Goldfarb-Lapidus algorithm, while the other one does not fair so well. The Murtagh-Sargent methods are still in their development stage and further computational experience is awaited.

Penalty Function Methods

These involve transforming a given constrained problem into a sequence of unconstrained problems, which are then solved by the unconstrained optimization techniques which have already been discussed in this chapter.

Consider the following mathematical programming problem with inequality constraints:

$$\text{Minimize} \quad F(\bar{x}) \tag{5.3.43}$$

$$\text{subject to} \quad G_i(\bar{x}) \geq 0 \quad i = 1,m \tag{5.3.44}$$

The above problem is transformed into an unconstrained one containing a 'penalty function'. The new problem is denoted by:

$$P(\bar{x}) = F(\bar{x}) + \sum_{i=1}^{m} \phi\left[G_i(\bar{x})\right] \tag{5.3.45}$$

where $\phi\left[G_i(\bar{x})\right]$ is a 'penalty function' corresponding to a particular constraint, $G_i(x)$.

Several different ways of choosing a penalty function have been proposed. Some of these, including Fiacco and McCormick's modification and extension of Carroll's "Created Response Surface Technique", Lootsma's generalization of Frisch's "Logarithmic Potential Method", and Zangwill's method will be discussed briefly.

11,52

Created Response Technique

Carroll proposed the following penalty function:

$$\sum_{i}^{m} r_i . {}^{1}/G_i(x) \qquad r_i > 0 \quad i = 1,2,\ldots,m \qquad (5.3.46)$$

where r_i is a 'weighting factor'. Consequently, the new unconstrained problem becomes

$$P(\bar{x},r_i) = F(\bar{x}) + \sum_{i}^{m} \left[r_i \; {}^{1}/G_i(x) \right] \qquad (5.3.47)$$

Relation (5.3.46) is sometimes called the 'boundary repulsion term'; its function is to prevent an unconstrained optimization technique from obtaining a point outside the feasible region.

Equation 5.3.47 was later modified by Fiacco and McCormic.[52] Instead they proposed defining a function.

$$P(\bar{x},r) = F(\bar{x}) + r \, \Sigma \, {}^{1}/G_i(x) \qquad (5.3.48)$$

with $r > 0$. The computational steps are then as follows.

We choose $r = r_1$, $(r_1 > 0)$. A point $\bar{x}^o$ is next chosen such that $G_i(\bar{x}^o) > 0$ (i.e. within the feasible region) for all i. We then proceed from $\bar{x}^o$ to $\bar{x}^1$ approximating, the minimum of $P(\bar{x}_1,r_1)$in a feasible region. A new function with $r = r_2$ $(r_2<r_1)$ is next formed; and the minimum of $P(\bar{x}_1 r_2)$ approximated from $\bar{x}^1$ to $\bar{x}^2$.

The process is repeated with monotonically decreasing values r_K, $K = 3,4,\ldots$ so that a sequence of points $\bar{x}(r_K)$ is generated, that approximate the minima of $P(\bar{x},r)$

The essential point about the procedure is that the sequence of P(r,x) minima converges to the optimum of the original programming problem (5.3.43; 5.3.44) as $r_K \to 0$: i.e. $P(r,\bar{x})$, $F(\bar{x}) \to F(\bar{x}^*)$ as $r_K \to 0$.

An important feature of this technique is that the optimization of P(x,r) yields a feasible solution $\bar{x}(r)$ as well as a feasible solution to the dual problem of 5.3.43. If $F(\bar{x})$ and $G_i(\bar{x})$ are convex, then the two values which bound $F(\bar{x}^*)$ can be found; namely:

$$F \quad \bar{x}(r) \quad + r \sum_{i=1}^{m} 1/G_i \; \bar{x}(r) \quad \leq F(\bar{x}^*) \leq F(\bar{x}(r)) \tag{5.3.49}$$

Relation (5.3.49) gives a convenient criterion for terminating the computational procedure. An extension of the method to problems having equality constraints has been proposed and successfully applied. If, in addition to the inequality constraints $G_i(\bar{x}) \geq 0$, $i = 1,\ldots,k$, we have a number of equations $G_i(\bar{x}) = 0$, $i = k+1,\ldots,m$ then the sequence of unconstrained problems to be solved becomes:

$$P(\bar{x},r) = F(\bar{x}) + r \sum_{i=1}^{K} 1/G_i(x) - \frac{1}{r} \sum_{i=K+1}^{m} \{\, G_i(x)\}^2 \tag{5.3.50}$$

Logarithmic Potential Method 55

This works on essentially the same principle as the Fiacco and McCormick's method. The boundary repulsion factor is

$$-r \sum_{i=1}^{m} \text{In } G_i(\bar{x}) \tag{5.3.51}$$

consequently a sequence of the unconstrained function

$$P(\bar{x},r) = F(\bar{x}) - r \sum_{i=1}^{m} \text{In } G_i(\bar{x}) \qquad (5.3.52)$$

is optimized for monotonically decreasing values of r_1 i.e. $r_1 > r_2 > \ldots > r_K \to 0$

A method of logarithmic potentials for solving linear programming problems was originally proposed by R. Frisch.[30] The above generalization (5.3.51) is due to Lootsma.[55]

As with equation (5.3.49), the optimum $F(\bar{x}^*)$ is bound by

$$F\ \bar{x}(r) - r \sum_{i=1}^{m} \text{In } G_i(\bar{x}) \leq F(\bar{x}^*) \leq F(\bar{x}(r)) \qquad (5.3.53)$$

where $r \sum_{i=1}^{m} \text{In } G_i(\bar{x})$ is the error term.

An outstanding feature of the logarithmic potential method is that the error term can be made arbitrarily small. Lootsma has shown that the error term can be approximated by

$$mr = r \sum_{i=1}^{m} \text{In } G_i(\bar{x}) \qquad (5.3.54)$$

where m is the number of constraints. Relation (5.3.53) enables one to choose a value of r in such a way that $F(\bar{x}^*)$ is approximated with a prescribed accuracy.

Like the method of Fiacco and McCormick, this method can also be extended to deal with equality constraints. So that corresponding to equation (5.3.51) we have

$$P(\bar{x},r) = F(\bar{x}) - r \sum_{i=1}^{K} \text{In } G_i(\bar{x}) + \sum_{i=K+1}^{m} \{G_i(\bar{x})\}^2 \tag{5.3.55}$$

A highly desirable feature of both methods (due to Fiacco et al and Lootsma) is that the necessity of coping separately with the boundary of the feasible region is avoided; that is, the new function, P(x,r) couples the objective function and the constraints in such a way that motion along the constraint boundary is avoided. For such motion is very cumbersome when the constraint surface is non-linear.

The main advantage of the methods is their ability to handle highly non-linear problems. However, they both suffer from the limitation that the starting point for the minimization process must be within the feasible region. Such a point may be difficult to obtain especially for large problems.

Zangwill's Method: [80]

Zangwill has proposed a penalty function procedure which is slightly different in concept to the above two. The major difference is that, a penalty is imposed only when a constraint is violated.

Consider a general mathematical programming problem with both equality and inequality constraints.

$$\text{Minimize} \quad F(\bar{x}) \tag{5.3.56}$$

$$\text{subject to} \quad G_i(\bar{x}) = 0 \quad i = 1,\ldots,m' \tag{5.3.57}$$

$$G_i(\bar{x}) \geq 0 \quad i = m'+1,\ldots,m \tag{5.3.58}$$

Zangwill suggested transforming the above problem into

$$P(\bar{x},r) = F(\bar{x}) + \frac{1}{r}\sum_{i=1}^{m'} |\xi_i| + \frac{1}{r}\sum_{i=m'+1}^{m} (g_i)^2 \tag{5.3.59}$$

where

$$\xi_i = \begin{cases} 0 & \text{if } G_i(\bar{x}) = 0 \\ G_i(x) & \text{if } G_i(\bar{x}) \neq 0 \quad i=1,\ldots,m' \end{cases} \tag{5.3.60}$$

and

$$g_i = \begin{cases} 0 & \text{if } G_i(\bar{x}) = 0 \\ G_i(\bar{x}) & \text{if } G_i(\bar{x}) < 0 \quad i = m'+1,\ldots,m \end{cases} \tag{5.3.61}$$

This method has the advantage that the initial minimization point is not required to be within the feasible region; so that the time that would otherwise be spent in driving all points into the constraint region (as the case for the above two methods) is saved.

Furthermore, the method is well suited for problems with large constraints. At any point in the minimization process, P(x,r) depends only on the unsatisfied constraints.

Consequently, when calculating the derivatives of $P(\bar{x},r)$ only the derivatives of the unsatisfied constraints need be considered. This clearly results in a definite saving in the computer storage requirement.

In general, the development and application of 'penalty functions' methods are receiving greater and greater attention. The methods have met with encouraging success; but there are still a number of computational problems that require further investigation. For example:

(i) what constitutes a good penalty function?

(ii) if a penalty function of the type considered above is chosen, how should the initial value of r be determined?

(iii) how should the value of r be reduced at each minimization step? Should the reduction be in specified steps or in a continuous fashion?

(iv) what unconstrained optimization technique should be used to solve the transformed successive unconstrained problems?

(v) for the interior penalty function methods (Fiacco et al and Lootsma) what is the most effective way of ensuring that the initial minimization point is within the feasible region?

(vi) to what extent can the rate of convergence be speeded up by some form of acceleration techniques?

(vii) what is the best way of handling linear constraints?

CHAPTER 6

DYNAMIC PROGRAMMING AND MAXIMUM PRINCIPLE.

6.0. In early 1950's R. Bellman and his co-workers developed a new general method for solving variational problems and called it dynamic programming.[6] The method has since been applied to a wide class of problems in optimal contrc l and general optimal sequential processes.

As a result oftheir work in the solution of optimal control problems in the mid-1950's Pontryagin and his pupils discovered the maximum principle.[65] Starting about 1956, the maximum principle was substantiated as a necessary and sufficient test for optimal processes in linear systems; and a necessary test for optimal processes in non-linear systems.

This chapter contains a brief discussion of the above methods.

6.1. DYNAMIC PROGRAMMING

Dynamic programming falls under the general class of sequential decision processes, and has been used widely for solving a certain class of optimization problems. It is based on the concept of multi-stage decision process: at each stage a choice (decision) is made, following which the next stage is reached.

The successive stages are related by known transformation rules. The values associated with the process depend both on the number of stages considered, and on the decisions made (per stage, and from one stage to another). For a given number of stages (with several possible states each) one set of decision sequence constitutes the "best" sequence; i.e. optimizes the given function.

The main elements of dynamic programming may be identified as: (a) states and state variables; (b) transformations; (c) decisions; (d) functional relations (recurrence relations); (e) Markovian-type processes and (f) principle of optimality.

In what follows, the above concepts will be defined precisely; and the inter-relationship amongst them established.

6.1.a. Markovian Type Process:

This is a very useful mathematical concept. A function, $F(x_1,x_2,\ldots,x_n)$ is Markovian if after a number of decisions, say m, the effect of the remaining (n-m) decisions upon the total return depends only upon the state of the sytem after the mth decision and subseqαent decisions; and not on the history of the decisions that preceeded the mth decision.

6.1.b. State and State Variables:

The state variables of a system (process) are those whose values completely specify the instantaneous situation of the system. So that the values of these variables tell all that need be known about the system for the purpose of making decisions.

We usually speak of state variables as specifying the state of the system. Thus the system will be in a particular state depending on the values taken by the state variables.

More generally, we speak of the state space as a set**, Ω , comprising all the possible states that the system may occupy. An element $X \epsilon \Omega$ is the state and may be interpreted as one of the situations in which the process may exist.

6.1.c. Decisions

The concept of decision may be viewed as the opportunity to change state variables - and hence the state of the system. For example, a decision to run a certain type of generator and stop another - in a generation scheduling problem - would lead to a change in the state variables.

In a more general context we may speak of $d_X(d_X \epsilon D_X)$ as representing one of the choices available when the system is in state X.

** See, for example, references[7] or[23] for a good account of the concept of a set.

6.1.d. Transformations

In dynamic programming, the process passes through the states in Ω in response to the decisions made at the various states. Thus when the process is in state X, selection of a decision, d_s determines a set $T(X,d_X)$ of states to which the process moves or might move from state X.

If a process is moved to a particular state with certainty, the set $T(X,d_X)$ would contain exactly one element. The set $T(X,d_X)$ would contain several elements in the case that two or more states result with certainty. The set function $T(X,d_X)$ is called the transition function (or transformation function) and it governs the evolution of the process.

In some states a particular decision will cause the process to terminate. If d_X is such a terminating decision at state X, then $T(X,d_X) = \phi$, the null set and no further transitions are possible.

If $T(X,d_X) = \phi$ for every $d_X \in D_X$ then X is called a termal state for the process.

6.1.e. Policy

In general the return obtained from a process depands on combinations of decisions, rather than on a single decision.

A policy, δ, may be considered as an ordered collection of decisions containing one decision for each state in Ω. The policy space, P_X, is the collection of all such policies. The . policy $\delta \in P_X$ prescribes a particular decision for each and every state $X \in \Omega$; and the policy space consists of all possible combinations of decisions at the various states; i.e. $P_X = \underset{X\in\Omega}{X.}D_X$

The policy space is defined in such a way that the decision selected for a particular states does not restrict the decisions available at the other states (though it may rule out transitions to certain states). This property is important in that it limits the class of problems to which dynamic programming can be applied effectively.

A policy which optimizes a prescribed return (objective) function is called an optimal policy.

6.1.f. Functional Equations

Let us define a real valued return function, $F_\delta(X)$, for each policy, δ, where $F_\delta(X)$ represents the return that would accrue if the process were started in state X and appropriate decisions in δ were applied at each of the states through which the process evolves.

We consider the return function to be of a simple additive form in which the total return function is taken to be the sum of a set of immediate returns associated with each of the status transversed by the process.

Suppose the immediate stage return is $r(X,d_X)$; then the total return

$$F_\delta(X) \quad = \quad \text{sum of all immediate returns } r(s,d_s) \tag{6.1.1.}$$

The sum is taken for all states traversed by the process starting from state X and evolving through states, s, in accordance with the corresponding decisions d_s in the policy δ.

So that the value of the additive return function (6.1.1) may alternatively be computed recurvsively from the relation

$$F_\delta(X) \quad = \quad \begin{cases} r(X,d_X) \quad \text{if} \quad T(X,d_X) = \phi & (6.1.2a) \\ r(X,d_X) + F_\delta(s) \quad \text{if } T(X,d_X) = \{s\} & (6.1.2b) \end{cases}$$

The recurrent relations (6.1.2a) and (6.1.2b) indicate that the total return from state X using policy δ is the sum of:

a) the immediate stage return, $r(X,d_X)$ from stage X using decision d_X;

b) the total return, F (s), under policy δ from the state, s, which results from choosing decision, d_X, while at state X.

6.1.g. Optimal Return Function

The optimal return from X is denoted by F(X) and is defined by

$$F(X) \;=\; \min_{\delta \in P_s} \; F_\delta(X) \qquad (6.1.3)$$

where the existence of a minimum is assumed.

For processes which evolve probabilistically, we speak of <u>expected returns</u>. So that the return, $F_\delta(s)$ from each state, $s \in T(X,d_X)$, is weighted by the probability $p(s;X,d_X)$ that the transition will occur to state s. This probability depends on X and d_X, but not on the other decisions in δ. The weighted returns are summed and added to the expected immediate return, yielding

$$F_\delta(X) \;=\; r(X,d_X) + \sum_{s \in T(X,d_X)} p(s:X,d_X)\, F_\delta(s) \qquad (6.1.4)$$

6.1.h. Principle of Optimality

Relations (6.1.3) and (6.1.4) have been arrived at via the application of a general technique called the <u>Principle of Optimality</u> which states:[6]

"An optimal policy has the property that whatever the initial state and initial decisions, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision".

The whole theory of dynamic programming relies very heavily on the Principle of Optimality. For, utilization of the latter guarantees that the decision made at each stage is the best decision in the light of the entire process.

6.1.i. Computational Aspects

Dynamic programming problems are often solved by numerical means rather than by strict analytical solutions.

A common method of solving dynamic programming problems is to set up a grid in the variables. Each node of the grid represents a set of numerical values for the variables. The various nodes are then explored to find the optimum node.

The above technique can readily be programmed for a computer. The approach makes it quite easy to test for inequality constraints; for example, if a node on the grid causes a constraint violation, the offending node is rejected. Adjacent nodes are then tested in order to establish the feasible region of the grid. A detailed example of the general computation process may be found in reference [67] . Appendix A7 of the thesis contains a few diagrams of the dynamic programming routine for solving a generation planning problem.

A major advantage of the dynamic programming procedure is that it effectively reduces the quantity of computation. Instead of solving the entire problem at one go, it proceeds in a step by step fashion. For example, in an N-stage process where there are d_X possible decisions, the combinational approach requires considering d_X^N possibilities. With dynamic programming, however, only d_X decisions are required at each stage; consequently only Nd_X possibilities are considered for the entire problem. This naturally leads to substantial savings in computer time; as a matter of fact, for large d_X and/or large N the time requirement for the combinational approach becomes prohibitive.

The main disadvantage of dynamic programming is that for high-dimensional problems (usually greater than four) and/or fine grid representation, the computer storage capacity and computation time become extremely large. This limits the size of the problems that can be handled by present-day computers.

Fortunately, however a number of devices are currently used to side-step the dimensionality problem. Some of these are listed below: [6,67]

(a) Linearizing the problem and using a Linear programming method of solution.

(b) Approximating the non-linear problem and solving by a quadratic programming algorithm.

(c) Employing a Lagrangian multiplier

(d) Employing polynomial approximations;

(e) Restricting the range over which the variables may vary from stage to stage

(f) Developing a variable grid system, rather than a fixed grid system.

6.2. MAXIMUM PRINCIPLE

This section contains a very brief account of the salient features of Pontryagni's Maximum Principle. First the conditions of optimality for a system described by ordinary differehtial equations are derived. This is followed by general discussions based on the extensions to the Maximum Principle to handle discrete cases as developed by Butkovskii [8]. Some relationship between the basic theorems of mathematical programming and Maximum Principle is also established.

Much of the theoretical development and application of Maximum Principle has been in the realm of optimal control processes. Consequently some of the notation used here will be of the type found in optimal control literature.

6.2.1. The Continuous Case

Given a performance index

$$F = \sum_{i=1}^{n} c_i x_i(t_n) \qquad (6.2.1.a)$$

the problem is to find a set of admissible controls u_i which transfers a given system from an initial state, $x(t_o)$, to a final state, $x(t_n)$ in such a way that 6.2.1. is minimized.

That is:

$$\text{Min} \quad F = \sum_{i=1}^{n} c_i x_i(t_n) \qquad (6.2.1.b)$$

subject to $\dot{\bar{x}} = \bar{f}(\bar{x},\bar{u}) \; ; \quad x(t_o) = x(o)$ (6.2.2)

$$\bar{x} \in \Omega \qquad (6.2.3)$$

$$\bar{u} \in U \qquad (6.2.4)$$

i.e. both $\bar{x}$ and $\bar{u}$ are bounded.

where (6.2.2) portrays the set of differential equations which characterize the dynamics of the system; $\bar{x} = (x_o, x_1, \ldots x_n)$ is a vector representing the state variables of the system and $\bar{u} = (u_1, u_2, \ldots, u_m)$ the control vector. $\bar{f}(\bar{x},\bar{u})$ is assumed to be continuous in both x_i and u_i and is continuously differentiable with respect to x_i, $i = 1,\ldots,n$,

The above system (6.2.1.b) is adjoined by a set of differential equations

$$\dot{\underline{\phi}} = - \sum_{j=0}^{n} \frac{\partial f_j}{dt} (\bar{x}(t),\bar{u}(t)).\ \phi_j(t) \qquad (6.2.5)$$

Systems (6.2.1.b) and (6.2.5) can then be combined by introducing the Hamiltonian, H, having the property that:[65]

$$H(\underline{\phi},\bar{x},\bar{u}) = \sum_{j=0}^{n} \phi_j f_j(\bar{u},\bar{x}) \qquad (6.2.6)$$

where H is a function of 2n+m+1 variables: $x_1,\ldots,x_n$; $\phi_o,\phi_1,\ldots,\phi_n$); and $u_1,u_2,\ldots,u_m$.

From (6.2.6) equations (6.2.1.b) and (6.2.5) can now be expressed in the form:

$$\dot{x} = \frac{\partial H}{\partial \phi_i} \qquad i = 0,\ldots,n \qquad (6.2.7)$$

$$\dot{\phi} = - \frac{\partial H}{\partial x_i} \qquad i = 0,\ldots,n \qquad (6.2.8)$$

For fixed values of $\bar{\phi}$ and $\bar{x}$, H is a function of $\bar{u}$ only. Suppose that the upper bound of the values of H is denoted by

$$M(\bar{\phi},\bar{x}) = \underset{\bar{u}\epsilon U}{\underline{S}up} \; H(\bar{\phi},\bar{\psi},\bar{u}) \qquad (6.2.9)$$

So that if H assumes its upper bound in U, then $\bar{M}(\bar{\phi},\bar{x})$ is the maximum values H for fixed $\bar{\phi}$ and $\bar{x}$.

A necessary condition for optimality states:

<u>Theorem 6.2.1</u>.

Let $\bar{u}(t)\epsilon U$ and let $\bar{x}(t)$ be the corresponding trajectory for equàtion (6.2.1.b). In order that $\bar{u}(t)$ and $\bar{x}(t)$ be optimal it is necessary that there exist a non-zero vector function $\bar{\phi}(t)$ corresponding to $\bar{u}(t)$ and $\bar{x}(t)$ such that

(a) for every t ($t_o \leq t \leq t_n$) the function $H(\bar{\phi}(t), \bar{x}(t), \bar{u})$ of variables $\bar{u} \epsilon U$ attains its maximum at $u = u(t)$ i.e.

$$H(\bar{\phi}(t), \bar{x}(t), \bar{u}(t)) = \bar{M}(\bar{\phi}(t), \bar{x}(t)) \quad (6.2.10)$$

(b) at the terminal time, t_n, the relations

$$\phi_o(t_n) \leq 0 \quad (6.2.11)$$

$$\bar{M}(\bar{\phi}(t_n), \bar{x}(t_n)) = 0 \quad (6.2.12)$$

are satisfied. Note however that if $\bar{\phi}(t)$, $\bar{x}(t)$ and $\bar{u}(t)$ satisfy (6.2.7), (6.2.8) and (a) then $\bar{\phi}(t)$ and $M(\bar{\phi}(t),\bar{x}(t))$ are constant; so that conditions 6.2.11 and 6.2.12 may be verified at any time t, $t_o \leq t \leq t_n$.

The principle content of the above theorem is equation (6.2.10) and is called the Maximum Principle. The essential point about the principle is summarized by the following:

Theorem 6.2.2.

If $\bar{u}^*(t)$ is the optimal control in that it minimizes the performance criterion, F, then it satisfies the maximum condition of the Hamiltonian, H, (i.e. maximizing H is a necessary condition for optimal control). In many problems, the uniqueness of $\bar{u}^*(t)$ can be shown; so that the above condition is also sufficient.

Theorem 6.2.3.[52,69]

For a linear system of equations of the type

$$\dot{x} = A(t)\bar{x} + B(t)\bar{u}.$$

and free right-end conditions (ie. the final values of the state variables are not bounded) the necessary and sufficient conditions for optimal control $\bar{u}(t)$ is the fulfilment of the maximum condition on the Hamiltonian, H.

A detailed proof of the Maximum Principle, together with extensions of the Principle are discussed in Refs.[65,69]

6.2.2. Discrete Maximum Principle

Since its original formulation, the maximum principle has been generalized to the case of minimizing an integral and to the case of bounded coordinates. By about 1959 Rozonoer[69] had established the connection between maximum principle and dynamic programming. He also proved the validity of the maximum principle for linear discrete-time systems.

More recently, a number of authors have tackled and advanced the theory of discrete maximum principle. The version by Butkovskii[8] will be discussed here. He obtained an analogous form of maximum principle which gives both necessary and sufficient conditions for optimality of systems described by difference equations.

We now consider the following problem:

$$\text{Minimize (Maximize)} \qquad F = \bar{c}^T.\bar{x}(N) \qquad (6.2.13)$$

subject to $\bar{x}(k+1) = \bar{f}(\bar{x}(k),\bar{u}(k)) \quad k = 0,\dots,N-1$

$$x(0) = \bar{a} \qquad (6.2.14)$$

$$\bar{x}(k) \in \Omega \qquad (6.2.15)$$

$$\bar{u}(k) \in U \qquad (6.2.16)$$

where $\bar{x}(k)$ is an nxl matrix $\bar{u}(k)$ an mxl matrix, $\bar{a}$ an nxl matrix and $\bar{f}(\bar{x}(k),\bar{u}(k))$ an nxl matrix.

The states of the system are described by $\bar{x}(k)$ at discrete time instants $K = 0,1,\dots,N$ and $\bar{u}(k)$ corresponds to the controls at $k = 0,1,\dots,N-1$. The function $\bar{f}_i(\bar{x}(k),\bar{u}(k))$ $i = 1,\dots n$ is assumed to be continuous in u_i $i = 1,\dots,m$ and have first partial derivatives in x_i $i = 1,\dots,n$.

As in the previous section, we introduce an adjoint system of equations $\bar{\phi}(k)$ and discrete Hamiltonian function $H(\bar{x}(k),\bar{u}(k), \bar{\phi}(k))$ such that

$$H\left(\bar{x}(k),\bar{u}(k),\bar{\phi}(k)\right) = \bar{\phi}(k)^T.\bar{f}\left(\bar{x}(k),\bar{u}(k)\right) \quad (6.2.17)$$
$$k = 0,1,2,\dots,N-1$$

For a fixed $\bar{x}(k)$ and $\bar{\phi}(k)$, $H\left(\bar{x}(k),\bar{u}(k),\bar{\phi}(k)\right)$ is defined to attain a local maximum at a point $\bar{u}^*(K) \in U$ if

$$H\left(\bar{x}(k),\bar{\phi}(k),\bar{u}^* k)\right) \geq H(\bar{x}(k),\bar{\phi}(k),\bar{u}(k)) \qquad (6.2.18)$$

for any point $\bar{u}(k)$ in the neighbourhood $\delta \in U$ f the point $\bar{u}^*(k)$.

The elements of $\bar{\phi}(k)$ are defined to satisfy the relation

$$\phi_i(k-1) = \frac{-\partial H\left(\bar{x}(k),\bar{u}(k),\bar{\phi}(k)\right)}{\partial x_i(k)} \qquad (6.2.19)$$

for $i = 1,\ldots,n$ and $k = 0,1,2,\ldots,N-1$ A necessary condition for optimality then states;

Theorem 6.2.4

Let the optimum control $\bar{u}^*(k)$, $(k=0,1,\ldots,N-1)$ exist and let the corresponding optimum trajectory $\bar{x}^*(k)$, $(k=0,1,\ldots,N)$ exist with the initial condition, $x(0) = \bar{a} \in \Omega$.

Then for $\bar{u} = \bar{u}^*(k)$, $k = 0,1,\ldots,N$ and $\bar{x} = \bar{x}^*(k)$ $k = 1,\ldots,N$ there exists a solution $\bar{\phi} = \bar{\phi}^*(k)$ $(k=1,\ldots,N-1)$ satisfying equation (6.2.19) and with the final condition

$$\bar{\phi}^*(N-1) = \bar{c}_o \qquad (6.2.20)$$

such that for a fixed $\bar{x} = \bar{x}^*(k)$ and $\bar{\phi} = \bar{\phi}^*$ the function

$$H\left(\bar{x}^*(k), \bar{\phi}^*(k), \bar{u}^*(k)\right) \geq H\left(\bar{x}^*(k), \phi^*(k), \bar{u}(k)\right) \qquad (6.2.21)$$

$k = 0,\ldots,N-1$, for any $\bar{u}(k)$ in the neighbourhood of the point $\bar{u}^*(k)$. When $k = N-1$ the function $H\left(\bar{x}^*(N-1), \bar{\phi}^*(N-1), \bar{u}\right)$ of $\bar{u}$ for $\bar{u} = \bar{u}^*(N-1)$ attains absolute maximum in the region U; i.e.,

$$H\left[\bar{x}^*(N-1), \bar{\phi}^*(N-1), \bar{u}^*(N-1)\right] \geq H\left[\bar{x}^*(N-1), \bar{\phi}^*(N-1), \bar{u}\right] \qquad (6.2.22)$$

for any $\bar{u} \in U$. Proof of the theorem is contained in [8]

It should be noted that although theorem 6.2.4 is quite similar to Pontryagni's maximum principle (theorem 6.2.1) for control systems described by ordinary differential equations, it is not an exact analogue of the latter. For the Hamiltonian in theorem 6.2.4. assumes a local minimum or stationary value on the optimal control trajectory rather than a global maximum. However, if the controls u_i enter the system linearly, the local maximum principle becomes the global maximum principle.

6.2.2.2. Extended Maximum Principle

Theorem 6.2.5.

Consider a function RH $\left[k, \bar{x}(k), \bar{u}(k) \right]$ defined by

$$RH\left[k,\bar{x}(k),\bar{u}(k)\right] = \bar{\phi}(k)^T.\bar{f}\,(\bar{x}(k),\bar{u}(k)) - \bar{\phi}(k-1)^T\bar{f}\;\bar{x}(k-1),\bar{u}(k-1)$$

$$= H\left[\bar{x}(k),\bar{\phi}(k),\bar{u}(k)\right] -| H\left[\bar{x}(k-1),\bar{\phi}(k-1),\bar{u}(k-1)\right]$$

$k = 1,\ldots,N-1$ such that (6.2.23)

$$RH\left[-1,\bar{x}(-1),\bar{u}(-1)\right] = 0 \qquad (6.2.24)$$

If $\bar{u}^*(k)$ and $\bar{x}(k)$, $k = 1,\ldots,N-1$ are such that

$$RH\left[k,\bar{x}^*(k),\bar{u}^*(k)\right] = \underset{\substack{\bar{u}(k)\in U \\ \bar{x}(k)\in\Omega}}{\text{Max}}\ RH\left[k,\bar{x}(k),\bar{u}(k)\right] \qquad (6.2.25)$$

then $\bar{u}^*(k)$, is the optimal control and $\bar{x}^*(k)$ the optimal trajectory: i.e. $F = \bar{c}^T.\bar{x}(N)$ is (Maximized).

6.2.2.3. Necessary and Sufficient Conditions

By combining theorems 6.2.4 and 6.2.5. one is able to formulate the result which expresses the necessary and sufficient conditions for optimality in the case of a linear system of equations defined by:

$$\bar{x}(k+1) = \bar{A}(k)\bar{x}(k) + \bar{B}(k)\bar{u}(k) \qquad (6.2.26)$$

$$k = 0,\ldots,N-1$$

where $\bar{A}(k)$ is an nxn matrix and $\bar{B}(k)$ an nxm matrix. Then in order that $\bar{u}^{*}(k)$, $k = 0,\ldots,N-1$, be an optimal control it is necessary and sufficient that

$$\bar{\phi}(k).\bar{B}(K).\bar{u}^{*}(k) = \underset{\bar{u}(k)\in U}{\text{Max}} \; \bar{\phi}(k)\bar{B}(k)\bar{u}(k) \qquad (6.2.27)$$

$$k = 0,\ldots,N-1$$

Just as the continuous maximum principle has found increasing use in system design, it is hoped that the discrete version will form a strong foundation upon which the optimal design of sampled-data systems will be based.

6.2.3. Relationship Between the Basic Theorems of Mathematical Programming and the Maximum Principle.

Let us consider a discrete dynamic system described by difference equation

$$\bar{x}(k+1) = \bar{x}(k) + f_{k}(\bar{x}(k),\bar{u}(k)) \qquad (6.2.28)$$

$k = 0,\ldots,N-1$ where $\bar{x}(k),\bar{u}(k)$ and $\bar{f}$ $\bar{x}(k),\bar{u}(k)$ have the same dimensions as those of section 6.2.2.

For a fixed initial state, x(o), the problem is to select controls $\bar{u}_{k}\in U$ $k = 1,\ldots,N-1$, such a performance index

$$F = c\cdot\bar{x}_{K} \qquad (6.2.29)$$

is maximized.

In the above subsections we have considered the posed problem as that of optimal control theory. In the following discussions, we shall consider it as a problem of mathematical programming. Consequently, the conditions of optimality shall be established by means of Kuhn and Tucker theorems[51] together with the extensions due to Karlin and others.[44]

We introduce a Lagrange function

$$\psi(\bar{x},\bar{u},\bar{\lambda}) = \bar{c}\bar{x}(N) + \sum_{K=0}^{N-1} \bar{\lambda}(k)\Big(\bar{x}(k+1)-x(k)-\bar{f}_k(\bar{x}(k),u(k))\Big) \tag{6.2.30}$$

where $\bar{\lambda}(k)$ is an nxl matrix.

<u>Theorem 6.2.6.</u>

If $\hat{\bar{x}},\hat{\bar{u}},\hat{\bar{\lambda}}$ is the saddle point of the Lagrange function; i.e.,

$$\psi(\bar{x},\bar{u},\hat{\bar{\lambda}}) \le \psi(\hat{\bar{x}},\hat{\bar{u}},\hat{\bar{\lambda}}) \le \psi(\hat{\bar{x}},\hat{\bar{u}},\bar{\lambda}) \tag{6.2.31}$$

for any $\bar{x}$, $\bar{\lambda}$ and $\bar{u}(k)\epsilon U$, then $\hat{\bar{u}}$ is the optimal control.

<u>Proof</u>: Using (6.2.30) the right-hand pair of the inequalities of (6.2.31) can be represented:[64]

$$c^T\hat{\bar{x}}(N) + \sum_{K=0}^{N-1} \hat{\bar{\lambda}}^T(k)\Big(\hat{\bar{x}}(k+1)-\hat{\bar{x}}(k)-f(\hat{\bar{x}}(k),\hat{\bar{u}}(k))\Big) \le$$

$$\bar{c}^T\hat{\bar{x}}(N) + \sum_{K=0}^{N-1} \bar{\lambda}^T(k)\Big(\hat{\bar{x}}(k+1)-\hat{\bar{x}}(k)-f(\hat{\bar{x}}(k),\hat{\bar{u}}(k))\Big)$$

Since the inequality must hold for any $\bar{\lambda}(k)$, it follows that $\hat{\bar{x}}(k),\hat{\bar{u}}(k)$ satisfy the system of equations (6.2.28).

And for any $\bar{x}(k)$ and $\bar{u}(k)$ satisfying (6.2.28) we have, on the basis of left-hand pair of inequalities

$$\bar{c}^T\bar{x}(k) \leq \bar{c}^T\hat{x}(k) \qquad \text{Q.E.D.}$$

If we assume that the function $f_K\{\bar{x}(k),\bar{u}(k)\}$ is differentiable with respect to $\bar{x}(k)$; then for $\bar{x},\bar{u},\bar{\lambda}$ to a saddle point to $\psi(\bar{x},\bar{u},\bar{\lambda})$, it is necessary that the following conditions be fulfilled.

$$\left.\begin{array}{l} \text{grad}_\lambda\psi = 0 \\ \text{grad}_x\psi = 0 \end{array}\right\} \quad \text{for } \bar{x} = \hat{x}, \quad \bar{\lambda} = \hat{\lambda} \text{ and } \bar{u} = \hat{u} \qquad (6.2.32)$$

$$\psi(\hat{x},\hat{u},\hat{\lambda}) = \max_{\bar{u}(k)\epsilon U} \psi(\bar{u},\hat{x},\hat{\lambda}) \qquad (6.2.33)$$

The first condition of (6.2.32) is equivalent to the requirement that $\bar{x}(k)$ and $\bar{u}(k)$ satisfy (6.2.28). While the second condition is equivalent to the difference equation, linear with respect to $\bar{\lambda}(k)$:

$$\bar{\lambda}(K-1)-\bar{\lambda}(k)-(\underset{x(k)}{\text{grad}}\ \bar{\lambda}^T(k)f(k))^T = 0 \qquad (6.2.34)$$

$$k= 0,\ldots,N-1$$

From condition (6.2.32) after differentiation of the Lagrange function (6.2.30) with respect to $\bar{x}(N)$, it follows that

$$\bar{\lambda}_{N-1} = -\bar{c} \qquad (6.2.35)$$

Let us introduce now, the Hamiltonian function

$$H_k\left[\bar{x}(k),\bar{u}(k),\bar{\lambda}(k)\right] = \bar{\lambda}^T(k).\bar{f}_k(\bar{x}(k),\bar{u}(k)) \quad (6.2.36)$$

Equations (6.2.28) and (6.2.34) may then be written in terms of the Hamiltonian function

$$\bar{x}(k+1) - \bar{x}(k) = \text{grad}_{\lambda(k)} H(k) \qquad (6.2.37)$$

$$\bar{\lambda}(k) - \bar{\lambda}(k-1) = - \text{grad}_{x(k)} H(k) \qquad (6.2.38)$$

with corresponding boundary conditions on $\bar{x}(o)$ and $\bar{\lambda}(N-1)$.

Condition (6.2.33) may be replaced by dropping the components which do not depend on $\bar{u}$, the requirements that $\bar{u}$ would supply the maximum of the sum

$$- \sum_{k=0}^{N-1} \hat{\lambda}^{T}(k) f_k \left[\hat{\bar{x}}(k), \bar{u}(k)\right]$$

which in turn may be replaced by the requirement of maximization of

$$- \hat{\bar{\lambda}}(k) f_k \left[\hat{\bar{x}}(k), u(k)\right]$$

Using equation (6.2.36), the following conclusion is arrived at:

Theorem 6.2.7.

For the point $\hat{\bar{x}}$, $\hat{\bar{u}}$, $\hat{\bar{\lambda}}$ to a saddle point of the function $\psi(\bar{x},\bar{u},\bar{\lambda})$, it is necessary that the following conditions be fulfilled:

(i) the sequences $\hat{\bar{x}}(k)$, $\hat{\bar{\lambda}}$, $k = 0,\ldots,N$ be solutions to the Hamiltonian system (6.2.37) and (6.2.38).

(ii) at each time, k the function $H_k(\hat{\bar{x}},\bar{u},\bar{\lambda})$ reaches a maximum for $\bar{u}(k) = \hat{\bar{u}}(k)$.

If all the functions $\bar{f}_k$ are linear, then the saddle point always exists.[44] Consequently $\hat{x}$, $\hat{u}$, supplies the optimum for the initial problem. Furthermore, for linear discrete system, the conditions for theorem 6.2.7, equivalent to the maximum principle are necessary for optimality.[69] Kuhn and Tucker have shown that conditions of theorem 6.2.7, and consequently, the maximum principle, are also sufficient.[51]

The relationship between maximum principle and Kuhn-Tucker conditions has also been derived by A.I. Propoi[66] Mangasarian and Fromovitz,[57] have, on the other hand used the generalized Fritz John necessary optimality criteria to establish between maximum principle and mathematical programming.

6.2.3.1. A General Formulation

In this section, we assume a similar approach for the establishment of a theorem of the type of maximum principle, but in the presence of additional constraints on the phase coordinates and on the selection of controls at each time depending on the values the phase coordinates have reached at the times in question. The function to be maximized is also of a more general nature.

Consider the problem:

$$\text{Max} \quad F = \sum_{k=0}^{N-1} \theta_k \left(\bar{x}(k), \bar{u}(k) \right) + \theta_N(x(N)) \tag{6.2.39}$$

subject to $\bar{x}(k+1) = \bar{x}(k) + f_k\left[\bar{x}(k),\bar{u}(k)\right]$ (6.2.40)

$k = 0,1,\ldots,N-1,\ x(o)=\bar{a}$

$G_k\left[\bar{x}(k)\right] \geq 0 \quad k = 0,\ldots,N$ (6.2.41)

where $\bar{\theta}(k)$ is assumed to be concave

$\bar{L}\ \left[\bar{x}(k),\bar{u}(k)\right] = 0 \quad k = 0,\ldots,N-1$

(6.2.42)

$\bar{u}(k)\in U \quad k = 0,\ldots,N-1$ (6.2.43)

The following Lagrange function is introduced:

$$\psi(\bar{x},\bar{u},\bar{\lambda},\bar{\mu},\bar{\rho}) = \sum_{k=0}^{N-1} \theta_k\left[\bar{x}(k),\bar{u}(k)\right] + \theta_N(\bar{x}(N)) + \sum_{K=0}^{N-1} \bar{\lambda}_k\left(\bar{x}(k+1)-\bar{x}(k)-f_k(\bar{x}(k),\bar{u}(k))\right) + \sum_{k=0}^{N} \bar{\mu}_k.G_k\left[\bar{x}(k)\right] + \sum_{k=0}^{N-1} \rho_k L_k\left[\bar{x}(k),\bar{u}(k)\right]$$

(6.2.44)

It has been shown that the existence of a saddle point to the above Lagrange function, is guaranteed if f_K is linear; and that at least one trajectory exists for which $G_K(\bar{x}(k)) > 0$. [15,44].

The necessary and sufficient conditions for $\hat{x},\hat{u},\hat{\lambda},\hat{\mu},\hat{\rho}$ to be a saddle point may be formulated in the form:

$\text{grad}_\lambda\ \psi = \text{grad}_\mu\ \psi = \text{grad}_x\ \psi = 0$ (6.2.45)

$\hat{\mu} \geq 0; \quad \hat{\mu}\ \text{grad}_\mu\ \psi = 0; \quad \text{grad}_\mu\ \psi \geq 0$ (6.2.46)

$\psi(\hat{x},\hat{u},\hat{\lambda},\hat{\mu},\hat{\rho}) = \underset{\bar{u}(k)\in U}{\text{Max}}\ \psi(\hat{x},\hat{u},\hat{\lambda},\hat{\mu},\hat{\rho})$ (6.2.47) where all the derivatives are calculated at the point $\hat{x},\hat{u},\hat{\lambda},\hat{\mu},\hat{\rho}$.

Condition (6.2.47) is equivalent to the requirement that the controls, $\bar{u}(k)$ supply the minimum to the Hamiltonian

$$H_k = \bar{\lambda}(k)^T.\bar{f}_k\left(\bar{x}(k),\bar{u}(k)\right) - \rho_k\left(\bar{x}(k),u(k)\right) - \bar{\rho}(k)^T L_k\left(\bar{x}(k),\bar{u}(k)\right) \qquad (6.2.48)$$

among all $\bar{u}(k)$, $k = 0,\ldots,N-1$.

We see, therefore, that the assumption of the existence of a relationship between the saddle point of a Lagrange function and the maximum achieved in the initial problem similar to theorem 6.2.6. remains in force, even for the general case considered above. This observation leads to the following theorem:

Theorem 6.2.8.

For optimality of control $\hat{u}$ and trajectory $\hat{x}$ in the problem (6.2.28), (6.2.29) with $\bar{u}(k)\epsilon U$, (6.2.41) and (6.2.42) it is necessary that they, together with the Lagrange factors $\bar{\lambda},\bar{\mu},\bar{\rho}$ satisfy the following relations:

$$\hat{\bar{x}}(k+1) - \hat{\bar{x}}(k) = \text{grad}_{\lambda(k)}H_k \quad , \quad x_o = x(o) \qquad (6.2.49)$$

$$\bar{\lambda}-\bar{\lambda}(k-1) = -\text{grad}_{x(k)}H_k + \hat{\bar{\mu}}(k)\ \text{grad}\ G_k(\hat{\bar{x}}(k)) \qquad (6.2.50)$$

$$\bar{\lambda}_{N-1} = -\text{grad}\ \psi_N(\hat{\bar{x}}(k)) \qquad (6.2.51)$$

$$\bar{\mu}(k) \geq 0 \qquad (6.2.52)$$

$$\bar{\mu}(k) G_k(\bar{x}(k)) = 0 \qquad (6.2.53)$$

$$\bar{G}_k(\bar{x}(k)) \geq 0 \qquad (6.2.54)$$

If the above conditions are satisfied, then the controls $\hat{\bar{u}}(k)$ supply the minimum to the Hamiltonian H_k for $\bar{u}(k) \in U$ $k = 0,\ldots,N-1$.

A.A. Pervozranskiy[64] has extended the above observations to formulate optimality conditions for problems of the minimax type. His main conclusions here are analogous to those obtained by Dubovitskiy[21] and Milyutin for the continuous case.

CHAPTER 7

DECOMPOSITION

7.0. Many practical mathematical programming problems are made up of almost independent sub-problems tied together with a common objective function and one or two sets of common constraints. Some of these problems are quite large, thus making heavy demand on computation time.

A possible way of handling such large problems is to "decompose" them into the almost independent "sub-problems" and the 'master' problem which ties together the sub-problems. The sub-problems on the one hand, and the 'master' problem on the other hand, are then solved in a way that takes into account the interaction between the two. After a finite number of iterative steps an optimal solution to the original problem is found (if one exists).

The decomposition principle was inspired by Ford and Fulkenson for solving multi-stage commodity network problems; and developed by Dantzig and Wolfe to solve a certain type of large linear programming problems.[15]

Since then, other techniques for solving both linear and non-linear convex (and non-convex) programmes have been proposed. A number of these will be discussed here.

7.1. Dantzig-Wolfe Decomposition Principle [5,15]

Consider a general linear programming problem of the form

$$\text{Maximize} \quad F(x) \;=\; \sum_{i=1}^{r} c_j x_j \qquad (7.1.1)$$

subject to

$$\begin{array}{lcl}
\underline{\hat{A}}_1\bar{x}_1 + \hat{A}_2\bar{x}_2 + \ldots + \hat{A}_j\bar{x}_j + \ldots + \hat{A}_r\bar{x}_r & = & b \\
A_1\bar{x}_1 & = & \bar{b}_1 \\
\qquad \bar{A}_2\bar{x}_2 & = & \bar{b}_2 \\
\qquad\qquad \bar{A}_j\bar{x}_j & = & \bar{b}_j \\
\qquad\qquad\qquad \bar{A}_r\bar{x}_r & = & \bar{b}_r \qquad (7.1.2)
\end{array}$$

$$\bar{x}_j \;\geq\; 0 \qquad (7.1.3)$$

where b and $\bar{b}_j$ are m-component vectors; $\hat{A}_j$ (the common rows) and $\bar{A}_j$ are $m_j.n_j$ matrices.

Dantzig and Wolfe have shown how the above problem can be decomposed into a number of linear sub-programmes, each of which is of a much smaller size than the original. The sub-programmes are coupled together by the first equation of 7.1.2

The set of points $\bar{x}_j \geq 0$ which satisfy $\bar{A}_j\bar{x}_j = \bar{b}_j$ is a closed convex set with only a finite number of extreme points. If the set is strictly bounded, it is a polyhedron; so that any point on the convex set can be represented as a convex combination of the extreme points.

The extreme points of the convex set are denoted by $\underline{\hat{x}}k_j$, $k = 1,\ldots,h_j$, $j = 1,\ldots r$. Then any feasible solution, $\bar{x}_j$ to $\bar{A}_j\bar{x}_j = \bar{b}_j$ can be written

$$\bar{x}_j = \sum_{k=1}^{h_j} \rho_{k_j} \hat{\underline{x}}_{k_j} \qquad (7.1.4)$$

$$\sum_{k=1}^{h_j} \rho_{k_j} = 1 \qquad (7.1.5)$$

$$\rho_{k_j} \geq 0 \qquad k = 1,\dots,h_j \qquad (7.1.6)$$

Hence, any solution $\bar{x}_j$, $j = 1,\dots,r$ solving (7.1.1) through to (7.1.3) can be re-expressed in terms of k_j:

$$\text{Maximize} \quad F(\rho_{k_j}) = \sum_{j=1}^{r} \sum_{k=1}^{h_j} \rho_{k_j} c_j \hat{\underline{x}}_j \qquad (7.1.7)$$

$$\text{subject to} \quad \sum_{j=1}^{r} \rho \sum_{k=1}^{h_j} \rho_{k_j} \bar{A}_j \hat{\underline{x}}_{k_j} = \hat{b} \qquad (7.1.8)$$

$$\sum_{k=1}^{h_j} \rho_{k_j} = 1, \quad j = 1,\dots,r \qquad (7.1.9)$$

$$\rho_{k_j} \geq 0 \quad j = 1,\dots,r \qquad (7.1.10)$$

The new (equivalent) problem is called the *full master problem*, while ρ_{k_j} are referred to as *proposals* from the sub-problem to the master problem. The set of constraints $\sum_{k=1}^{h_j} \rho_{k_j} = 1$ is sometimes referred to as *convexity constraint* for the sub-problems.

The set of constraints (7.1.8) through to (7.1.10) is equivalent to the constraints (7.1.2) and (7.1.3). Every feasible solution to (7.1.1) through to (7.1.3) determines a set of $\rho_{k_j} \geq 0$ which satisfy (7.1.8), (7.1.9) and (7.1.10); and vice versa. Note however that a set of $\rho_{k_j} \geq 0$ satisfying (7.1.8),(7.1.9) and (7.1.10), uniquely determines a set of $\bar{x}_j$ satisfying (7.1.2) and (7.1.3);

whereas a set of $\bar{x}_j$ which satisfy (7.1.2) and (7.1.3) may not uniquely determine $\rho_{k_j} \geq 0$ satisfying (7.1.8), (7.1.9) and (7.1.10). There will be at least one such ρ_{k_j}.

If the optimal solution to (7.1.7), (7.1.8) (7.1.9) and (7.1.10) is $\rho^*_{k_j}$, $k = 1,\ldots,h_j$; $j = 1,\ldots,r$ then optimal solution to the original problem is

$$x^*_j = \sum_{k=1}^{h_j} \rho^*_{k_j} \underline{\hat{x}}k_j \qquad j = 1,\ldots,r. \tag{7.1.11}$$

In general, the new linear programming problem (7.1.7 - 7.1.10) has the advantage of possessing fewer constraints than the original problem. Ho ever, it (the new problem) usually has more variables; for the number of extreme points of the convex set of feasible solutions to $\bar{A}_j\bar{x}_j = \bar{b}_j$ is bound to be greater than the number of components in $\bar{x}_j$.

The main advantage of the new formulation (7.1.7 - 7.1.10) is that it is not necessary to generate every extreme point $\underline{x}_{k_j}$ before the problem is solved; rather, these are generated when needed in the course of the solution.

The extreme points are generated as follows: Let the constraints (7.1.8) and (7.1.9) be written as

$$\sum_{j=1}^{r} \sum_{k=1}^{h_j} \rho_{k_j} \bar{q}_{k_j} = \bar{b} \tag{7.1.12}$$

we see that

$$\bar{q}_{k_j} = \left[A_j \underline{\hat{x}}_{k_j}, \bar{e}_j\right]^T \text{ and } \bar{b} = \left[\hat{b}, 1'\right]^T$$

where $\bar{e}_j$ is the jth unit vector $j = 1,\ldots,r$, and 1' is the sum vector having r components.

Suppose that an initial basis, $\bar{B}$, for (7.1.12) exists, where $\bar{B}$ is an (m+r) matrix. Let $\bar{}_b$ denote the vector containing basic variables and $\underline{\hat{c}}_b$ a vector containing prices in the basis.

Let $\bar{\sigma} = \bar{\sigma}_1, \bar{\sigma}_2{}^T$ where $\bar{\sigma}_1$ contains the first m components of $\bar{\sigma}$ and $\bar{\sigma}_2$ contains the last r components.

Then the relative cost is

$$\underline{\bar{c}}_{k_j} = \underline{\hat{c}}_b B^{-1}\bar{q}_{k_j} - c_j \underline{\hat{x}}_{k_j}$$

$$= (\bar{\sigma}_1 A_j - c_j)\underline{\hat{x}}_{k_j} + \bar{\sigma}_{2j} \qquad (7.1.13)$$

where $\bar{\sigma}_{2j}$ is the jth component of $\bar{\sigma}_2$.

To test whether the given basic feasible solution is optimal, min $\underline{c}_{k_j}$ must be computed over all k,j; i.e.

$$\min \underline{c}_{k_j} = \min \; \min_k(\underline{c}_{k_1}), \min_k(\underline{c}_{k_2})\ldots\min_k(\underline{c}_{k_r}) \qquad (7.1.14)$$

If (7.1.14) is non-negative, the given solution is optimal, otherwise more iterations are made.

Observe (from 7.1.13) that for a given j, $\min_k(\underline{c}_{k_j})$ occurs at the extreme point of the convex set of feasible solutions to $\bar{A}_j\bar{x}_j = \bar{b}_j$. Consequently, since each extreme point $\underline{\hat{x}}_{k_j}$ is a basic feasible solution to $\bar{A}_j\bar{x}_j = \bar{b}_j$, $\min_k(\underline{c}_{k_j})$ is σ_{2j} plus the optimal value of the objective function for the linear programming problem

$$\text{Min} \quad f_j = (\bar{\sigma}_1 A_j - \bar{c}_j)\bar{x}_j$$
$$\text{Subject to} \quad \bar{A}_j \bar{x}_j = \bar{b}_j$$
$$\bar{x}_j \geq 0 \qquad (7.1.15)$$

Moreover, an optimal basic solution to (7.1.15) gives an extreme point $\hat{\underline{x}}_{k_j}$ for which the corresponding $\underline{c}_{k_j}$ has the smallest value over k. This $\hat{\underline{x}}_{k_j}$ can then be used to generate the corresponding $\bar{A}_j \hat{\underline{x}}_{k_j}$, $\bar{c}_j \hat{\underline{x}}_{k_j}$ and $\bar{q}_{k_j}$.

Problem (7.1.15) is the sub-problem. There are r such sub-problems for the general case considered.

To determine min $(\underline{c}_{k_j})$ over all k,j,r linear programming sub-problems of the form (7.1.15) are solved. Let f_j^* be the optimal value of f_j for the jth such sub-problem. Then

$$\underset{\text{all } k,j}{\text{Min}} (\underline{c}_{k_j}) = \underset{j}{\text{Min}} (f_j^* + \bar{\sigma}_{2j})$$
$$= f_j^* + \bar{\sigma}_{2s} \qquad (7.1.16)$$

Let $\hat{\underline{x}}_{rs}$ be an optimal extreme point of (7.1.15) for j = s; then $\bar{q}_{rs} = \left[\bar{A}_s \hat{\underline{x}}_{rs}, e_s\right]$ enters the basis at the next iteration, and the price associated with $\bar{q}_{rs}$ is $\bar{c}_s \hat{\underline{x}}_{rs}$. Thus a vector to enter the basis has been generated. We now return to the master problem; new values of $\bar{B}^1$, $\bar{\sigma}$ and $\bar{\ }_b$ are obtained. These are then used to obtain a new set of objecfive functions for the r sub-problems of the type (7.1.15). The solution to the r sub-problems give the next vector to enter the basis in the master problem

(7.1.7 through to 7.1.10). This process is continued until an optimal solution is obtained.

Note that from the start, we do not have to store the full master programme (7.1.7 - 7.1.10). Rather, all the columns are dropped, except those in the basis and the new columns (eg. $\bar{q}_{rs}$) added in the course of the iterations. A programme thus obtained is called the restricted master programme.

The theory of the simplex method guarantees that an optimal solution will be obtained in a finite number of steps. Either the standard or the revised simplex method can be used to solve both the master and the sub-problems.

Dantzig and Wolfe used the revised simplex algorithm. This algorithm has been used successfully for solving a large variety of large linear programming problems, especially for the case with very few common rows.

However, as the number of common rows increases, the problem becomes more difficult; and no systematic rules are available for tackling such problems. Here is an area where further research work is needed.

7.2. Other Methods for Decomposing Linear Programming Problems:

Apart from the Dantzig-Wolfe "decomposition principle" several other methods have been developed. Some of these are briefly considered here. They include Beale's 'pseudo-basic' variable method, Abadie-Williams 'dual and parametric' method, decomposition by dynamic programming and Kron's method of diakoptics.

(a) Beale's pseudo-basic variable method:[39]

Whereas the Dantzig-Wolfe algorithm solves the primal problem, Beale's algorithm is designed for a dual problem of a more specialized structure.

The essential idea of Beale's method is that the linking variables are regarded as parameters. These parameters are assigned specific values, which then change after each pivotal operation.

The method of solution is essentially simplex, except for the following modification. When, after a number of iterations, one of the basic variables becomes zero, the basic variable is not made non-basic in place of the parameter, as this would spoil the special structure of the problem. Instead a transformation of the parameter (which would have otherwise entered the basis) is made whereby if one of the other parameters is changed, the zero-valued basic variable is not changed (i.e. not removed from the basis).

The zero-valued basic variable is referred to as "pseudo-basic": hence the name of the algorithm.

Not much computational experience with the method has been reported, although Beale claims that it would be more efficient than the Dantzig-Wolfe method for certain specially structured problems.

(b) The Dual and Parametric Method:[39]

Abadie and Williams have also proposed a dual decomposition algorithm. It differs from the Dantzig-Wolfe method in the manner of choosing the vector to introduce into the basis. The latter does the selection at the sub-problem level, while the Abadie-Williams method employs a selection procedure which does not require the vectors (to be selected) to be explicit. Here lies the advantage of the Abadie-Williams technique; for it allows certain parametric linear programmes to be solved by decomposition. A detailed exposition of the method may be found in [39]

(c) Decomposition by Dynamic Programming:[61]

G.L. Nemhauser has proposed a decomposition scheme derived by a dynamic programming approach. This results in a series of parametric linear programmes whose recursive solution yields the solution to the original programming problem. It has the advantage that each sub-programme need be solved only once.

Consider a parametric linear programme of the original problem:

$$\text{Max:} \quad F_{r-1}(Y_{r-1}) = \sum_{j=1}^{r-1} c_j \bar{x}_j \tag{7.2.1}$$

subject to

$$\begin{array}{lllllr}
\hat{A}_1 x_1 + & \hat{A}_2 x_2 + \dots + & \hat{A}_j \bar{x}_j + & \dots . + \hat{A}_{r-1}\bar{x}_{r-1} & = & Y_{r-1} \\
\bar{A}_1 \bar{x}_1 & & & & = & \bar{b}_1 \\
 & \bar{A}_2 \bar{x}_2 & & & = & \bar{b}_2 \\
 & & & & & \vdots \\
 & & \bar{A}_j \bar{x}_j & & = & \bar{b}_j \\
 & & & & & \vdots \\
 & & & \bar{A}_{r-1}\bar{x}_{r-1} & = & \bar{b}_r
\end{array} \tag{7.2.2}$$

$$\bar{x}_j \geq 0 \tag{7.2.3}$$

From the theory of parametric linear programming it can be established that F_{r-1} is a piecewise linear, concave function of Y_{r-1}. The points at which F_{r-1} changes slope correspond to the values of Y_{r-1} at which there is a change in the basis required to maintain primal feasibility.

Suppose that F_{r-1} were known for all values of Y_{r-1} which satisfy

$$\hat{A}_r \bar{x}_r + Y_{r-1} = \bar{b} \tag{7.2.4}$$

$$\bar{A}_r x_r = \bar{b}_r \tag{7.2.5}$$

$$x_r \geq 0 \tag{7.2.6}$$

Then from the dynamic programming principle of optimality (Chapter 6) it follows that

$$F_r = \underset{x_r, Y_{r-1}}{\mathrm{Max}} \quad c_r x_r + F_{r-1}(Y_{r-1}) \qquad (7.2.7)$$

subject to (7.2.4),(7.2.5) and (7.2.6). The solution of equation (7.2.7.) requires the maximization of a pricewise linear constraints. To solve (7.2.7) as a linear programme, the pricewise linear functions are replaced by smooth linear functions.

The sub-programme from which $F_{r-1}(Y_{r-1})$ is determined can be decomposed in the same way as the original problem. Applying this decomposition scheme r times, the following recursion relations are obtained.

$$F_j(Y_j) = \underset{x_j, Y_{j-1}}{\mathrm{Max}} \quad c_j x_j + F_{j-1}(Y_{j-1})$$

$$\text{subject to} \quad \begin{aligned} \hat{A}_j \bar{x}_j + Y_{j-1} &= Y_j \\ \bar{A}_j \bar{x}_j &= \bar{b}_j \\ x_j &\geq 0 \quad j = 1, \ldots r \end{aligned} \qquad (7.2.8)$$

where $F_o = 0$, $Y_o = 0$ and $Y_r = \hat{b}$

Each of the r successive linear programmes (except the first) have (n_j+m) variables and (m_j+m) constraints. In each sub-problem, however, additional variables and constraints are required to take care of the piecewise linear portions of the objective functions.

The first problem (i.e. j=1) contains n_1 variables and (m_1+m) constraints; and there are no piecewise linear functions.

For $j = 1,\ldots,r-1$, Y_j are parameters, so parametric linear programming algorithm must be used. However, since $Y_r = \hat{b}$, the last optimization need not be parametric unless a sensitivity analysis or $\hat{b}$ is desired.

A major advantage of this method is its theoretical simplicity. Similar schemes for decomposing quadratic and convex programming problems are possible extensions.

However, no experience with this method on large programmes is available, since the method has only been proposed quite recently; and it is hoped that further research in the field will be forthcoming.

(d) Of some particular interest to electrical engineers in a decomposition procedure proposed by G. Kron [50]. This is based on Kron's extensive work on the application of tensor methods to piecewise solution of large electrical networks: the interconnected system is first 'torn' into small subdivisions, each of which is solved as if the other ones did not exist. The solutions to the subdivisions are then combined in a systematic manner and modified to take the interconnections into account.

Kron has developed the idea of meshes and junction pairs, which together give rise to the coneept of orthogonal networks.

The decomposition procedure proposed by Kron is based on the topological analogue between the concept of orthogonal networks and the general equations of the linear programming problem (at least for the transportation or assignment type problems); and on the equivalence between the Simplex method and the process of orthogonal transformations.

Kron has applied the procedure to obtain a solution to a simple transportation problem[50]. But as he himself admits, a lot of research effort is still required before his procedure can be systematized to a worthwhile algorithm capable of handling a linear programming problem of a meaningful size.

7.3. Decomposition of Non-Linear Programmes

Decomposition of non-linear programmes is receiving more attention, now that efficient methods of solving non-linear programming problems are available (Chapter 5). Notable contributions in this field are due to Rosen, Fromovitz and Zangwill.[82]

Rosen has successfully applied his method of 'gradient projection' (Chapter 5) to 'partition' a

non-linear convex problem in linear constraints .
He has also proposed a way of optimizing a general convex programming in convex constraints[39].

In a recent study Fromovitz shows that a 'randomized strategy'; e.g. a procedure whereby one chooses a mixture of different solutions with specified probabilities may be better, on the average, than any non-randomized strategy, especially if some constraints need not be satisfied for each individual component of the strategy provided they are satisfied on the average[31].

According to this approach of decomposition the common row constraints are treated as ones that need only be satisfied on the average. On the other hand, the constraints within the sub-problems must be satisfied for each component of the strategy.

Fromovitz's work is of theoretical interest, especially for cases involving decomposible, non-convex, non-linear programming problems. But it is hoped that some of these ideas will soon be incorporated into an effective computational algorithm.

Zangwill, in his recent paper, has proposed two algorithms that may be useful in solving very large non-linear programming problems. Instead of solving the given problem, several small non-linear programming sub-problems are solved.

An important feature of the algorithms is that, under moderate regularity conditions, if the original problem has an optimal solution, only a finite number of sub-problems need be solved.

These are essentially large-step algorithms (methods of feasible directions - Chapter 5). However, the constraints are classified into 'tight' or 'slack' ones. Given any feasible point $\bar{x}$, if for some j, $G_j(\bar{x}) = 0$ then the constraint is said to be 'tight', on the other hand if $G_j(\bar{x}) \leq 0$ the constraint is said to be 'slack'.

Let a constant $\epsilon > 0$ be defined; then any slack constraint such that $G_i(\bar{x}) < \epsilon$ is said to be 'close'.

Using the above classification of the constraints a general procedure of the algorithms is as follows:

Using the method of feasible directions, a sequence of feasible points $\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^k$ is generated. At $\bar{x}^k$ the constraints are checked to see which ones are tight and which ones are close.

The sub-problem of optimizing the original objective function but subject to only those constraints that are tight or close at $\bar{x}^k$ are solved. An optimal point to the sub-problem is thus obtained. Point $\bar{x}^{k+1}$ is then generated for the original problem and the process of sub-problem optimization repeated. The process converges after solution of a finite number of sub-problems.

Although no large scale problems have been tried, this method appears to be quite promising. But as is the case with many other methods of feasible directions, convergence may be hampered by 'jamming'. This may occur when the algorithm repeatedly leaves a boundary, almost immediately bumps into another boundary, and then returns to the first boundary. Alternatively, the same sub-problem may be solved over and over again. Furthermore, jamming may result any time when, in determining the direction of move, the boundaries in the immediate neighbourhood are neglected.

Several techniques of avoiding jamming have been proposed. The ϵ-perturbation is such an approach: the boundaries in an $\epsilon > 0$ neighbourhood of the point are considered when determining the next direction to move. Consequently, a path for at least a distance of ϵ from the boundary is assured. Any path with a distance less than ϵ from the boundary is thus avoided.

PART II

Applications

CHAPTER 8

OPTIMAL OPERATING POLICIES OF WATER RESOURCES

In general water resources from a reservoir can be used in many different ways for different beneficial purposes: e.g. hydro-electric energy generation, irrigation, flood control and protection, and improvement of navigation. With all these uses in mind, it becomes necessary to determine the 'best' method of operating the reservoir in such a manner as to derive the maximum overall benefit (economic and/or social) subject to the physical or operating limitations of the reservoir in question.

In this chapter, simple mathematical models for digital computer studies are developed to elucidate the relations between the variables pertaining to irrigation and hydro-electric energy uses. The models incorporate some of the chief factors affecting the efficiency of a multi-purpose operation for hydro-electric energy generation and increasing agricultural productions.

A method of 'feasible directions' (Chapter 5) is used for the solution of the models. The application of a dynamic programming algorithm (Chapter 6) to the problem is also discussed.

In the last sections of the chapter some modifications and elaborations to the model considered are discussed, including extensions to account for stochastic streamflow, storage capacity and draft from the reservoir.

8.2. Statement of the Problem

Three simple models will be used to illustrate "complementarity" between hydro-power production and agriculture water supply.

The objective here is to determine a satisfactory compromise between electricity and agricultural production based on economic and hydrological analysis of the costs and benefits associated with each demand.

The degree of complementarity attainable will, of course, depend upon topographic, hydrological and meteorological conditions together with the economic factors prevailing in the particular country or region considered. Some of these include the size and shape of the reservoir; the magnitude of the natural inflow and how this varies from season to season, and from year to year. Furthermore maximum agricultural productivity is possible only if the plants are watered at a rate which nearly approximates the rate of evaporation; the rate of evaporation is in turn determined by the meteorological and climatic conditions which again vary from season to season. Electrical energy production is, on the other hand, determined primarily by economic considerations: e.g. the type of industries; and the degree of industrial development of the country or part of the country in question; and the prevalence of other forms of primary source of energy (oil or gas).

In most cases, electrical energy demand is unform throughout the year.

In general, full complementarity between agricultural and power generation is not attainable because the seasonal water demand patterns for both uses are different.

For the models considered the following conditions prevail. During the warm period when crop growing conditions are favourable, evaporation rates are high; consequently, irrigation water requirements are large. The demand of electricity, on the other hand, does not change appreciably from season to season.

In addition to the above difference in demand patterns, the distribution of river flow is non-uniform throughout the year. Thus in a typical year about two-thirds of the annual flow occurs in August, September and October. In the April, through to July period, when conditions for crop growth are favourable, the natural river flow is low. Table8.1 illustrates percentage river-flow distributions, together with demand patterns for electric energy produ:tion and irrigation respectively.

8.2.1. Model 1:

(a) Essential Structure: The model considered here is a very simple one with a single reservoir. Stored water passes through the turbines and releases electrical energy. The water then flows downstream and is diverted through the headgates of the irrigation system. A schematic diagram is given in Fig. 8.1.

(b) Assumptions: The model has been formulated with the following assumptions in mind:

(i) Quantity of irrigation water is preassigned ·

(ii) Demand for electricity is uniform throughout the year.

(iii) Natural inflows into the reservoir are known for the entire period of study.

(iv) The reservoir storage and natural inflows are such that the demand for both electricity and irrigation can be met throughout the period of study.

(v) Depletion of water through evaporation is accounted for.

(vi) The objective function is a quadratic function.

(vii) The average head on the turbines is a known preassigned value.

(viii) Effect of head variations is neglected.

(ix) The volume of water released during any period is less than or equal to the contents of the reservoir at the beginning of the period plus the flow into the reservoir during the period.

(x) The contents of the reservoir at the beginning of any period is less than or equal to the amount left over from the previous period.

(c) Constraints: Several physical and operational constraints are applicable to each period of study. These include:

(i) The volume of water released from the reservoir must be sufficient to meet the period's irrigation demand, where the latter is a proportion, α_t, of the annual irrigation demand - values of α_t are given in Table 8.2.

(ii) The amount of electricity generated at each period, t, must be at least a specified proportion, β_t, of the annual energy output E, where the latter is expressed in terms of units of river flow. The twelve values of β_t are given in Table 8.2.

(iii) The sum of the volume of water released from the reservoir for each of the periods must be equal to a preassigned value, this being the total volume of water allowable for the entire period of study.

(iv) Volume of water released at any period must take non-negative values only.

(d) Problem Formulation: With the above assumptions and constraints in mind the mathematical programming problem may be stated as

Maximize : $K_1E + K_2E^2$ (8.2.1)

subject to : $x_i \geq \alpha_i A$ (8.2.2)

$x_i \geq \beta_i \ (K_1E + K_2E^2)$ (8.2.3)

$\Sigma x_i = 80$ (8.2.4)

$x_i \geq 0 \quad i = 1,\ldots,12$ (8.2.5)

where E is the total electric energy demand for the year; A is the level agriculture required; x_i the release during month i; α_i and β_i represent distribution coefficients of demand for irrigation and power demand respectively; K_1 and K_2 are conversion factors which transform power demand in kilowatt-hours into units of river-flow.[56]

Seven levels of A were used ranging from 50 m.c.m. to 80 m.c.m*

It should be emphasized that the numerical values of α_i and β_i, i = 1,2,...,12 are only rough estimates of the demand patterns. These would naturally vary depending on the type of model considered.

8.2.2. Model 2:

Model 2 is essentially similar to Model 1 except that in the former the objective function has been expanded

* milliards of cubic meters. Values refer to those of Aswan Dam, Egypt.

TABLE 8.1.

Seasonal Distribution of Inflows to the Reservoir and Water Demands for Irrigation and Power Generation for the Models investigated*

Season	Inflow to reservoir.	Hydro-power demand	Irrigation demand
January-March	9	28	15
April-June	3	25	34
July-September	53	22	35
October-December	35	25	16

* Figures represent percentage of annual supply and demand.

TABLE 8.2.

Distribution Coefficients of Water Demand for Irrigation and for Electric Energy: (values of β_t and α_t).

Period	Irrigation Coeff.	Elec. Energy Coeff.
1	.027	.093
2	.033	.094
3	.066	.093
4	.101	.089
5	.149	.083
6	.138	.078
7	.114	.074
8	.101	.072
9	.094	.073
10	.081	.078
11	.046	.084
12	.026	.089

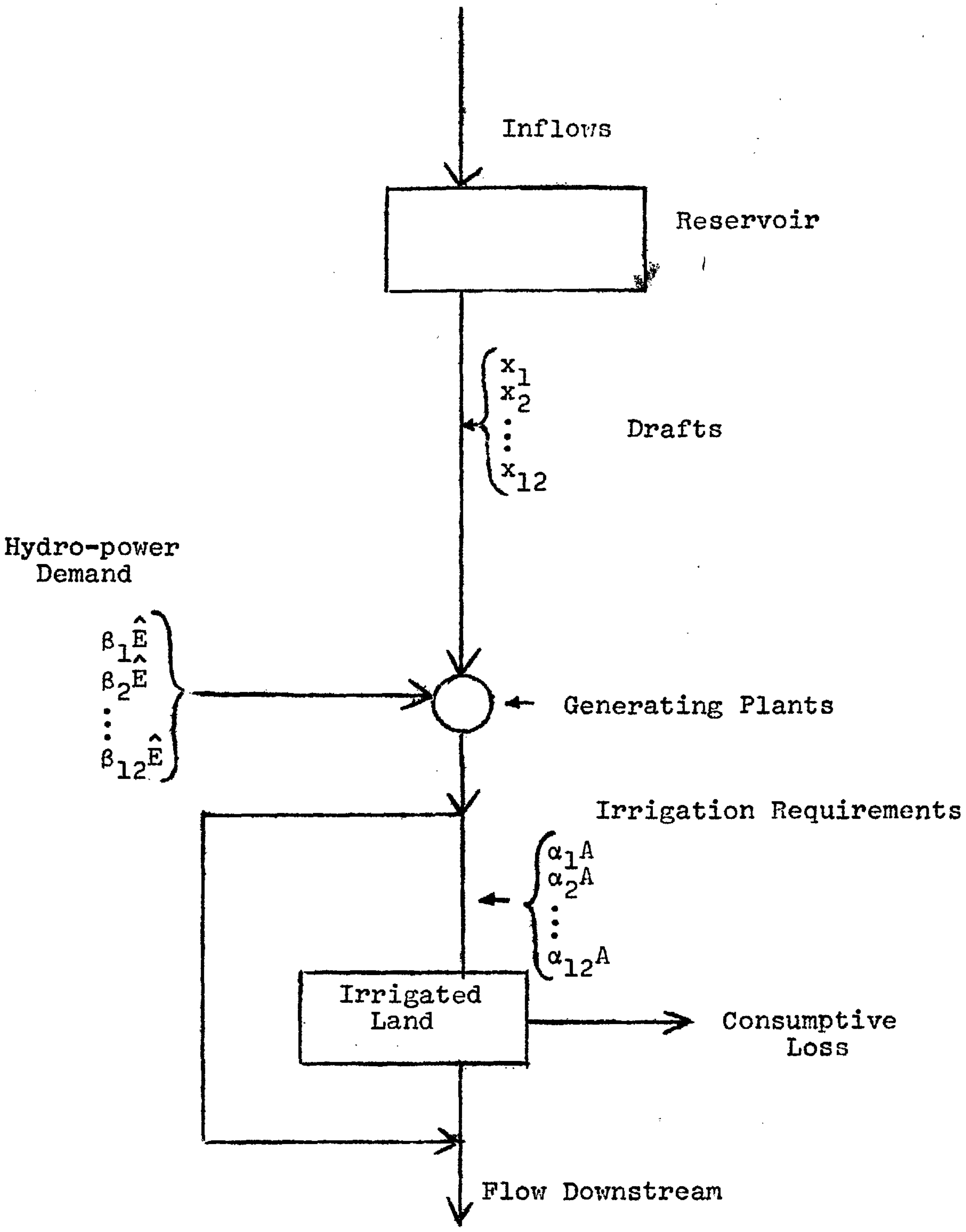


Fig. 8.1. Schematic Outline of the System for Models 1 and 2.

to include a performance index of the benefit which accrues from agricultural production. The problem can then be formulated as:

$$\begin{aligned} &\text{Maximize} \quad && \rho_\alpha A + \rho_e(K_1E + K_2E^2) && (8.2.6)\\ &\text{subject to} \quad && x_i \geq \alpha_i A && (8.2.7)\\ & && x_i \geq \beta_i(K_1E+K_2E^2) && (8.2.8)\\ & && x_i = 80 \quad i = 1,2,\ldots,12 && (8.2.9) \end{aligned}$$

where ρ_α and ρ_e are preference coefficients for agriculture and hydro-power respectively. Several numerical values of ρ_α and ρ_e have been used. Here again, it must be emphasized that these values are not definitive: rather they are used to illustrate the technique of analysis.

The preference coefficients ρ_α and ρ_e are proportional to the benefits arising from unit increment in agricultural and power production respectively. Clearly very many variations of ρ_α and ρ_e may be used depending on the relative importance attached to either agriculture or electrical energy production. For example, several different values are used and the implication of any particular choice is examined.

8.2.3. Model 3:

This model is essentially more complicated than the above two. The assumptions and constraints discussed in section 8.2.1. still hold - together with several other constraints to be discussed below.

The object here is to investigate the feasibility of increasing the complementarity between agricultural and power production by means of a down-stream storage. A portion of this storage could be used to regulate the flow released from the main reservoir. Thus in this example water released during the winter months to meet peak power demand can be stored (probably underground) and then pumped during spring and summer when needed for crops.

Fig. 8.2. is a schematic illustration of the new model. ξ_i represent flow routed from the main reservoir, through the generators, to the down-stream storage during period i; r_i is the volume released from the down-stream storage for irrigation during period i;x_i is the volume released from the main reservoir and routed (via the power plant) straight for irrigation.

(a). Further constraints:

Apart from the set of constraints outlined in section 8.2.1.c, the following restrictions are imposed.

(i) the total flow through the generating plant during period i is $x_i + \xi_i$;

(ii) the total volume of water for irrigation is $x_i + r_i$.

With all the above factors in mind, the task is to determine, for a given set of benefit and cost coefficients, irrigation and power output, and routing schemes that optimize the net benefits. From the corresponding values of ξ_i and r_i, the approximate capacity of the down-stream storage can be determined.

The mathematical programming formulation is as follows:

$$\text{Maximize} \quad \rho_\alpha A + \rho_e \hat{E} - \rho_w \Sigma r_i \qquad (8.2.10)$$

$$\text{subject to} \quad x_i + r_i \geq 0 \quad \alpha_i A \qquad (8.2.11)$$

$$x_i + \xi_i \geq 0 \quad \beta\hat{E} \quad i=1,..,12 \qquad (8.2.12)$$

$$\sum_{i=1}^{12} \xi_i - \sum_{i=1}^{12} r_i = 0 \qquad (8.2.13)$$

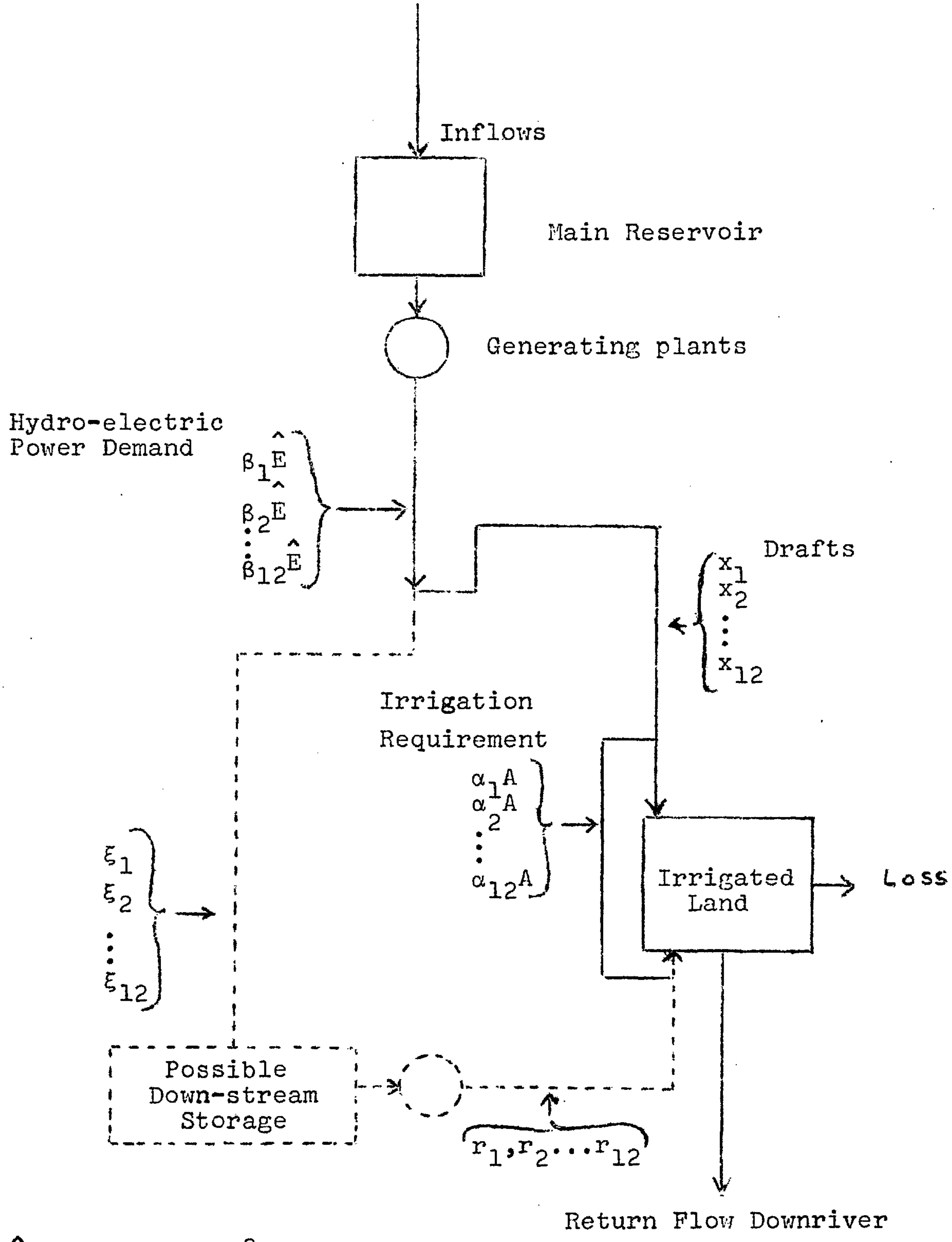
$$\sum_{i=1}^{12} x_i + \sum_{i=1}^{12} \xi_i = 80 \qquad (8.2.14)$$

$$\xi_i, r_i, x_i \geq 0 \quad i = 1,...,12 \qquad (8.2.15)$$

where $\hat{E} = k_1 E + k_2 E^2$. A range of possibilities is examined by varying the cost and benefit coefficients. The coefficient ρ_w is the unit cost of wells, pumps and any other item associated with supplemental storage.

8.3. Method of Solution

All the three models were solved by a method of feasible directions. A general version of the method is discussed in Chapter 5, section 5.3.2.b. A common feature of the three models is that all the constraints are linear. Although a quadratic cost function has been used for the present discussions, a more ncn-linear function



$$\hat{E} = k_1 E + k_2 E^2$$

Fig. 8.2.: Schematic Diagram for Model 3.

may be used without loss of generality.

The pertinent features of the method are summarized in the flow-chart in Appendix A2. This version handles only linear constraints, and is based on the survey paper by Dorn.[20] An initial feasible solution may either be provided, or may be computed by the programme.

Like many other non-linear programming techniques, this method will only compute a local minimum (maximum). Of course, if the function to be optimized is convex (concave), then the local optimum is also the global optimum.

All the inequality constraints are transformed into equality ones by the addition of appropriate 'slack' variables.

The programme has been written in Fortran IV code, and successfully run on the IBM 7090, 7094 and 360 digital computers.

8.4. Results:

Optimal operating policies of Model 1 are given in Table 3. Values of irrigation target, A, ranging from 50 to 80 m.c.m. have been used. The results are also sketched in Fig. 8.3.

The values of $\hat{E}$ in Table 3 all fall on the segmented line BC...LM in Fig. 8.3. The vertices of this line intersect lines from the origin with slopes α_i/β_i. Any point on the segmented line may be viewed as representing a feasible and optimal mix of electric power and agricultural production. For example, point J represents an optimal operating scheme in which $A = 54.4$ m.c.m. and $\hat{E} = 76.8$ m.c.m.

Any point falling below and to the left of the segmented line represents a feasible but non-optimal policy: whereas any point above and to the right of the line is outside the feasible region.

Point B represents a single-purpose agricultural production with hydro-electric power generation treated as supplementary; on the other hand point M indicates a single-purpose hydro-electric energy generation with agriculture treated as residual.

Optimal operating policies for Model 2: with various values of preference coefficients are given in Table 4.

In Fig. 8.3. point M' represents the operating policy that maximizes the function $F = 2 \times A + 10 \times \hat{E}$; point K for $F = 4 \times A + 10 \times \hat{E}$; point H' for $F = 5(A + \hat{E})$ and $F = 6.86 \times A + 5.85 \times \hat{E}$; and point J for $F = 6.8 \times A + 9.85 \times \hat{E}$.

The dashed lines Fig. 8.3. have negative slopes for the five values of ρ_α/ρ_e. Lines parallel (to the dashed lines) are tangent to the segmented line at points M', K, H' and J respectively. A desirable feature about Fig. 8.3. is that for any values of ρ_α and ρ_e, the point of tangency of a straight line (parallel to the negative slope of ρ_{α}/ρ_{e}) with the segmented line indicates the optimal values of A and $\hat{E}$ that maximizes the preference function $F = \rho_{\alpha}A + \rho_{e}\hat{E}$. This provides a convenient method of investigating the implication of implementing any particular economic objective. Thus if ρ_{α}/ρ_{e} is near zero, points near M will be indicated; whereas if ρ_{α}/ρ_{e} approaches a large positive value, points near B are indicated.

The figure (8.3) has been constructed along the lines indicated by H.A. Thomas et al.[75]

We see that for Model 2 with the preference function $6.8 \times A + 9.85 \times \hat{E}$, the total benefit in multipurpose operation is proportional to

$$6.8(54.388) + 9.85(76.872) = 1126.998$$

If, however, there had been full complementarity, the full potential of the water resource system for both electrical energy and agriculture would be realized; and the total benefit in multipurpose operation would be proportional to

$$6.8(80) + 9.85(80) = 1432$$

So that the actual degree of complementarity is 1126.988/1432 = 78.5% for this particular preference function.

Optimal schemes for Model 3 are given in Table 5. For small ρ_{w}, the objective function is maximized for both irrigation target, A and hydro-power generation equal to 80 m.c.m. - point Q on Fig. 8.3. With all the other

TABLE 3

Monthly Releases at Various Levels of Irrigation Requirements to Maximize Electric Energy Production.

Month	x(mcm)	A*=50	55	60	65	70	75	80
Jan	x_1	7.334	7.081	6.635	6.038	5.240	4.316	2.822
Feb	x_2	7.413	7.158	6.706	6.103	5.300	4.362	2.997
Mar	x_3	7.366	7.112	6.663	6.164	5.263	4.950	5.280
Apr.	x_4	7.018	6.776	6.849	6.565	7.070	7.575	8.080
May	x_5	7.450	8.195	8.940	9.865	10.430	11.175	11.920
June	x_6	6.900	7.590	8.280	8.970	9.600	10.350	11.040
July	x_7	5.842	6.270	6.840	7.410	7.980	8.550	9.120
Aug.	x_8	5.667	5.554	6.060	6.565	7.070	7.575	8.080
Sept.	x_9	5.255	5.170	5.640	6.110	6.580	7.050	7.520
Oct.	x_{10}	6.140	5.930	5.555	5.265	5.670	6.075	6.480
Nov.	x_{11}	6.614	6.387	5.984	5.446	4.726	3.687	3.680
Dec.	x_{12}	7.018	6.776	6.349	5.780	5.014	3.924	2.837
Σx_i		79.999	79.999	80.011	80.001	79.933	79.589	79.836
$K_1E + K_2E^2$		79.030	76.309	71.493	65.067	56.467	45.789	27.55

* Irrigation Requirement is in milliards of cubic metres per year.

TABLE 4

Optimal Cutputs for Irrigation and Hydro-power and Schedule of Flow Releases

		Economic Objective				
		ρ_α = 2	4	5	6.8	6.86
		ρ_e = 10	10	5	9.85	5.85
Optimal value of A =		45.393	50.893	61.547	54.383	61.547
Optimal value of $\hat{E}$ =		80.001	78.721	70.003	76.872	70.003
Benefit $F = \rho_\alpha A + \rho_e \hat{E}$		897.000	990.784	657.75	1126.998	831.731
Month	Flow Release					
Jan.	x_1	7.482	7.305	6.496	7.134	6.496
Feb.	x_2	7.562	7.384	6.566	7.211	6.566
Mar.	x_3	7.514	7.337	6.524	7.165	6.524
Apr.	x_4	7.159	6.990	6.216	6.826	6.216
May	x_5	6.764	7.583	9.171	8.103	9.171
June	x_6	6.264	7.023	8.494	7.505	8.494
July	x_7	5.942	5.802	7.016	6.200	7.016
Aug.	x_8	5.781	5.644	6.216	5.512	6.216
Sept.	x_9	5.361	5.235	5.785	5.112	5.785
Oct.	x_{10}	6.264	6.112	5.439	5.973	5.439
Nov.	x_{11}	6.748	6.590	5.859	6.434	5.859
Dec.	x_{12}	7.159	6.990	6.216	6.826	6.216
Annual Σx_i		80.0	79.995	79.998	80.001	79.996

TABLE 5

Optimal Operating Schemes for Regulation of Flow

	ρ_w											
	0;	2;	8	9;	9.5		9.7;	10;		10.5		
Month	x_i	i	r_i	x_i	i	r_i	x_i	i	r_i	x_i	i	r_i
Jan.	2.580	4.970		7.040	.298		7.550			6.496		
Feb.	3.060	4.660		2.520	5.020		7.620			6.566		
March	5.600	2.00		5.010	2.380		5.350	2.280		6.524		
April	7.200		1.170	7.040		.660	7.150			6.216		
May.	6.750		5.550	6.410		4.760	6.560		3.820	9.171		
June	6.450		5.120	6.270		4.360	6.140		3.530	8.494		
July	6.070		3.480	5.880		2.860	5.920		2.150	7.016		
Aug.	5.900		2.500	5.810		2.010	5.820		1.345	6.216		
Sept.	6.040		1.830	5.910		1.365	5.350		.712	5.785		
Oct.	6.460		.420	6.290			5.760	.555		5.439		
Nov.	4.100	2.700		3.560	3.120		3.360	3.950		5.859		
Dec.	2.240	5.040		2.160	5.200		1.860	5.300		6.216		
Annual	60.18	19.800	19.840	63.930	16.018	16.015	68.420	11.555	11.557	79.996	0	0
$\hat{E}$	80.			80			80			70.003		
A	80.			76.8			70.05			61.547		
F	1016.8		958.8	851		843	842.45		838.45	831.731		

$\rho_\alpha = 6.86$ $\rho_e = 5.85$

$$F = \rho_\alpha A + \rho_e \hat{E} - \rho_w \Sigma r_i$$

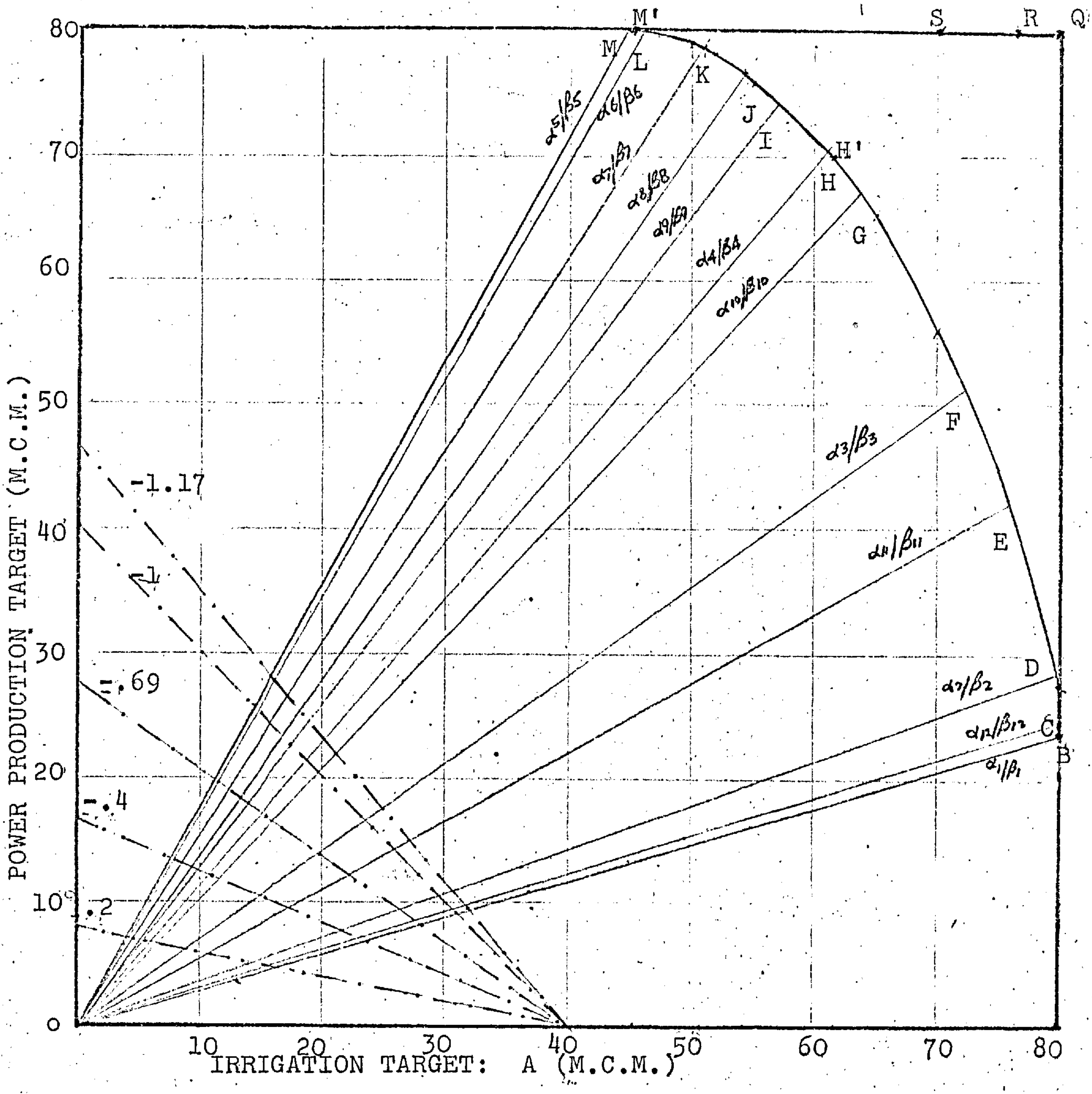

POWER PRODUCTION TARGET (M.C.M.)
IRRIGATION TARGET: A (M.C.M.)
0
10
20
30
40
50
60
70
80
M'
S
R
Q
M
L
K
J
I
H'
H
G
F
E
D
C
B
α_5/β_5
α_6/β_6
α_7/β_7
α_8/β_8
α_9/β_9
α_4/β_4
α_{10}/β_{10}
α_3/β_3
α_{11}/β_{11}
α_2/β_2
α_{12}/β_{12}
α_1/β_1
-1.17
-1
-.69
-.4
-.2

Fig. 8.3

coefficients (ρ_α and ρ_e) kept constant increase in the value of ρ_w up 8 has no effect on the values of x_i, ξ_i, and r_i. The value of the objective function is, of course, reduced (see equation 8.2.10).

For values of $\rho_w = 9.0$ and 9.5 respectively the optimal operating point is R; while for values of 9.7 and 10 respectively, the operating point is S. For values of $\rho_w = 10.5$ and above ξ_i, $r_i = 0$, $i = 1,\ldots 12$ and the operating point falls on the segmented line BC...LM.

From Table 5 we note that the months in which r_i is positive are consecutive. So that Σr_i represents both the volume of water to be released from storage and the total volume required for regulation. As ρ_w is increased the volume of water required for storage decreases from 19.184×10^3 cubic metres (at point Q) to 16.015×10^3 cubic metres (point R) then to 11.557×10^3 cubic metres and finally to zero.

The above simple computational examples indicate how mathematical programming, and especially the method of feasible directions, may be usefully employed as an aid in determining the best operating policy of a reservoir for electrical energy and agricultural production. Although the models considered are simple, this approach can provide quite useful first approximations to the actual operating conditions.

The assumption that stream flows can be predicted with certainty is a very strong one; and is in fact never realized in practice. The inclusion of the desirable degree of uncertainty will, of necessity, complicate the objective function and/or the constraints, especially for large problems. In the next section a formulation that takes into account the said uncertainty is suggested. The model represented is suitable for solution by the method of dynamic programming discussed in Chapter 6.

Further refinements towards a more realistic model should also include probability relationships for storage of water in the reservoir(s) and for drafts; introduction of head on the turbines as a variable; introduction of flood protection and control as a purpose with economic benefits. These will naturally result in a non-linear objective function with non-linear constraints. Many new variables will be introduced and the number of constraints increased. The problem may then be solved by any of the non-linear programming problems discussed in Chapter 5 or by a version of the method of feasible directions that incorporates non-linear constraints.

8.5. A Possible Solution by Dynamic Programming

The objective here is to indicate - qualitatively - how a model similar to Model 3 may be formulated and solved by the method of dynamic programming.

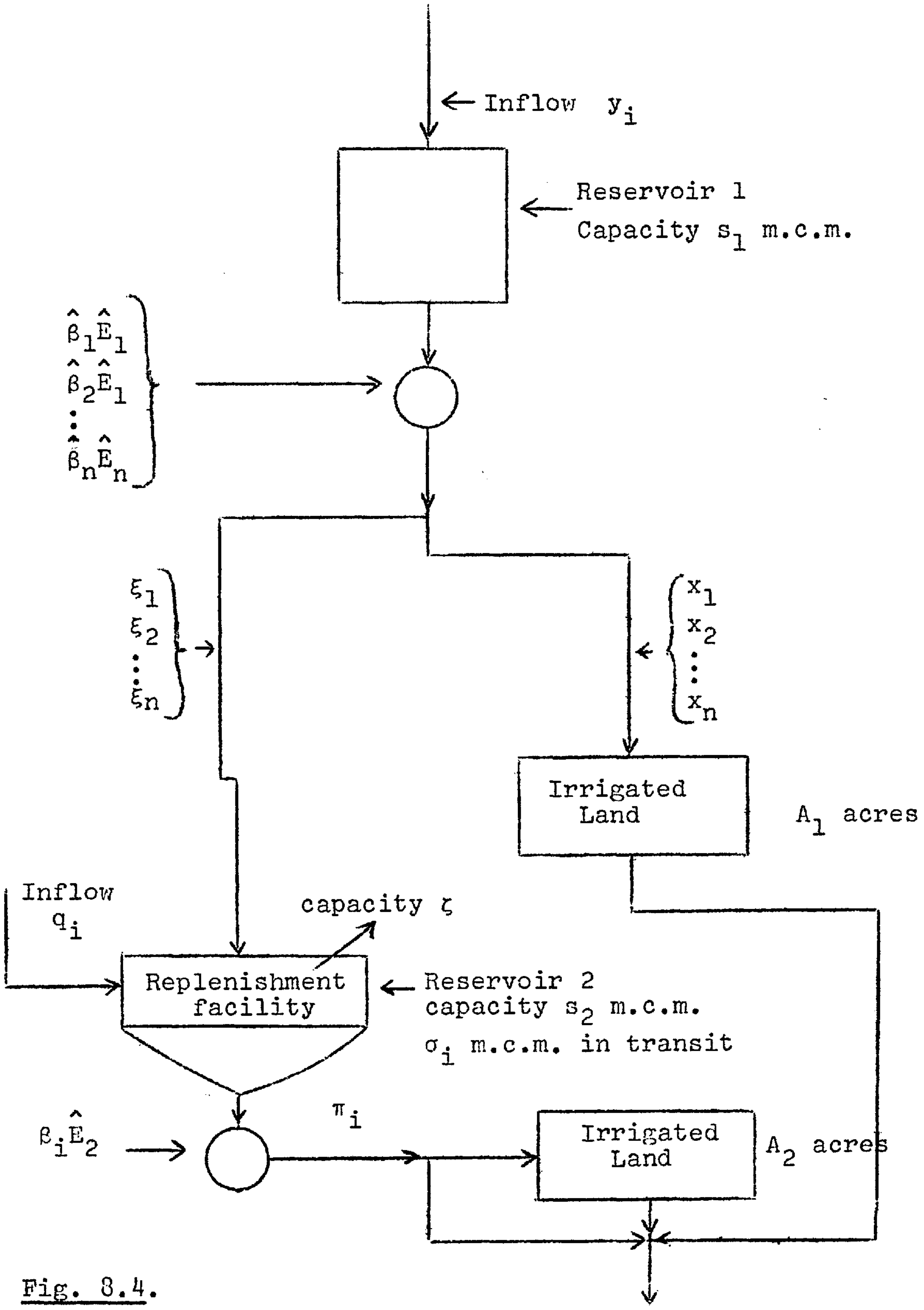
A schematic diagram of the model is shown in Fig. 8. 4 . It is slightly more complicated than Model 3: the time lag between the time when water is released from reservoir 1 and the time when the water is available for utilization in the lower reservoir is taken into account. Furthermore, there is a second stream flow (q_i) into reservoir 2. In addition, the quantities of water in both the reservoirs enter into the computations directly.

8.5.1. Problem Formulation:

The two reservoirs are connected in series, with y_i m.c.m. entering reservoir 1 at time interval i. The capacity of reservoir 1 is S^1 and $s_i^{\ 1}$ (m.c.m) ($0 \leq s_i^{\ 1} \leq S^1$) is stored at the beginning of period i.

The capacity of reservoir 2 is S^2 and the corresponding volume of water stored is $s_i^{\ 2}$ m.c.m. In addition reservoir 2 is replenished by ζ m.c.m./season, part of which is released from reservoir 1 (ξ_i) and the other part (q_i) by natural flow. The time lag effect is allowed for by assuming that σ_i m.c.m. of water will be available for use at the $(i+1)^{th}$ sub-interval.

As with the previous models (1,2 and 3) the purpose of this particular one is twofold: to generate electricity, and to supply water for irrigation. The net benefit from



Inflow y_i
Reservoir 1
Capacity s_1 m.c.m.
$\hat{\beta}_1\hat{E}_1$
$\hat{\beta}_2\hat{E}_1$
$\vdots$
$\hat{\beta}_n\hat{E}_n$
ξ_1
ξ_2
$\vdots$
ξ_n
x_1
x_2
$\vdots$
x_n
Irrigated Land
A_1 acres
Inflow q_i
capacity ζ
Replenishment facility
Reservoir 2
capacity s_2 m.c.m.
σ_i m.c.m. in transit
$\beta_i\hat{E}_2$
π_i
Irrigated Land
A_2 acres

Fig. 8.4.

both purposes is to be maximized.

The vector describing the state of the system has the following components:

(i) amount of water stored in reservoir S^1

(ii) amount of water stored in reservoir S^2

(iii) amount of water in transit

Our task is then to establish an optimal policy of the system during a long period of time, starting in any state of the reservoirs and the replenishment capacity. This policy involves three decisions to be taken into account at the beginning of the interval: x_i, ξ_i and π_i.

8.5.2. Dynamic Programming Formulation

The method of dynamic programming has been discussed in Chapter 6. We wish to show how the method may be used to formulate the above problem.

In this problem the three decisions x_i, ξ_i, and π_i constitute a policy vector $\delta=(x,\xi,\pi)$. The policy vector is constrained to be within a set of all admissible policies:

$$(x,\xi,\pi) \in P\delta$$

Since time lag is allowed for in the transit of water from the up- to the down- stream reservoir, the returns derived for process of duration N = 1 is

$$\Phi(x,\xi,\pi) = \Phi_A(x,\pi) + \Phi_E(x,\xi,\pi) \qquad (8.5.1)$$

where $\Phi_A(x,\pi)$ is the return from irrigation and $\Phi_E(x,\xi,\pi) = \Phi_{E1}(x,\xi) + \Phi_{E2}(\pi)$ is the return from hydro-power utilization.

Hence the expected optimal return from the first stage process (N=1) is

$$F_1(x,\xi,\pi) = \underset{(x,\xi,\pi)\in P_\delta}{\mathrm{Max}} \; \Phi_A(x,\pi) + \Phi_E 1(x,\xi) + \Phi_E 2(\pi) \qquad (8.5.2)$$

Following the operation of the system for one stage, the system is transformed from state

$$S = (s^1, s^2, \sigma) \qquad (8.5.3)$$

to state $\underline{S} = (\underline{s}^1, \underline{s}^2, \underline{\sigma})$. (8.5.4)

This transition is due to transformation

$$T = T(\bar{x}, \bar{\xi}, \bar{\pi}, \bar{y}, \bar{q}) \qquad (8.5.5)$$

Observe that the transformation vector T is composed, in part, of certain elements which are at our discretion: x, ξ, π; and in part of certain elements which are stochastic: $\bar{y}$ and $\bar{q}$.

For this example it is assumed that the variability in the flow of $\bar{y}$ during the period of study is quite marked; and that the variability of the flow of $\bar{q}$ is so small that the flow can be reasonably represented by an average value.

Consequently the new state $\underline{S}$ is denoted by:

$$\underline{S} = \int_{-\infty}^{\infty} (s^1+y-x-\xi,\; s^2+\sigma-\pi,\; \xi+q)h(y)dy \qquad (8.5.6)$$

where h(y) is the probability distribution function of the inflow into reservoir 1.

Let us assume that the operating stages correspond to monthly periods. Hence the return from the second stage is denoted by

$$F_2(s^1,s^2,\sigma) = \underset{(x,\xi,\pi)\in P\delta}{\text{Max}} \{\Phi(x,\xi,\pi) + \int_{-\infty}^{\infty} F_1(s^1+y-x-\xi,s^2+\sigma-\pi,\xi+q)h(y)d \qquad (8.5.7)$$

So that, in general, following the Principle of Optimality (Chapter 6), the following equation is derived:

$$F_N(s^1,s^2,\sigma) = \underset{(x,\xi,\pi)\in P\delta}{\text{Max}} \{\Phi(x,\xi,\pi) + \int_{-\infty}^{\infty} F_{N-1}(s^1+y-x-\xi,s^2+\sigma-\pi, \xi+q)h(y)dy\} \qquad (8.5.8)$$

We observe that the state vector of the functional equation is composed of three elements, thus giving rise to a three-dimensional problem.

Existing digital computing facilities can now handle three-dimensionsl dynamic programming problems of the type formulated above. And it is hoped that in future some research effort will be directed at programmes that are capable of achieving such solutions.

In order to facilitate a programme of the type suggested, the following steps have to be taken:

(i) the integral is replaced by a summation; i.e. the continuous distribution function is discretized, thus giving rise to

$$F_N(x,\xi,\sigma) = \underset{(x,\xi,\pi)\in P\delta}{\text{Max}} \{\Phi(x,\xi,\pi) + \sum_{j=1}^{M} \lambda_j \, F_{N-1}(s^1+y_j-x-\xi, s^2+\sigma-\pi,\xi + q)\} \qquad (8.5.9)$$

$$\text{with } \lambda_j \geq 0$$
$$\Sigma\lambda_j = 1$$

(ii) The set, Pδ, of admissible solutions is specified. For this example, constraints of the type

$$0 \leq x + \xi \leq s^1$$
$$0 \leq \pi \leq s^2$$
$$0 \leq \xi + q \leq \zeta$$
$$0 \leq s^1 \leq \text{specified number}$$
$$0 \leq s^2 \leq \text{specified number}$$
$$0 \leq \sigma \leq \text{specified number}$$

A programme that solves a problem of the type discussed above would yield a long-run optimal operating procedure.

The benefit function could take any general form, either linear or non-linear.

CHAPTER 9

OPTIMAL OPERATION OF A HYDRO-THERMAL ELECTRIC SYSTEM

In this chapter we consider the solution of a scheduling problem involving a system that has both hydro-electric and thermal units. First the general case is formulated; and then a sample model with two hydro-electric and one thermal unit is solved to minimize an annual cost index. The solution is obtained by means of a two sequential unconstrained optimization techniques.

9.1. The General Problem

The object here is to formulate a general problem consisting of any (finite) number of fixed head hydro-electric stations and any number (finite) of thermal stations.

(a) Variables of the Problem

For a specified set of natural river flow, the power generated at a hydro-electric plant at any instant of time is a function of the quantity of water stored in the reservoir and the rates of change of storage at the plant under consideration and all the up-stream plants.

Moreover, for any given system load, the energy which has to be supplied by the thermal plants, and the cost thereof, are functions of the reservoir storages. and rates of change of the storages. Consequently, the storage draw-down curves of the various reservoirs in the system may be treated as variables of the problem.

(b) Assumptions

In the course of the formulation of the general model, several assumptions are made. Some of these are listed below.

(i) the cost of water for generating electricity is negligible;

(ii) there is no depletion of water supply through evaporation;

(iii) water flowing into the reservoir during the i^{th} sub-interval is not available for electricity generation until the next sub-interval;

(iv) the function, $F(S_{ik})$ of the thermal units is strictly convex and is differentiable;

(v) thermal generation is required throughout the study period.

(c) Constraints

Broadly speaking, these fall into: physical and operating restrictions. The physical constraints will vary according to the problem in question. Some of the common ones are:

(i) maximum plant (turbine) discharge for any given sub-interval;

(ii) minimum plant discharge for a given sub-interval;

(iii) maximum and minimum storage capacity for each reservoir at any sub-interval,

(iv) maximum and minimum output of thermal plants during any sub-interval of time.

The main operating restriction is that of supplying the electric energy demand (allowing for losses) at any given interval of time. For some systems, in order to ensure continuous operation for the hydro-electric plants, it may be necessary to specify bounds on the quantity of water that may be used for electricity generation over a period of time (i.e. a day, week, month or year).

(d) Statement of the Problem:

With the above points in mind, the problem of determining the optimum system schedule may be stated as follows:

Determine the average thermal output per sub-interval: S_{tji} $t = 1,\ldots,T$; $j = 1,\ldots,M$; $i = 1,\ldots,N$; and the average plant discharge, $Q_{t,\ell}$, $t = 1,\ldots,T$; $\ell = 1,\ldots,L$ for which the total thermal cost over a specified period:

$$\sum_{t}^{T} \sum_{i}^{N} \sum_{j}^{M} F_{tij}(S_{tji}, Q_{t,\ell}, R_{t\ell}, R_{t\ell}) \qquad (9.1.1)$$

is a minimum, subject to the constraints:

$$\sum_{i=1}^{N} S_{tj} + \sum_{\ell=1}^{L} H_{t\ell}(Q_{t,\ell}) \geq D_t + P_{Lt} \qquad (9.1.2)$$

$$\sum_{t=1}^{T} Q_{t\ell} = V_\ell \qquad \ell = 1,\ldots,L \qquad (9.1.3)$$

$$Q_t \min \leq Q_{t\ell} \leq Q_t \max \qquad \ell = 1,\ldots,L \qquad (9.1.4)$$

$$R_t \min \leq R_{t\ell} \leq R_t \max \qquad (9.1.5)$$

$$S_{ji} \min \leq S_{tji} \leq S_{ji} \max \qquad (9.1.6)$$

$$j = 1,\ldots,M$$

$$i = 1,\ldots,N$$

$$t = 1,\ldots,T$$

where $Q_{t\ell}$ is the average discharge through a hydro-electric station ℓ during the sub-interval t; $t = 1,\ldots T$. $H_{t\ell}$ is the hydro-generation corresponding to $Q_{t\ell}$. V_ℓ is daily or weekly allotment of water at hydro-station ℓ. $R_{t\ell}$ is the storage capacity of reservoir ℓ during sub-interval t. D_t is the average system demand during sub-interval t and P_{Lt} the average system loss during the sub-interval t, and is a specified function of $H_{t\ell}$ and S_{tji}.

For generality, other terms are usually added to the cost function (in the form of penalty functions). The penalties are for violation of operating constraints; e.g. a penalty for not meeting the required demand or for producing much more electricity than is required for any given sub-interval of time.

Various forms of penalty functions may be used; e.g. for failing to meet the demand, the following penalty may be imposed:

$$C^{f}_{pt} = \begin{cases} 0 & \text{if all the demand is met.} \\ k_f|x_t - x_{at}|^{n} & \text{when energy supplied is less than that demanded} \end{cases} \qquad (9.1.7)$$

where $(x_t - x_{at})$ is the degree of violation, at any sub-interval t; and k_f and n are constants chosen to produce high penalty cost when a violation occurs.

Similarly excessive over-supply may be penalized:

$$C^{s}_{pt} = \begin{cases} 0 & \text{if the excess supply is within a specified acceptable limit.} \\ k_s\ |x_{s,t} - x_t|^{m} & \text{when the specified limit is exceeded.} \end{cases} \qquad (9.1.8)$$

The final cost function to be minimized is then:

$$F = \sum_{t}^{T} \sum_{i}^{N} \sum_{j}^{M} F_{tij} + \sum_{t=1}^{T} c^{f}_{pt} + \sum_{t=1}^{T} c^{s}_{pt} \qquad (9.1.9)$$

subject to constraints 9.1.2 through to 9.1.6.

9.2. The Particular Model Investigated:

The system investigated is illustrated in Fig. 9.1. It consists of two hydro-electric plants and one thermal plant. The plants are arranged in a parallel series stream inter-connection. Consequently, discharge at the upstream plant affects the downstream plant.

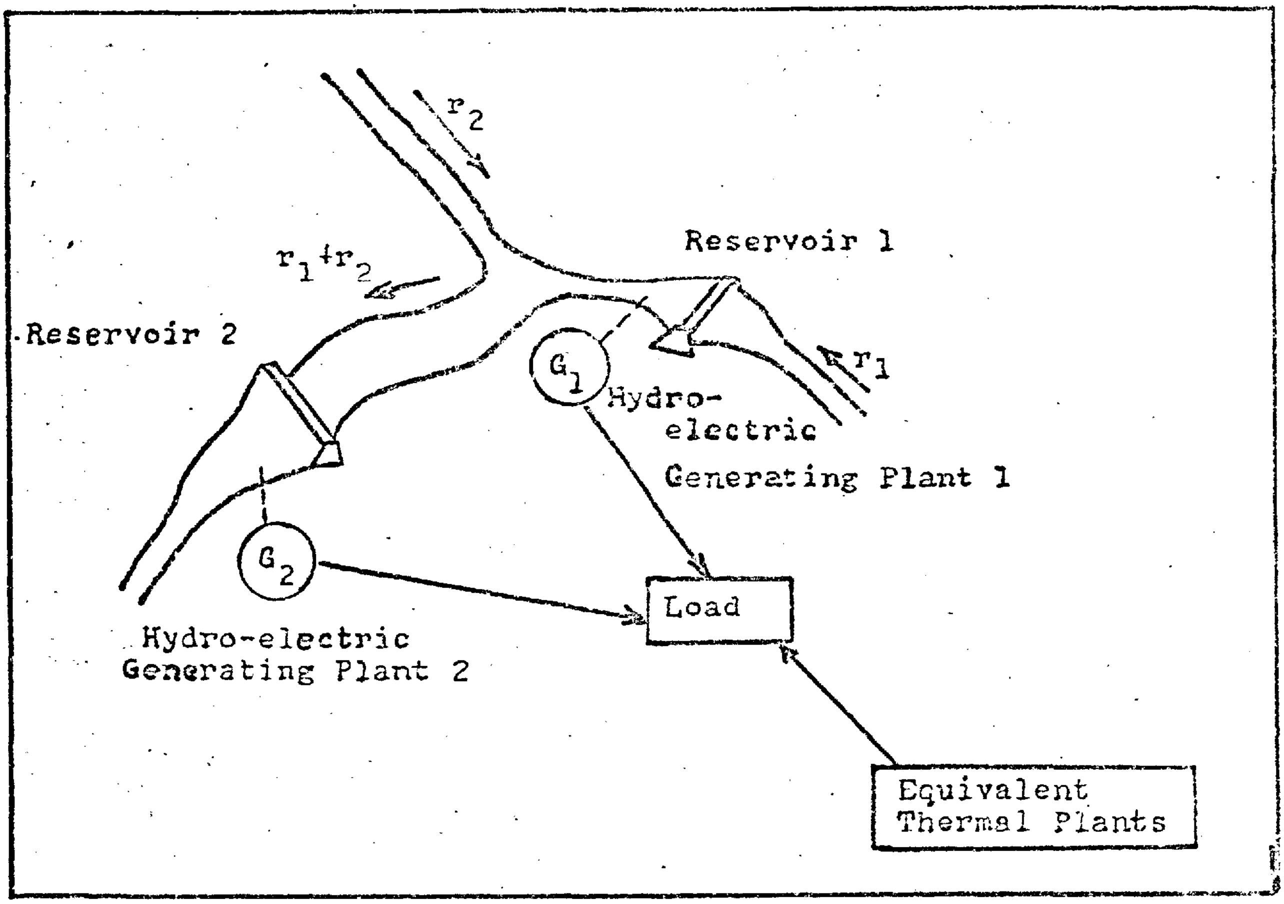

r_2
Reservoir 1
r_1+r_2
Reservoir 2
G_1
Hydro-electric Generating Plant 1
r_1
G_2
Load
Hydro-electric Generating Plant 2
Equivalent Thermal Plants

Fig. 9.1. Schematic Diagram of the Sample Problem.

In Fig. 9.2. the variations in the value of the natural stream flow for reservoirs 1 (upstream) and 2 are plotted. A graph of variations in peak demand is also included in the figure.

(a) Assumptions

For the system considered, we shall assume the following:

(i) no charge is made on the cost of water for generating electricity.

(ii) penalty cost for violating the operating constraints is not included in the cost function. The thermal units have sufficient capacity to meet the defecit that cannot be met by the hydro-electric units.

(iii) all the thermal units are represented by an equivalent unit S_t, $t = 1,\ldots,T$.

(iv) the cost function is quadratic: of the form $F(S,H) = .5 + 2 \times S_t + 0.006 \times S_t^2$ /sub-interval.

(v) No transmission losses are included, these being assumed negligible.

(vi) the thermal units are in service when required. The start-up and shut-down costs are reflected in the operating cost expression.

(b) Plant Characteristics

The following functional relationships and constraints were used for the model in question. They are essentially,

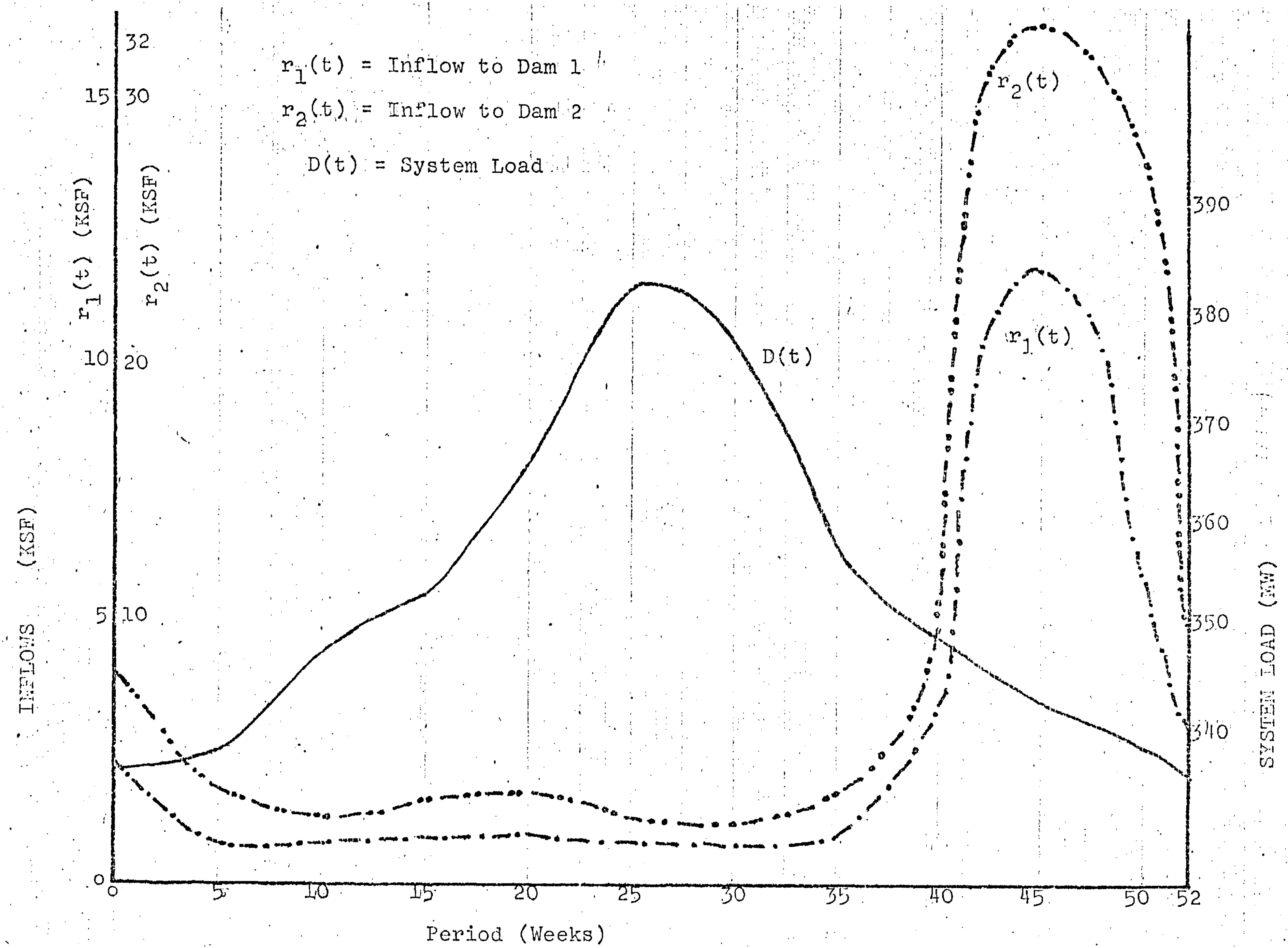


Fig. 9.2 Stream Flows and Load Demand

similar to those used by Cypser[13]; and were obtained by fitting curves to portions of tabulated data.

(i) Forebay elevations:

$$Y_{1,t} = -7.59077 \times 10^{-5} \times S^2_{1,t} + .25965 S_{1,t} + 3343.24584 \qquad (9.2.1.)$$

$$Y_{2,t} = .01666 \times S_{2,t} + 2883 \qquad (9.2.2)$$

(ii) Total Plant Discharge:

$$Q_{1_t} = r_{1,t} - \dot{S}_{1,t} \qquad (9.2.3)$$

$$Q_{2_t} = r_{2,t} - \dot{S}_{1,t} - \dot{S}_{2,t} \qquad (9.2.4)$$

(iii) <u>Tail-water elevation</u>

$$Y_{1T,t} = 3072. + 1.5012 * Q_{1,t} - 0.03404 \times Q^2_{1,t} \qquad (9.2.5)$$

$$Y_{2T,t} = 2697. + .5687 * Q_{2,t} - .00359 \times Q^2_{2,t} \qquad (9.2.6)$$

(iv) <u>Effective head</u>:

$$h_{1,t} = Y_{1,t} - Y_{1T,t} \qquad (9.2.7)$$

$$h_{2,t} = Y_{2,t} - Y_{2T,t} \qquad (9.2.8)$$

(v) <u>Power Generated:</u>

$$H_{1,t}(Q_{1,t}) = .06486 \times Q_{1,t}(h_{1,t} - 20 + 38.107 \times Q_{1,t} - 2.863 \times Q^2_{1,t}) \qquad (9.2.9)$$

$$H_{2,t}(Q_{2,t}) = 0.076 \times Q_{2,t}(h_{2,t} - 5 + 2.885 \times Q_{2,t} - 0.018 \times Q^2_{2,t}) \qquad (9.2.10)$$

Constraints:

The physical and operating constraints imposed included:

(i) minimum plant discharge:

$$Q_{1,t} \text{ min} = 0 \qquad (9.2.11)$$

$$Q_{2,t} \text{ min} = 0 \qquad (9.2.12)$$

(ii) minimum storage:

$$R_{1,t} \text{ min} = 0 \qquad (9.2.13)$$

$$R_{2,t} \text{ min} = 0 \qquad (9.2.14)$$

(iii) maximum plant discharge:

(A) As a function of effective head:

$$Q_{1,t} \text{ max} = 0.034327 \times h_{1,t} - 3.32696 \quad \text{if } h_{1,t} < 387.9883 \qquad (9.2.15)$$

$$Q_{1,t} \text{ max} = 18.07152 - .020825 \times h_{1,t} \text{ if } h_{1,t} \geq 387.9883 \qquad (9.2.16)$$

$$Q_{2,t} \text{ max} = 0.036636 \times h_{2,t} + 3.28796 \text{ if } h_{2,t} < 188.14659 \qquad (9.2.17)$$

$$Q_{2,t} \text{ max} = 19.79267 - 0.05108 \times h_{2,t} \text{ if } h_{2,t} \geq 188.14659 \qquad (9.2.18)$$

(B) As a function of reservoir storage

$$Q_{1,t} \text{ max} = 22.0 + .005 \times R_{1,t} \text{ if } R_{1,t} < 750 \qquad (9.2.19)$$

$$Q_{1,t} \text{ max} = 28.0 - .003 \times R_{1,t} \text{ if } 750 \leq R_{1,t} \leq 1360.5 \qquad (9.2.20)$$

$$Q_{1,t} \text{ max} = 23.9185 + .375(R_{1,t} - 1360.5) \text{ if } R_{1,t} > 1360.5 \qquad (9.2.21)$$

Constraint (iii)(A) was used for the present problem

(iv) Operating

$$H_{1,t} + H_{2,t} + S_t \geq D_t \qquad t = 1,\ldots,T$$

The period considered was one year, divided into 10 sub-intervals of 5 weeks each and the ~~relevant~~ remaining sub-interval of 2 weeks. All these led to 99 constraints in 33 unknowns. The operating cost was based on hourly production cost which was then multiplied by the appropriate factor to obtain the total cost for the entire year.

9.3. Method of Solution

Two related methods of non-linear programming have been used to solve the above problems. These are the sequential unconstrained minimization technique developed by Fiacco and McCormick and the logarithmic potential method developed by Lootsma. As we have indicated in Chapter 5 both methods employ penalty function as a means of staying within the feasible region.

Of the two, the version by Fiacco and McCormick has been the more extensively used, Lootsma's version being relatively less known. In the power-systems domain Mitter and Liacco have used Fiacco and McCormick's method to obtain solutions for reliable operation of a power system under different operating conditions.[59] Sasson has used both methods to solve load-flow problems.[70]

A common feature of both the methods is the transformation of a constrained into an unconstrained problem. In the above example we have

$$F(\bar{x}) = \sum_{i=1}^{11} (.5 + 2.x_i + 0.006.x_i^2) \qquad (9.3.1)$$

subject to

$$G_i(x_j) \geq 0 \quad i = 1, 99 \qquad (9.3.2)$$
$$j = 1, 33$$

This is transformed into an unconstrained function

$$P(r,\bar{x}) = F(\bar{x}) + r \sum_{i=1}^{99} 1/G_i(\bar{x}) \qquad (9.3.3)$$

or

$$P(r,x) = F(\bar{x}) - r \sum_{i=1}^{99} \text{In. } G_i(\bar{x}) \qquad (9.3.4)$$

Equation (9.3.3) is due to Fiacco and McCormick and Equation 9.3.4. is Lootsma's Logarithmic potential method. Both methods have been discussed in Chapter 5. Appendix A3 illustrates in very general terms, the pertinent steps of the algorithm of each of the methods.

In each case, the sequence of unconstrained minimization of $P(r_k,\bar{x})$ is carried out by Fletcher and Powell's modified version of Davidon's variable metric algorithm (Chapter 5). A flow chart of Fletcher and Powell's method is given in Appendix A4.

9.4. Results

The optimal operating policy for one year is given in Table 9.2. Table 9.1. gives the initial (feasible) solution. The variations in load deficit correspond to the variations in stream flows and system load shown in Fig. 9.1. After about the 44th week, the hydro-units together produce power in excess of the system demand. For an integrated system this would mean exporting the excess power generated. For an isolated system, on the other hand, the turbine discharge would have to be limited to supplying only the system demand; so that some volume of water has to be spilled over.

It must be emphasized that the total costs for the period will depend upon the specified state of the reservoirs at the beginning and end of the planning period. For each set of specified conditions, there will be a different policy and the above results refer to only one such conditions.

A more realistic formulation should have allowed for the uncertainty related to the stream flow and load estimates. The manner of handling such forms of uncertainty will, of course, depend on the statistic on record and the method of evaluating these. This is a field in which further research work could be pursued.

The same initial solution was used for both methods (Logarithmic potential method and the method due to Fiacco and McCormick); and both methods gave practically identical optimal solutions. A noticeable advantage of

TABLE 9.1.

INPUT DATA AND INITIAL SOLUTION

Period	End of Period Storage		Natural Stream Flows		Generated Power		Load	Deficit
(Weeks)	Kilo-sec-ft-days)		r1	r2	G_{1k}	G_{2k}	(MW)	(MW)
	Res.1.	Res.2.			(MW)	(MW)		
0	1380.	600.	4.0	10.	160.972	158.381	335.	15.647
5	1350.	550.	.8	3.5	59.241	33.966	337.	243.793
10	1345.	550.	.8	2.5	37.215	26.856	348.	283.429
15	1350.	525.	.9	3.5	55.843	27.433	352.	268.724
20	1340.	550.	.9	3.5	49.442	31.643	367.	285.915
25	1335.	545.	.8	2.5	37.837	26.826	382.	317.337
30	1350.	545.	.8	2.5	37.199	21.314	372.	313.487
35	1350.	550.	.9	3.5	52.042	28.841	358.	277.117
40	1350.	545.	3.5	3.	126.987	126.421	350.	96.592
45	1345.	550.	11.8	34.0	650.905	383.978	342.	0
50	1350.	545.	6.2	28.0	519.376	232.298	338.	0
52	1335.	545.	4.0	10.	160.972	158.831	335.	15.647

Cost = £12,785.704 / hour-week.

TABLE 9.2.

Optimal Solution

Period	End-Period Storage		Gen. Power		Load	Deficit
	(Kilo-sec ft.dys)		G_{1k}	G_{2k}		
(Weeks)	Res.1	Res.2	(MW)	(MW)	(MW)	(MW)
0	1380.	600.	125.408	80.969	335	128.623
5	1350.	550.	59.241	33.966	337	243.793
10	1200.	500.	43.440	68.698	348	235.862
15	1130.	430.	61.255	47.700	352	243.045
20	1020.	380.	58.398	58.240	367	250.362
25	900.	300.	46.494	56.059	382	279.447
30	800.	230.	44.954	49.009	372	278.037
35	700.	150.	61.085	50.676	358	246.239
40	630.	90.	129.941	128.980	350	91.079
45	1100.	300.	602.243	285.198	342	0
50	1300.	500.	480.993	167.102	338	0
52	1380.	600.	125.408	80.969	335	128.623

Cost £11,191.148/ hour-week.

the Logarithmic potential algorithm over the Fiacco-McCormick one is the former's better convergence to the solution - at least for the example considered.

Another advantage of the Logarithmic potential method is that the error function

$$e(r_j,x) = r_j \sum_{i=1}^{m} \text{In } G_i(x) \qquad (9.4.1)$$

can be approximated by a constant

$$\delta_e = m.r_j \qquad (9.4.2)$$

where m is the number of constraints. So that by specifying the value of δ_e, a value of r_j can be obtained such that the optimal value of $F(\bar{x})$ is found within a prescribed accuracy in one minimization process. For subsequent minimisation procedures, δ_e is reduced by a specified factor - (≡10 for the above example). This is an advantage which the Fiacco-McCormick method fails to offer; i.e. in their method, there is no way of approximating the error function

$$e'(r_j,x) = r_j \sum_{i=1}^{m} 1/G_i(\bar{x}) \qquad (9.4.3)$$

In the original Fiacco-McCormick algorithm (Appendix A3) a method of obtaining an initial value of r (i.e. r = r_1) is suggested. However the author has experienced some computational difficulty in applying the same procedure. First negative values of r are obtained; and by the time a corrective procedure has been invoked to give a positive value of r it is invariably found that one or more constraints are being violated. Consequently, an arbitrary initial value of r ($\equiv 0.11 \times F(\bar{x})$) has been chosen.

Furthermore, an acceleration by extrapolation suggested by Fiacco and McCormick[52] is found to lead to repeated violations of a number of constraints and the time required to drive the latter back to the feasible region counterbalanced the net increase in the rate of convergence. As a result, the acceleration subroutine has been discarded. Similar computational difficulties have also been experience by Sasson.[70]

A major weakness of the Fiacco-McCormick method and the logarithmic potential method is that the initial solution must be feasible. For a large problem with many constraints, a lot of time has, therefore, to be spent in obtaining such a solution.

Moreover, as we have indicated in Chapter 5, both the above methods employ 'boundary repulsion factor'; the Fletcher-Powell unconstrained minimization algorithm used in both cases, however, proceeds by taking successive steps (specified) along the generated conjugate directions (Chapter 5). Quite often, if the current point is close to the feasible boundary, the direction generated may be such that the minimization step will give a point outside the constrained region. A corrective procedure incorporated in the programme for handling such a situation, is to reduce the step length by a prescribed factor, until a point which lies in the constrained region has been obtained. For a large problem with 'tight' constraints, much time may, therefore, have to be spent in merely trying to keep within the constraint region, rather than carrying out the actual minimization process.

CHAPTER 10

ELECTRIC TRANSMISSION SYSTEM DESIGN

10.0 This chapter is concerned with investigations into the feasability of applying several mathematical techniques to the design of a transmission network. The methods considered include the Simplex algorithm and the modifications to handle integer linear programming problems; and two sequential unconstrained optimization techniques (already applied in Chapter 9).

In the following sections, assumptions are made about cost factors, demand of electrical energy, limitations of the power transfer capacities of the transmission lines and the reliability of operation of the lines. From these assumptions and the inherent restrictions (to be discussed) salient features of the most economical design are obtained.

Notable contributions in the application of mathematical programming method to electrical design are due to U.G. Knight[47,48,49]. Recently Burstall has proposed a heuristic method of solving linear programming problem of a network model of the type considered by Knight.[9] The approach followed in this chapter differs in concept to either of the two mentioned.

10.1 Problem Statement: General:

In general, the design problem is that of finding the total costs of a power system network subject to given constraints. The total costs include capital as well as operating costs; while the constraints may refer to the physical capability of the equipment in use and also to the design practice. For the present discussions, the above factors may be summarized as follows:

Find the minimum of:

(a) (Transmission line cost) + (b) (Switchgear cost) + (c) (transformer cost) + (d) (cost of reactive generation and control equipment) + (f) (Operating and maintenance costs of (a), (b), (c) and (d) + (g) cost of transmission active and reactive power loss;

Subject to:

(i) given reliability factor at substation being met;

(ii) power (m.w.) demand at each substation being met;

(iii) reactive power (m.v.a.r.) at each substation being met;

(iv) thermal limit of each equipment not to be exceeded;

(vi) power transfer capacity of each line should not exceed the stability loading limit of the line in question;

(vii) total power generated and/or imported must be sufficient to supply the total demand at given time.

The above list is by no means exhaustive: there are many more constraints that the designer could impose. Some of the above constraints are interrelated; some are severe and have decisive influence on the mode of operation and economic choice of the system to be designed, while others are mild and have negligible influence. The designer is particularly concerned with the severe and decisive constraints. The problem then becomes that of determining the degree of importance of the constraints. There are no systematic rules for achieving this; and here is where the designer's experience, skill and judgment plays an important role.*

have been treated at

Simple sy

main generating station and five load centers. The load demand at each center is indicated. Table 10.1 lists the distances between the stations.

The following assumptions are made:

* In addition, sensitivity analysis may provide a useful way of establishing the relative importance of the constraints considered; and also of checking the validity of the model.

(vii) total power generated and/or imported must be sufficient to supply the total demand at given time.

The above list is by no means exhaustive: there are many more constraints that the designer could impose. Some of the above constraints are interrelated; some are severe and have decisive influence on the mode of operation and economic choice of the system to be designed, while others are mild and have negligible influence. The designer is particularly concerned with the severe and decisive constraints. The problem then becomes that of determining the degree of importance of the constraints. There are no systematic rules for achieving this; and here is where the designer's experience, skill and judgment plays an important role.* Some of the difficulties involved have been treated at length in Chapter 1.

10.2 Linear Programming Model:

For an example, we shall investigate the simple system shown in Fig. 10.1. It consists of a main generating station and five load centers. The load demand at each center is indicated. Table 10.1 lists the distances between the stations.

The following assumptions are made:

Fig. 10.1. System for the Linear Programming Model

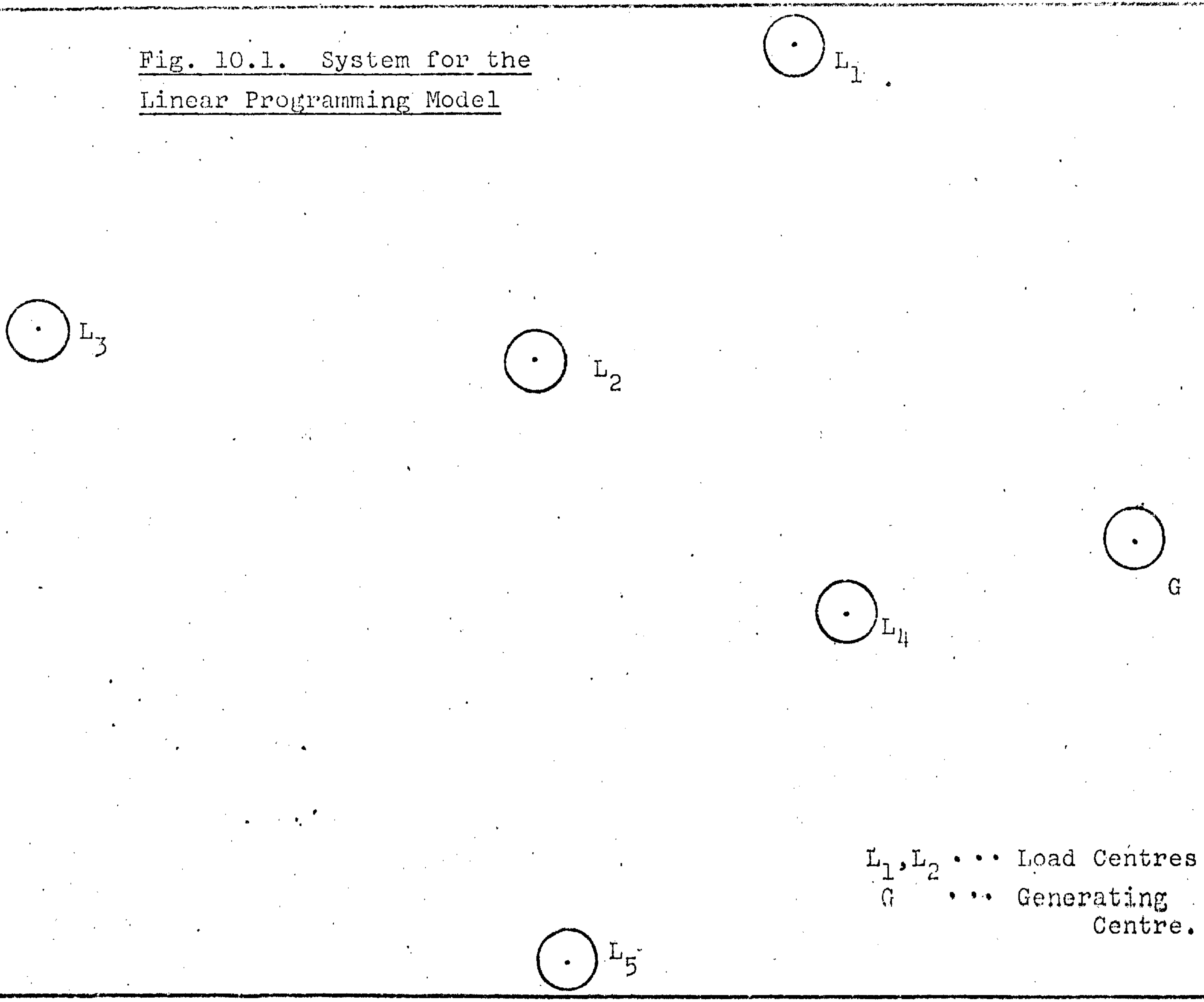


Table 10.1 Distances between Load Centers. (Linear Programming Model)

Path	Length (miles)	Path	Length (miles)
G–L_1	115	L_1–L_5	197
G–L_2	122	L_2–L_3	94
G–L_3	214	L_2–L_4	77
G–L_4	54	L_2–L_5	130
G–L_5	137	L_3–L_4	165
L_1–L_2	78	L_3–L_5	158
L_1–L_3	153	L_4–L_5	113
L_1–L_4	102		

Table 10.2 Distances between Load Centers (Nonlinear Programming Model

Path	Length (Miles)
1–2	140
1–3	164
1–4	180
2–3	144
2–4	260
3–4	144

(a) The generating capacity is sufficient to meet the current load demand;:

(b) costs considered (objective function) are proportional to the length of the line; and to the number of lines in any given path;

(c) only transmission costs are considered;

(d) there is sufficient reactive power equipment for supply and control as required.

10.2.1 Problem Formulation:

The main constraints are based on the security requirement at each substation and also on a modified block transfer requirement to a group of substations. For example it may be required that the number of lines supplying sub station A must be greater than a specified value.

The exact value specified will be based on the degree of reliability envisaged; on the power demand at each substation; on the thermal limit of each line; and on the transfer capacity limitof each line

If constraints specifying all the possible combinations were imposed, the resulting model would be extremely large even for a small size problem. For this example, the method used for imposing constraints incorporates an adaptive approach: the smallest number constraints possible is introduced. This results in a linear programming model, which is then solved by the Simplex algorithm. The solution is then examined; if the resulting network configuration, does not appear 'reasonable', a selected number of constraints is added. The process is repeated until the desired configuration has been obtained.

For the system shown in Fig. 10.1, the following linear programming model was eventually arrived at:*

the load centers should be at least four;

(b there should be at least one line between the generating center and load center one (L_1) ;

(c) there should be at least two lines between the generator and load center 2 (L_2) , load center 4 (L_4) and lcad center 5 (L_5) respectively.

X_i refer to the lines: e.g. line between generator and load center one ($\ell_{01} = \ell_{10} = X_1$) .

* Note that some of these constraints arose from earlier solutions - in the course of the adaptive process described above.

incorporates an adaptive approach: the smallest number constraints possible is introduced. This results in a linear programming model, which is then solved by the Simplex algorithm. The solution is then examined; if the resulting network configuration, does not appear 'reasonable', a selected number of constraints is added. The process is repeated until the desired configuration has been obtained.

For the system shown in Fig. 10.1, the following linear programming model was eventually arrived at:*

$$\text{Minimize} \quad F(\bar{x}) = \sum_{i=1}^{15} C_i X_i \qquad (10.2.1)$$

Subject to:

(a) the number of lines entering each of the load centers should be at least four;

(b there should be at least one line between the generating center and load center one (L_1) ;

(c) there should be at least two lines between the generator and load center 2 (L_2) , load center 4 (L_4) and load center 5 (L_5) respectively.

X_i refer to the lines: e.g. line between generator and load center one $(\ell_{01} \equiv \ell_{10} = X_1)$.

The above formulation results in a set of ten inequalities in fifteen unknowns.

There are several advantages to this approach. First, the number of constraints and hence the size of the problem matrix is reduced considerably: in this example by up to a factor of 4 or more. This makes it possible to handle large problems which would otherwise be toobig for the existing computer capacities. Secondly, the process is similar to the one that a design engineer would follow in practice; and hence should have more appeal. Thirdly (related to the second point) the designer has virtual control over the design process; furthermore, by changing or modifying the constraints, the designer is, in effect, engaged in sensitivity analysis of the relative importance of the constraints.

10.3 Method of Solution

An integer linear programming subroutine - from IBM SHARE Library was used. The subroutine uses the revised simplex method (Chapter 4) and is capable of handling problems not exceeding 120 constraints in 300 unknowns.

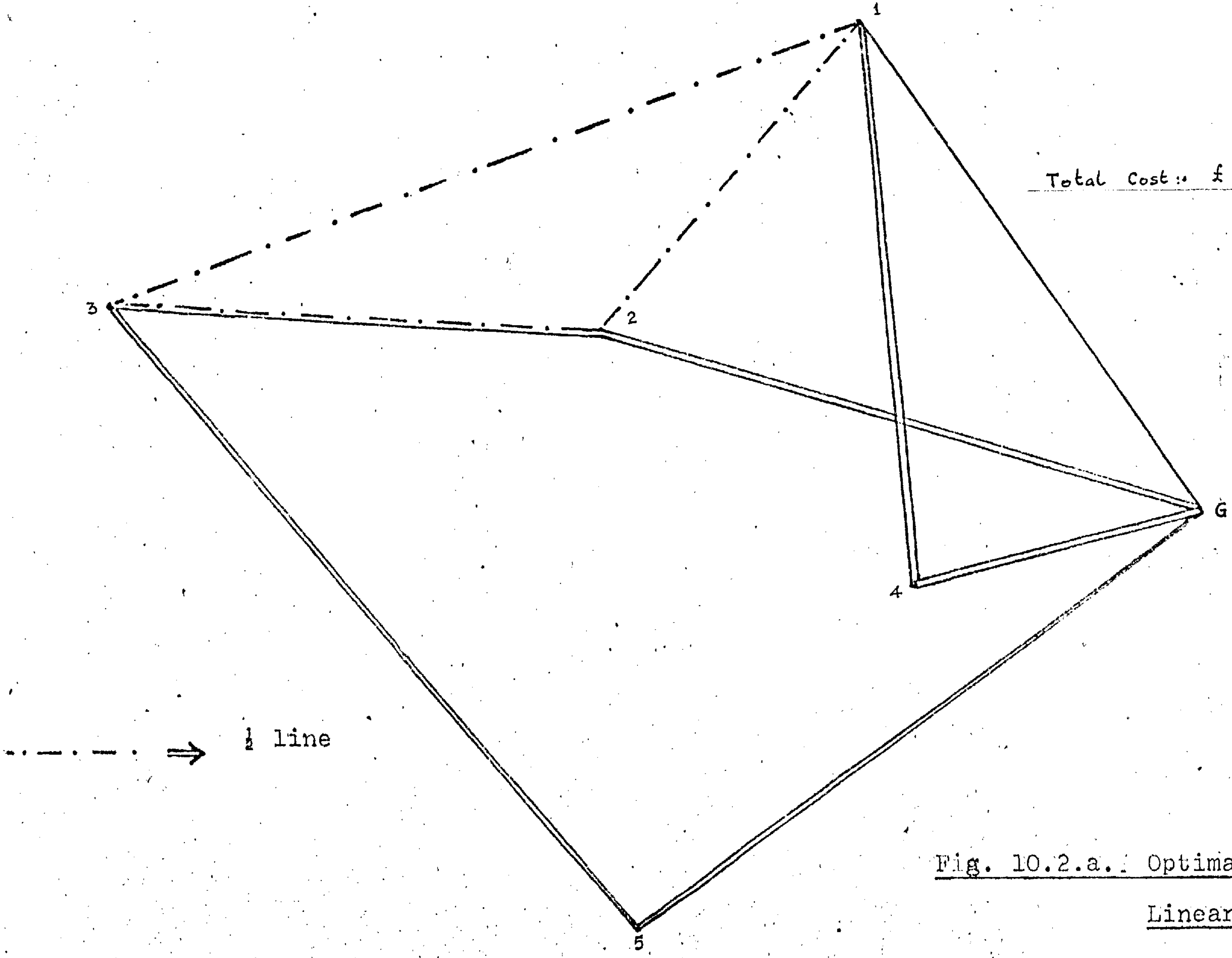
The solution process is as follows: first an optimal solution to the linear programming problem is obtained; a check is next made to determine whether or not the solution is all-integer. If not, a 'Gomory Constraint' is generated for a particular non-integer variable and added to the problem (Chapter 4). The program then seeks an optimal solution satisfying all the constraints (including the Gomory Constraints). The cycle is repeated until all the solution variables are integer.

The limit on the size of the problem (120 constraints in 300 unknowns) is to allow for the additional 'Gomory Constraints' and artificial variables (Chapter 3) to be generated by the program. The version used was written in Fortran II code; and the computation time is about 0.4 minutes, including compiling time (for a binary deck).

10.4 Results:

Fig. 10.2.a shows a network configuration resulting from an optimal linear programming solution. The dashed lines refer to $^1/2$ line solution. Fig. 10.2.b. shows the configuration of an all integer solution to the same problem.

The imposition of the requirement for an all-integer solution to a linear programming problem tends



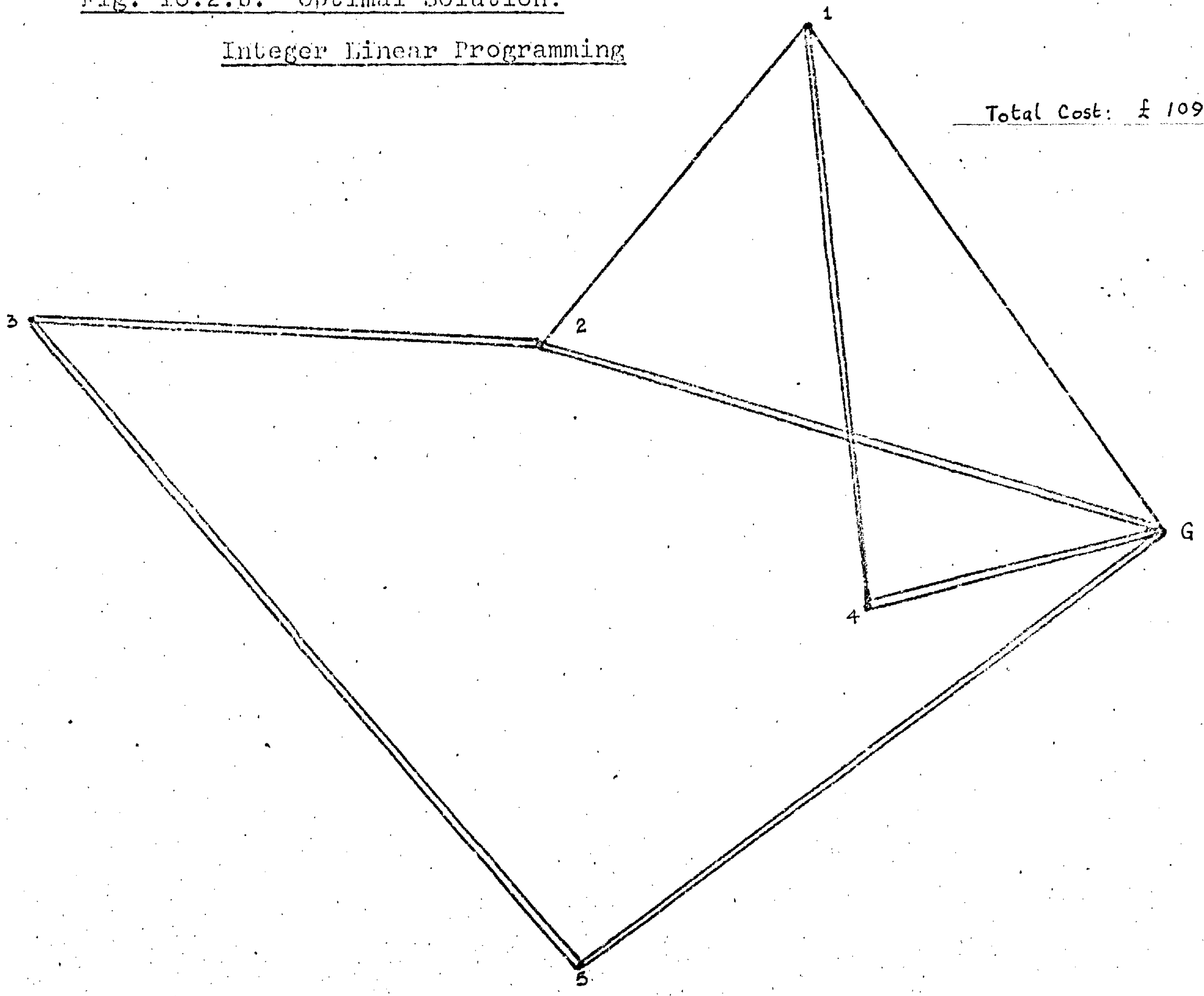
Total Cost : £ 108.74×10^6

Fig. 10.2.a.: Optimal Solution

Linear Programming

Fig. 10.2.b. Optimal Solution:

Integer Linear Programming



Total Cost: £ 109.41 x 10^6.

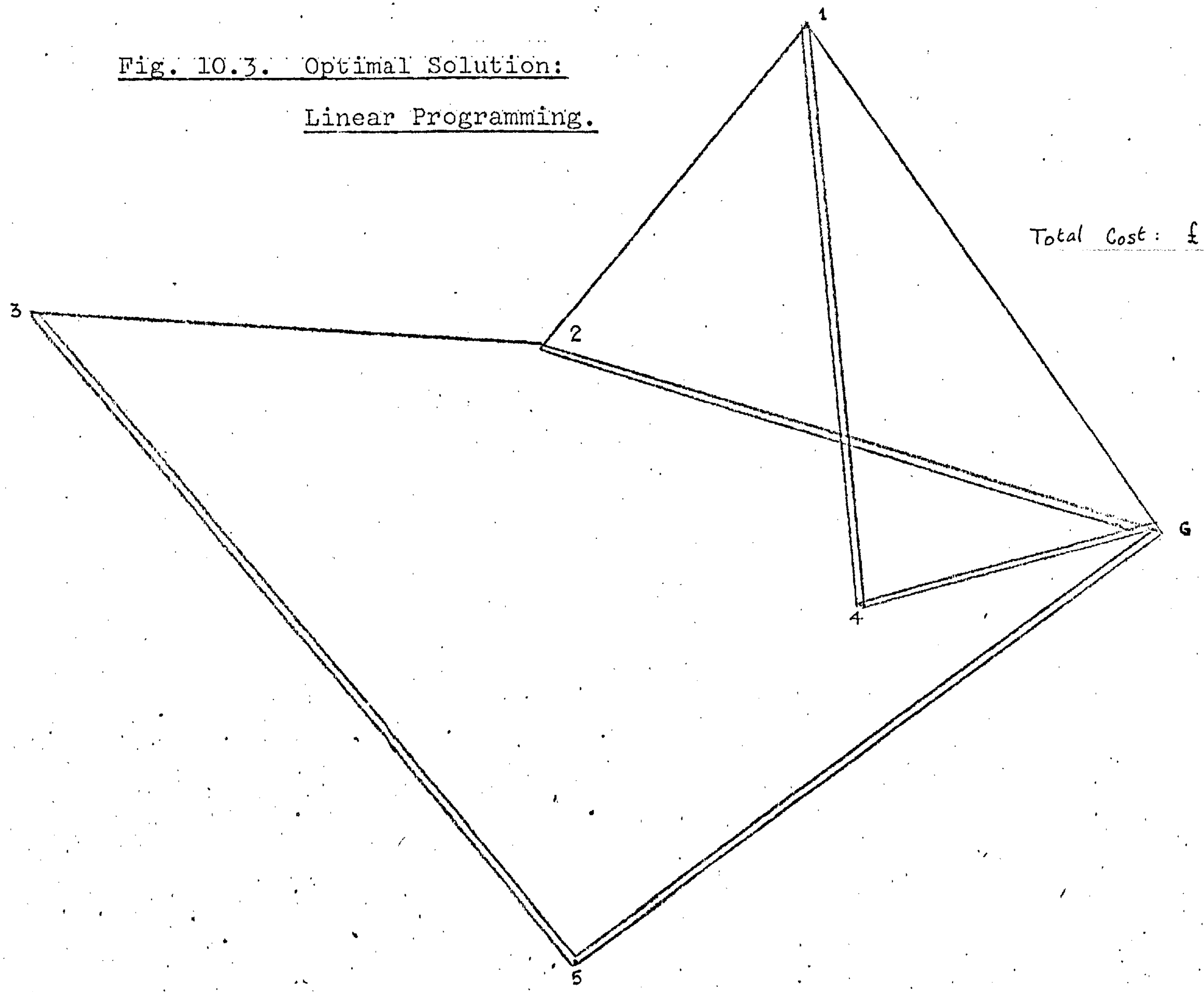
to increase the total cost of the design, as is clearly indicated by comparing the total costs for configurations 10.2.a and 10.2.b. (cf. £108.745 x 10^6 and £109.41 x 10^6)

If for the problem considered it was required that the number of lines entering load center 3 (L_3) be three while all the other constraints remained the same, the resulting optimal solution (both linear programming and integer linear programming) would be as shown in Fig. 10.3.

The requirement that the number of lines entering load centers 1, 2, 3, 4, 5 be 4, 4, 3, and 4 respectively for the load demands indicated may be too conservative. For with the present reactive power regulation capabilities, lines are capable of carrying much larger loads[12]. In which case the requirement should be that the number of lines entering load centers 1, 2, 3, and 5 should be 3, 3, 2 and 3 respectively. With these latter constraints, together with the added restraints that there should be at least a line each between the generating station and load centers 1, 2, 4 and 5 respectively, a model is obtained whose optimal solution is shown in Fig. 10.4.

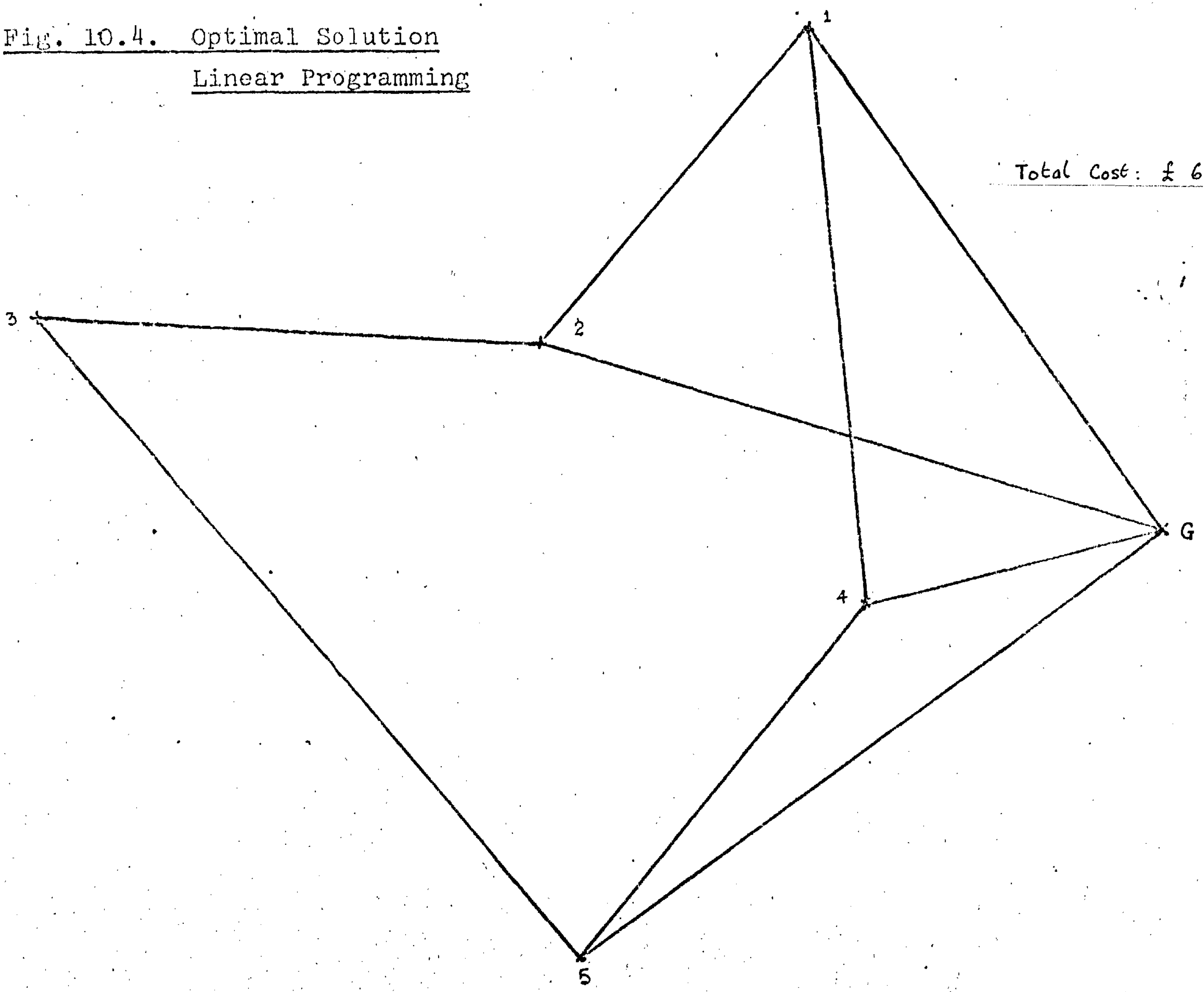
Fig. 10.3. Optimal Solution:

Linear Programming.



Total Cost: £ 102.83 $\times 10^6$

Fig. 10.4. Optimal Solution Linear Programming



Total Cost: £ 69.37 x 10^6

Some General Comments:

The above example has shown that a power system design problem can be formulated as a linear programming problem and solved. However, it must be emphasized that there are definite limitations to this approach.

Firstly, the essence of linear programming is the existence of linear relations, both in the objective function, and amongst the variables themselves. In the above example we have assumed that the cost of lines is linearly related to both the length of the line and the length of lines per given path. Such an assumption is very idealistic; for in some situations, local conditions may be such as to make a short line more expensive to construct and operate than a relatively longer line. Moreover, the cost of two lines between, say A and B is usually less than double the cost of one line between the same.

Secondly, linear programming -- and especially integer linear programming - solutions tend to act in an all-or-nothing discrete fashion; and not in a continuous fashion. Consequently, a small change in the coefficient of the variables may either change nothing or effect quite a large change. Due account should be taken of this fact when interpreting the results.

However, apart from the above limitations, it is felt that, linear or integer linear programming solutions to design problems of the type considered above forms a very useful guide to the type of configuration that the system might eventually attain. This, of course, refers to the case of designing a system where none existed before; or more likely, to the situation where a higher voltage system is to be superimposed on the existing lower voltage one: for example, in Great Britain, the development of the 275 kv system to reinforce the existing 132 kv one or the current development of the 400 kv system.

10.5 A Nonlinear Programming Model:

A much simpler model shown in Fig. 10.5 is next investigated. Table 10.2 shows the distances between the substations.

The objective function (the cost of transmission lines) is still assumed to be linear; but the constraints now bear nonlinear relations. As in the previous section, the main constraints are based on the security requirement at each substation. The major constraints are:

(i) total power requirement at each substation

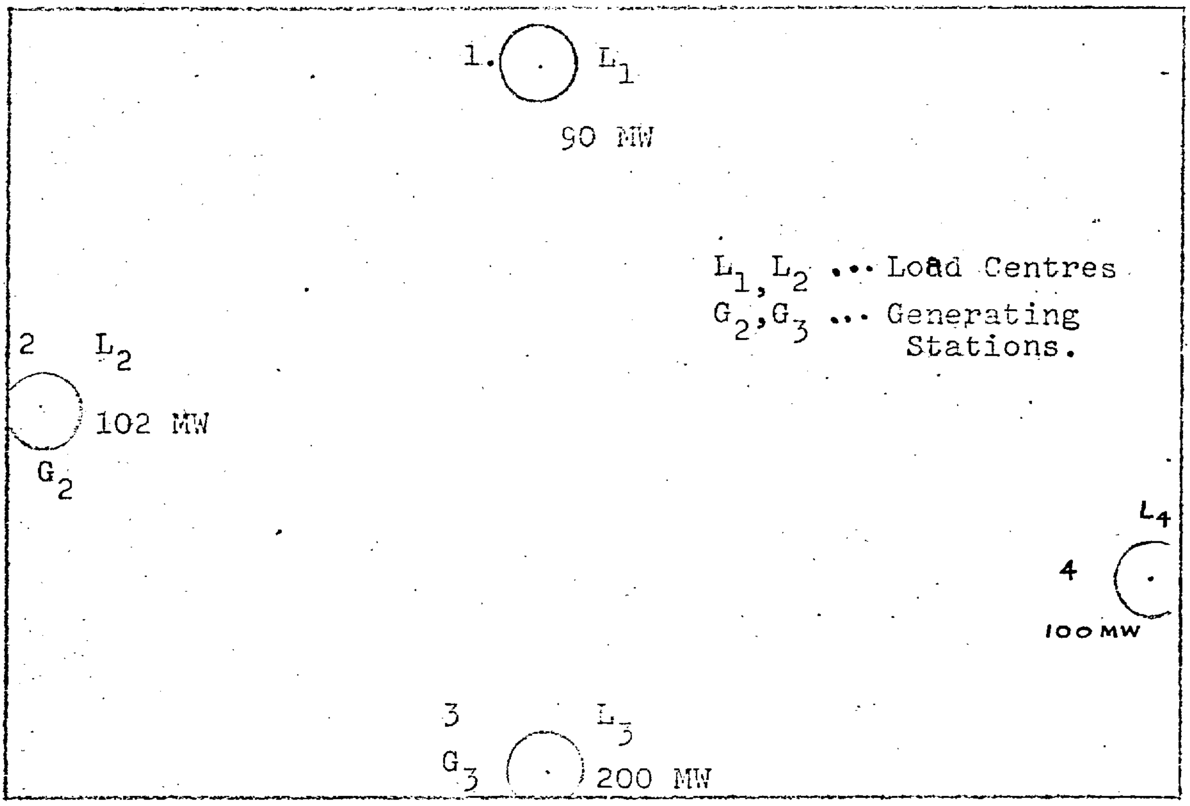

1.
L_1
90 MW
L_1, L_2 ... Load Centres
G_2, G_3 ... Generating Stations.
2
L_2
102 MW
G_2
L_4
4
100 MW
3
L_3
G_3
200 MW

Fig. 10.5: System for Solution by Non-linear Programming Methods

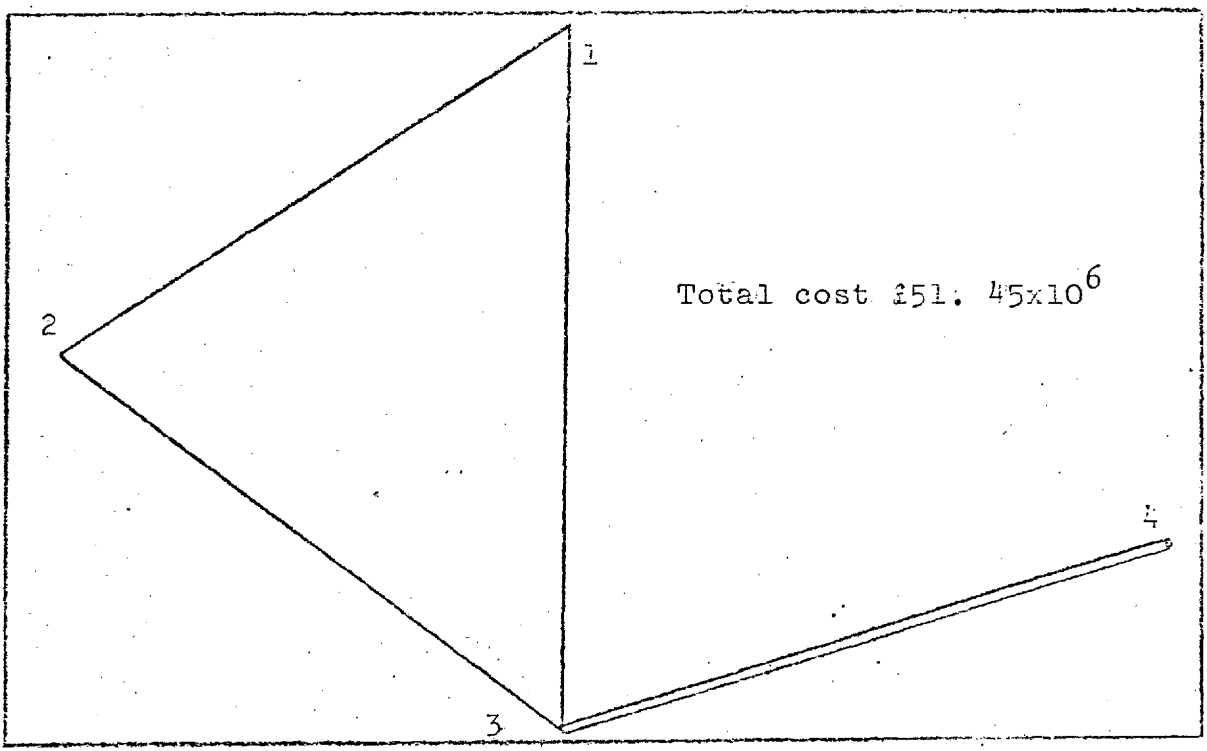


Fig. 10.6: Optimal Linear Programming Solution to the System Shown in Fig. 10.5.

Table 10.3 Optimal Solution to the Nonlinear Programming Model

Path	Number of lines
1-2	.579
1-3	1.367
1-4	0.011
2-3	1.613
2-4	.380
3-4	1.599

Total Cost =£ 60.790 x 10^6

Table 10.4 Optimal Solution of the Linear Programming Model by Means of the Logarithmic Potential Method:

G-L_1	.996	L_1-L_5	.004
G-L_2	1.994	L_2-L_3	1.498
G-L_3	.001	L_2-L_4	.003
G-L_4	1.999	L_2-L_5	.003
G-L_5	1.998	L_3-L_4	.003
L_1-L_2	.500	L_3-L_5	.003
L_1-L_3	2.491	L_4-L_5	1.985
L_1-L_4	.006		

Total Cost = £109.484 x 10^6

should be able to meet the demand at the particular sub-station. The actual formulation is based on the relation between power transfer capacity and the length of a given line [12] :

$$P = .5 \times \frac{(kv)^2}{\ell} \quad \text{in} \quad M.W. \qquad (10.5.1)$$

where ℓ is the distance in miles

So that the actual constraints are:

$$\Sigma\, P_{i1} \geq \text{specified demand at station 1} \qquad (10.5.2)$$

$$\Sigma\, P_{i2} \geq \text{" " " " 2} \qquad (10.5.3)$$

$$\vdots$$

$$\Sigma\, P_{i4} \geq \text{" " " " 4} \qquad (10\ 5.4)$$

$$i = 1,\ 4$$

(ii) for each line j , the power transferred must be less that a specified value governed by the stability limit of the line in question:

$$P_{i,j} = \frac{.5x(kv)^2}{\ell_{i,j}} \leq \text{specified value} \qquad (10.5.5)$$

$$i,j = 1, \ldots, 4,\ i \neq j,\ P_{i,j} = P_{j,i},\ \ell_{i,j} = \ell_{j,i}$$

These, together with the requirement that $\ell_j \geq 0$ $j = 1, \ldots, 6$ constitute the set of constraints: 22 constraints in 6 unknowns.

A logarithmic potential method (Chapters 5 and 9) was used to obtain a solution to the above problem. As has been explained, this is a nonlinear programming technique which transforms the constrained problem into a series of unconstrained problems, which are then solved by means of unconstrained optimization techniques.

The result is shown in Table 10.3. A similar result is obtained when in addition to the above constraints, a set of reliability constraints is added. This may be due to the fact that the constraints (reliability) are not tight enough. Or alternatively, that in a design problem of the type considered, the particular set of reliability constraints are of relatively less importance than the other ones.

The method employed in generating reliability constraints is given in Appendix A6. It is based on estimating the length of time during which a substation should operate without power interruption to the consumer, for a given year; for example, if it is assumed that 2 lines are required to supply load center one; then during a fault the following possibilities may obtain:

(i) no line is affected;

(ii) line one is out of service;

(iii) line two is out of service;

(iv) both lines are out of service.

If it is further assumed that the transfer capacity of each line is such that by itself, it can supply the demand of the particular substation, the interruption of service to the consumer occurs only when both lines go out of service. This fact is used to estimate the expected time during which no interruption should occur at the particular sub-station. The design requirement is then that the cost should be minimized subject to the expected un-interrupted running time being greater than the estimated value. Eq:*

*↖ for each substation we have

$$\sum_{(i,j)\in FS} \left\{ \frac{\prod_{k=1}^{n} M_{ik}}{U_{\ell(i,j)}} \right\}^{-1} \sum_{i\in g} \prod_{k=1}^{n} M_{ik} \geq \text{specified value}$$

where: M_{ik}, $\ell(i,j)$, g, U, and D are as defined in Appendix A6.

present glaring limitation of the nonlinear programming techniques in current use. There is as yet no known way of ensuring that the solution be all integer. However, it is encouraging to learn that some serious[76] research effort is being directed at this problem (integer nonlinear programming) and it is hoped that some useful algorithm will be forthcoming.

(iii) line two is out of service;

(iv) both lines are out of service.

If it is further assumed that the transfer capacity of each line is such that by itself, it can supply the demand of the particular substation, the interruption of service to the consumer occurs only when both lines go out of service. This fact is used to estimate the expected time during which no interruption should occur at the particular sub-station. The design requirement is then that the cost should be minimized subject to the expected un-interrupted running time being greater than the estimated value. E.g: *

For the simple problem considered the introduction of reliability requirements led to the addition of 26 more constraints.

An important feature of the results is that the solution variables are non-integer. This is the present glaring limitation of the nonlinear programming techniques in current use. There is as yet no known way of ensuring that the solution be all integer. However, it is encouraging to learn that some serious[76] research effort is being directed at this problem (integer nonlinear programming) and it is hoped that some useful algorithm will be forthcoming.

A linear programming solution of the. same problem (Fig. 10.5) is shown in Fig. 10.6. The method of formulation is similar to the one discussed in Sections 10.2 and 10.3. This is clearly a much better solution than the one by nonlinear programming technique.

As a matter of general interest, the logarithmic potential method was used to solve the linear programming model discussed in sections 10.2 and 10.3 and whose optimal solution is shown on Figs. 10.2a and 10.2.b. Table 10.4 shows the final solution after 10 functional evaluations (minimization cycles) and 6 minutes of computer time. The rate of convergence was extremely slow, and the final value of the objective function still far from the exact minimum. The linear programming result (Simplex method) on the other hand, was obtained after 0.4 minutes of computer time (including compilation time); the result is shown in Fig. 10.2.a. This confirms my observation in Chapter 2 that for a linear programming model, the simplex algorithm is the most efficient method of solution

CHAPTER 11

GENERATION PLANNING

11.0

Practically all power utilities face the necessity of having to decide upon the increase in the generating capacity to meet the growing load demand.

Generation planning does, in fact, involve the investment of a large amount of money. It is, therefore, imperative that a careful analysis be carried out to ensure that the most efficient and economic combination of the units is added into the system.

Such analysis requires a thorough evaluation of a large number of factors bearing on the problem: for example

(a) prediction of load demand;

(b) estimation of the load duration curve;

(c) reserve capacity requirements;

(d) availability of the installed capacity;

(e) inter-connection of utility systems;

(f) expected (or estimated) maximum unit size in the latter stages of the planning period;

(g) expected development in the methods of electricity generation (conventional or otherwise);

(h) location of the units and topography to the terrain;

(i) expected transmission system development;

(j) capital and operating costs of different sizes and types of units etc.,

Evaluation of all or most of the above factors so as to arrive at an optimum choice is quite a difficult task, especially if planning studies are carried out over long periods of time. For there is a large number of possible alternatives to choose from.

First, there are many different types of generating units that could be added to a system. These include different types and sizes of thermal and/or hydro-electric units or nuclear units. The units may be for base of peak loads. The choice may also be between installing only units having the best available fuel rates (at high capital costs) or for a mixture of high and low fuel rates. Still another possibility may be to consider the use of only very large units; or a mixture of large and small units.

The final choice should only be made after considering a large fraction of all the possible alternatives.

Much research effort has been directed at the problem of determining the 'best' combination of the generating units to add - with regard to both capital and operating costs over long periods of time. In this chapter the possibility of applying the methods of dynamic programming; and sequential unconstrained optimization techniques are suggested and discussed.

11.1 Problem formulation: A General Case:

In very general terms, the planning problem may be formulated as follows:

Let load duration curves be given for each year of the entire planning period; this information is obtained from the expected load curves. A typical load duration curve is shown in Fig. 11.1. The objective is then to minimize the total capital and operating cost over the period of study subject to the following constraints:

(a) maximum demand in each year must be met;

(b) there should be adequate reserve capacity;

(c) allowance for planned outage of the installed generators for repair;

(d) energy output from the units must be sufficient to meet the demand in each time interval of the load duration curve for each period of time considered;

(e) maximum allowable capacity of the unit to be installed not to exceed a specified limit (for each year of the study period);

(f) maximum available energy output for each type of plant for each time period should be within certain specified limits.

The above points may be summarized in the following compact mathematical programming formulation:

$$\text{Minimize:} \quad F(P_{it}, E_{itm}, E_{etm}) = \sum_{i}^{I} \sum_{t}^{T} C'_{it} P_{it} +$$

$$\sum_{i=1}^{I} \sum_{t=0}^{T} \sum_{m=1}^{M} E_{itm} C_{it} + \sum_{e=1}^{N} \sum_{t=0}^{T} \sum_{m=1}^{M} E_{etm} C_{et} \quad (11.1.$$

Subject to:

$$\sum_{i=1}^{I} \sum_{t=1}^{T} P_{it} + \sum_{e=1}^{N} \sum_{t=0}^{T} P_{et} \geq P_{tm} \qquad (11.1.2)$$

$$\sum_{i=1}^{I} E_{itm} + \sum_{e=1}^{N} E_{etm} \geq E_{tm} \qquad (11.1.3)$$

$$P_{it} \leq P_{it} \qquad (11.1.4)$$

$$E_{itm} \leq \alpha_{im} \; P_{it} \qquad (11.1.5)$$

$$E_{etm} \leq \alpha_{em} \; P_{et} \qquad (11.1.6)$$

$$E_{itm}, E_{etm}, P_{it}, P_{et} \geq 0 \qquad (11.1.7)$$

where P_{it} refers to the capacity of plant of type i to be added in period t ; E_{itm} represents energy output of plant of type i in the interval m of period t ; P_{et} refers to capacity of the existing plant, type e , in period t ; E_{etm} represents energy output of plant of type e in the interval .m of period t ; C'_{it} , C_{it} and C_{it} are cost coefficients and α_i and α_e are availability factors for the plants to be installed and the existing plants respectively and P_{it} is a specified value.

Requirements (b) and (e) are allowed for in equations (11.1.2) and (11.1.3) respectively. Thus P_{tm} is increased by the required reserve capacity; similarly E_{tm} is increased by the amount corresponding to the energy output of the planned output for the period and time intervals considered.

Note that the above formulation does not include transmission and geographical constraints. These can, however, be incorporated without loss of generality, as has, indeed, been done by U.G. Knight in his formulation of the linear programming model.[49]

11.2 Possible Solution by Nonlinear Programming Methods:

As has been pointed out in Chapter 5, methods for solving nonlinear programming problems are still in their infancy and have not yet acquired the degree of efficiency enjoyed by the 'Simplex' Method for solving linear programming problems. One of the areas that require further investigation is the size (number of variables and/or constraints) and complexity of the problems that the existing nonlinear programming methods can handle.

Such information is clearly of immense interest to planning engineers. For formulations of the type given above would involve several hundreds of constraints and about the same number of variables - if the planning study is to be meaningful.

Once the information is forthcoming, the methods could become very useful tools in the search for the 'best' plan. For example, the above problem could be solved (11.1.1 through to 11.1.7) by one of the sequential unconstrained optimization techniques (e.g. the Logarithmic Potential Method). In which case the problem is transformed to:

$$\text{Minimize}\quad FP(r, P_{it}, E_{itm}, E_{etm}) = F(P_{it}, E_{itm}, E_{etm}) -$$

$$r\Big\{\ln.\Big(\sum_{i}\sum_{t} P_{it} + \sum_{e}\sum_{t} P_{et} - P'_{tm}\Big) + \ln\Big(\sum_{i} E_{itm} + \sum_{\ell} E_{etm} - E'_{tm}\Big)$$

$$+ \ln\Big(\hat{P}_{it} - P'_{it}\Big) + \ln\Big(\alpha_{im}\sum P_{it} - E'_{itm}\Big) + \ln\Big(\alpha_{em}\sum P_{et} - E'_{etn}$$

$$+ \ln\Big(E'_{\underset{\ell}{\cancel{i}}tm}\Big) + \ln\Big(E'_{etm}\Big) + \ln\Big(P'_{et}\Big)\Big\} \qquad (11.2.1)$$

Subject to no constraints. Where P'_{tm}, E'_{tm}, P'_{it}, E'_{itm}, E'_{etm} are equal to $P_{tm}+\varepsilon_1$, $E_{tm}+\varepsilon_2$, $P_{it}+\varepsilon_3$, $E_{itm}+\varepsilon_4$ and $E_{etm}+\varepsilon_4$ respectively: ε_i $i = 1,\ldots,4$ is a small constant greater than zero.

Alternatively the original problem could be solved by Zangwill's method of 'Penalty Functions' (Chapter 5). Thus the unconstrained optimization problem would be:

$$\text{Minimize}\quad FP = F + \frac{1}{r}\Big\{(G_1)^2 + (G_2)^2 + (G_3)^2 + (G_4)^2$$

$$+ (G_5)^2 + (G_6)^2 + (G_7)^2 + (G_8)^2\Big\} \qquad (11.2.2)$$

where

$$G_1 = \begin{cases} (\sum_i \sum_t P_{it} + \sum_e \sum_t P_{et} - P'_{tm}) & \text{if } 11.1.2 \text{ is violated} \\ 0 & \text{otherwise} \end{cases} \qquad (11.2.3)$$

similarly

$$G_2 = \begin{cases} (\sum_i E_{itm} + \sum_a E_{etm} \quad E'_{tm}) & \text{if } 11.1.3 \text{ is violated} \\ 0 & \text{otherwise} \end{cases} \qquad (11.2.4)$$

and so on for the rest of the constraints.

It is hoped that some research effort into the applicability of these methods to power system planning (as suggested above) will be forthcoming.

11.3 Possible solution by Dynamic Programming

The generation planning problem can also be formulated by means of the direct application of the principles of dynamic programming.[49] However, a major limitation of the dynamic programming approach (Chapter 6) is the computational difficulty involved in handling problems with greater than four constraints. So that with the planning models of the type envisaged (with hundreds of constraints) such direct application of dynamic programming is strictly out of the question.

In a recent article Dale has suggested a dynamic programming method for the selection and timing of generation plant additions.[14] His approach is based on the selection of several basic types and sizes of the units which are then combined in a large number of ways for each of the time intervals in the entire planning period. Each of these combinations will result in different costs; and at least one of them will give the least-cost plan.

The main weakness of Dale's method is that the shape of the load duration curve is not taken into account. As a possible improvement on his approach the author makes the following suggestions, which effectively take into account the shape of the load duration curve:

The load duration curve for each year should be divided into three (or more) sections, comprising base load, 'medium' load and peak load. See fig 11.2 for an illustration. In general the base load for a given system will increase by a certain percentage from year to year. So will the medium and the peak loads respectively Consequently, three sets of curves can be drawn for the predicted growth in the base, medium and peak curves respectively (Figs. 11.3.a, 11.3.b and 11.3.c).

With the above information available, Dale's approach can now be applied to obtain the 'best' plan for each of the divisions. For example, suppose that, for the base load division we have three possible patterns A_b, B_b and C_b of developments for high-load-factor, high capital cost units. For each pattern, say A_b, we may have several different types (and sizes) of units e.g. nuclear, coal fired, oil-fired or hydroelectric; this will give rise to numerous combinations for the entire planning period as illustrated in Fig. 11.4. As the stage (year) of study increases, so does the number of possible combinations of the units. For a given stage, not all the possible number of combinations need be considered: the number may be reduced by allowing only for the most likely combinations. This would, of course, depend on the policy of the particular electric utility and may be influenced by whether or not the system under investigation is an isolated one; or is part of a larger integrated system.

A dynamic programming algorithm can then be used to obtain the most economic plan for pattern A_b. By the same approach economic plans for patterns B_b and C_b are obtained. One of the three results gives

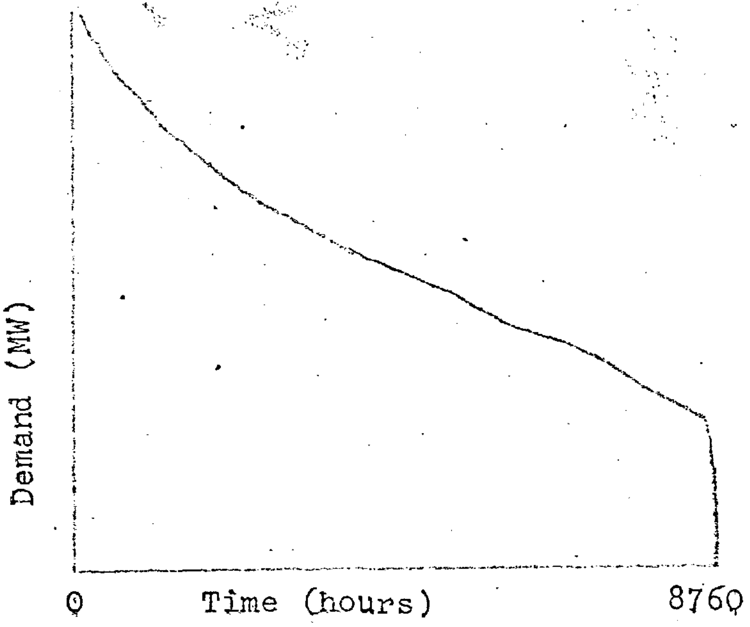


Fig. 11.1 A Typical Load Duration Curve

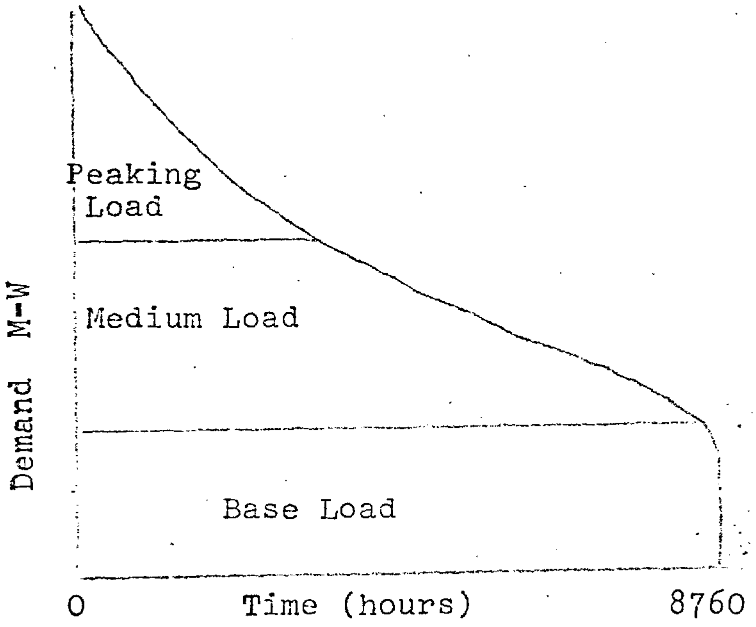


Fig. 11.2. Suggested divisions in the Load Duration Curve

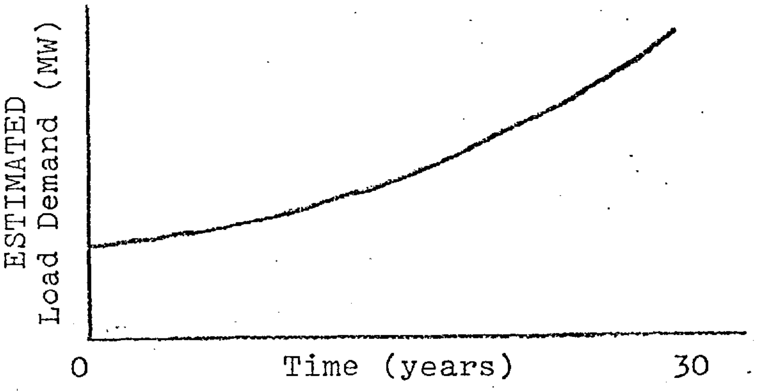


Fig. 11.3.a Predicted Load-growth for base load

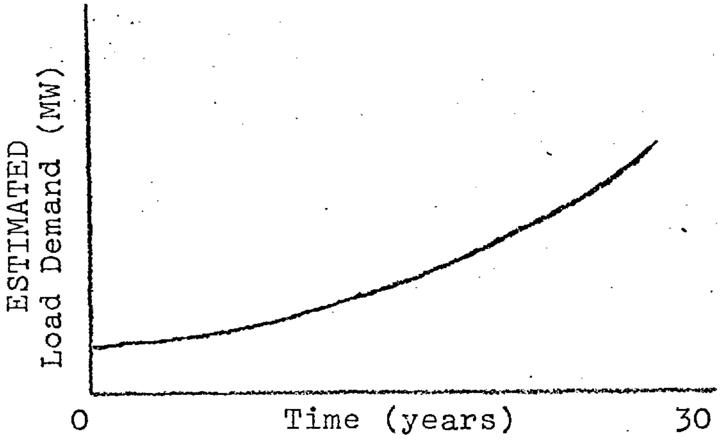


Fig. 11.3.b Predicted Load-growth for 'medium' load

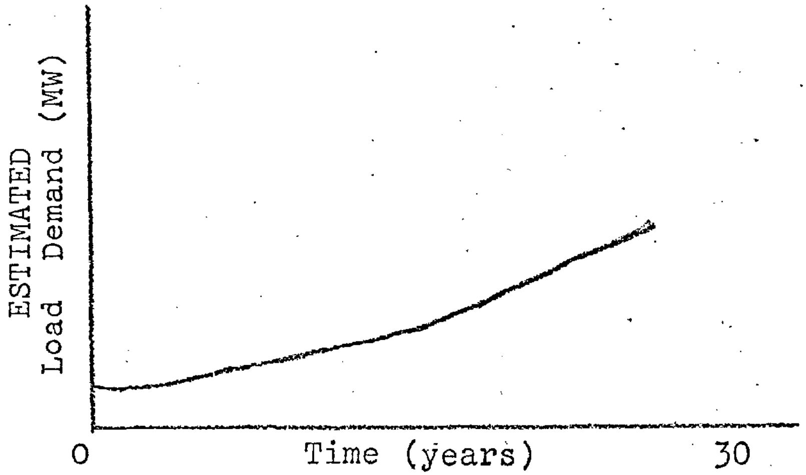


Fig. 11.3.c Predicted Load-growth for peak loads.

Stage (years)

State	0	1	2	
1	Existing System	none	none	none
2		$N_{A_{b1}}$	$N_{A_{b1}}$	$N_{A_{b1}}$
3		$N_{A_{b2}}$	$N_{A_{b2}}$	$N_{A_{b2}}$
4		$C_{A_{b1}}$	$C_{A_{b1}}$	$C_{A_{b1}}$
5		$C_{A_{b2}}$	$C_{A_{b2}}$	$C_{A_{b2}}$
6		O_{A_b}	O_{A_b}	O_{A_b}
7		H_{A_b}	H_{A_b}	H_{A_b}
8			$N_{A_{b1}} + N_{A_{b2}}$	$N_{A_{b1}} + N_{A_{b2}}$
9			$N_{A_{b1}} + C_{A_{b1}}$	$N_{A_{b1}} + C_{A_{b2}}$
10			$N_{A_{b1}} + C_{A_{b2}}$	$N_{A_{b1}} + C_{A_{b2}}$
11			$N_{A_{b1}} + O_{A_b}$	$N_{A_{b1}} + O_{A_b}$
12			$N_{A_{b1}} + H_{A_b}$	$N_{A_{b1}} \quad H_{A_b}$
13			$N_{A_{b2}} + C_{A_{b1}}$	$N_{A_{b2}} + C_{A_{b1}}$
14			$N_{A_{b2}} + C_{A_{b2}}$	$N_{A_{b2}} + C_{A_{b2}}$
15			$N_{A_{b2}} + O_{A_b}$	$N_{A_{b2}} + O_{A_b}$
16			$N_{A_{b1}} + H_{A_b}$	$N_{A_{b2}} + H_{A_b}$
17			etc	$2N_{A_{b1}}$
18				$2N_{A_{b2}}$
19				$2C_{A_{b2}}$
20				$2O_{A_b}$
21				$2N_{A_{b1}} + H_{A_b}$
22				$2N_{A_{b2}} + O_{A_b}$
23				etc

Where $N_{A_{b1}}$: nuclear unit of size 1 in planning pattern A_b

$N_{A_{b2}}$: " " " " 2 " " " "

$C_{A_{b1}}$: coal fired" " " 1 " " " "

$C_{A_{b2}}$: " " " " " 2 " " " – A_b

O_{A_b} : oil fired unit of a specified size for pattern A_b

H_{A_b} : hydroelectric" " " " " " " A_b

Fig. 11.4: Possible Combinations of units in Planning Pattern A

the lowest cost plan for the base load expansion program.

A similar selection method is applied to the medium-load factor units (the medium load division); and then to the low load fact, low-efficiency units for peaking purposes.

Suppose that for the base load program pattern A_b gives the lowest cost; for medium load program pattern B_m gives the lowest cost; and for peaking units pattern A_p gives the lowest cost. Then the overall lowest cost expansion program is the sum: $(A_b+B_m+A_p)$**.

Appendix A7 gives a typical flow diagram of a dynamic programming algorithm that may be used to obtain solutions along the lines suggested above.**

** As a further refinement, the 'base', 'medium' and 'peaking' load divisions (Fig. 11.2) may be shifted ('up' or 'down') and a new set of predicted load growth curves - similar to Figs. 11.3.a, 11.3.b. and 11.3.c - obtained; and another overall lowest cost programme $(A'_b + B'_m + A'_p)$ calculated. If $(A'_b + B'_m + A'_p)$ is less than $(A_b + B_m + A_p)$ then the former is taken as the best plan.

the lowest cost plan for the base load expansion program.

A similar selection method is applied to the medium-load factor units (the medium load division); and then to the low load fact, low-efficiency units for peaking purposes.

Suppose that for the base load program pattern A_b gives the lowest cost; for medium load program pattern B_m gives the lowest cost; and for peaking units pattern A_p gives the lowest cost. Then the overall lowest cost expansion program is the sum: $(A_b+B_m+A_p)$[x].

Appendix A7 gives a typical flow diagram of a dynamic programming algorithm that may be used to obtain solutions along the lines suggested above.[xx]

It must be emphasized that the above discussions are only suggestions and that no actual results are available to confirm their validity.

[x] the subscripts refer to the divisions e.g $A_b \equiv$ pattern A in the base load division; $B_m \equiv$ plan B in the 'medium' load division and $A_p \equiv$ plan A in the peak load division.

CHAPTER 12

CONCLUSIONS AND SUGGESTIONS FOR FURTHER WORK

12.0. "Systems Approach" is finding wider application in a large variety of situations. An essential property of the Approach is that of seeking to optimize the overall systems functions according to specified objectives, and to achieve the best compatibility of its parts.

Systems Engineering forms one of the major components of the Systems Approach. An important step in the systems engineering methodology involves either mathematical or physical modelling.

Formulation of models is a very difficult task requiring quantitative knowledge of how the system variables interact and the relative importance of the constraints. Further difficulties involved in modelling include: finding a suitable way of expressing the objective function in terms of the variables; limiting the number of constraints; and deciding on how much idealization is allowable and still obtain satisfactory results.

12.1. Part I

Mathematical programming has witnessed a phenomenal rate of growth over the last decade; and the fast growth-rate is likely to be maintained for a long time to come.

Lagrange multipliers in one form or another have played an important role in the recent development of mathematical programming theory. One of the most important theoretical results in this field is the work due to Kuhn and Tucker, which is an extension of the classical Lagrange multipliers rule in its most general form (that encompasses both equality and inequality constraints).

Associated with each linear and non-linear programme is the dual programme. The concept of duality has led to the development of a number of very useful computational techniques in both linear and non-linear programming fields; and is currently being successfully applied in the generation of a two-level decomposition technique.

One of the most efficient and widely used algorithms is the simplex method developed by Dantzig for solving linear programming problems. Several variants of the simplex method, including the dual-simplex, primal-dual and self-dual algorithms, have since been developed; and are finding useful application especially for parametric problems.

Several algorithms have also been developed for solving integer linear programming problems. Some of these: e.g. the cutting-plane, primal and mutual-primal, methods employ the principles of the simplex method (or variants of this). In general the cutting-plane and primal methods have proved erratic: sometimes performing well and at others giving no solutions. One of the most promising developments is that of the branch and bound algorithms - especially the version of Bala's partial enumeration algorithm which requires less storage capacity and computation time and which is now being extended to handle non-linear integer programming problems.

Non-linear programming techniques may be broadly classified into direct and indirect methods. Of the indirect methods one of the latest techniques is the method of geometric programming, which has definite potentialities in a variety of fields.

Much of the discussions have been concerned with the direct methods, which may be further grouped into the unconstrained and the constrained optimization techniques. Of the unconstrained methods, the modified-gradient ones especially the Fletcher-Powell algorithm has, by far, the best convergence properties. However, the algorithm suffers from the fact that the positive definite matrix, H, required may be quite large for large problems, thus limiting the size of problems to consider. Another modified gradient technique: the method of conjugate gradients, excels in its simplicity. However, it does not converge as fast as the Fletcher-Powell method.

The constrained non-linear optimization techniques may be divided into: those which are extensions of the simplex method e.g. reduced gradient methods or methods of approximation programming; methods of feasible directions; and sequential unconstrained optimization methods.

The essential feature of the first division (e.g. reduced gradient method) is the approximation of the non-linear function by a linear one through the use of Taylor Series expansions. Some of these methods exhibit reasonably fast convergence properties; others are quite slow while others are only capable of solving convex problems.

There are many versions of the method of feasible directions. These use the same general approach as the methods of unconstrained optimization. So far the versions due to Zontendijt, Rosen and Goldfarb have performed quite satisfactorily, especially for non-linear programming problems with linear constraints.

The sequential unconstrained optimization methods (also known as "penalty function methods") transform a given constrained problem into a sequence of unconstrained problems, which are then solved by unconstrained optimization techniques (e.g. modified gradient methods). They exhibit good convergence and are capable of handling highly non-linear problems. Of these, both the Logarithmic potential method and the Fiacco-McCormick method require an initial feasible solution - a procedure that may be time-consuming, especially for large problems. With Zangwill's method, however, the initial solution is chosen arbitrarily, and a penalty is imposed only on the constraints that are violated. This makes the method capable of handling large problems.

All the three penalty function methods suffer from one major weakness: their inability to distinguish between non-linear and strictly linear constraints. If special subroutines for handling linear constraints could be incorporated into the algorithm the methods would become more efficient. Further research work is also required to establish the following: a) computationally the most efficient way of solving the unconstrained problems; b) the best way of choosing the initial value of r; and c) the best way of decreasing r.

Dynamic programming is a very efficient method for solving multi-stage decision processes. The problems can be linear or non-linear. However, as the dimension of the problem increases, storage capacity requirements multiplies manifold. Consequently, only small dimension multi-stage decision problems can be solved by the existing class of computers.

Although not as versatile as dynamic programming, the Maximum Principle is beginning to find increasing applications in a large variety of problems. The concepts have further been extended to the discrete case and should now find some applications in the field of power systems engineering. Close similarities exist between the fundamental concepts of the Maximum Principle and those of Mathematical Programming in general.

Several successful decomposition techniques for solving large linear programming problems have been developed. The basis of these algorithms is similar to the 'simplex' method. Others, including decomposition by dynamic programming and Kron's method of 'diakoptics' are still at the development stage. Current research effort is now directed at decomposition techniques for non-linear programming problems and it is hoped that efficient algorithms will be forthcoming.

12.2. Part II

The method of feasible directions may be usefully employed as an aid in determining the best operating policy of a reservoir for electrical energy and agricultural production. Although the models considered are simple (with only up to36 variables in 26 constraints ~~unknowns~~) this approach provides very useful first approximations to the actual operating conditions. The models developed and solved were all deterministic. Further refinements towards a more realistic model should allow for uncertainty in the stream flows, storage of water in the reservoirs and drafts respectively. A dynamic programming formulation a three-dimensional model and which allows for uncertainty in stream flows has been presented and it is hoped that programmes capable of solving such problems (three-dimensional or more) will be developed soon.

The economical use of reservoir water in the generation of electrical energy is becoming an important consideration in any system with both hydro-electric and thermal units. Optimal operating policies obtained (Chapter 9) show clearly that non-linear programming methods, especially the sequential unconstrained optimization techniques (Logarithmic potential and the Fiacco-McCormick algorithm) provide very useful guide-lines with regard to the economics of such combined operations. The model considered comprises two variable-head hydro-units. A larger problem would probably require decomposition along the lines suggested by Zangwill (Chapter 7).

For electric transmission system design, the 'adaptive' approach followed in the formulation of a linear programming model results in substantial reductions in the number of constraints, thus making it possible to tackle relatively larger problems. The approach also enables the designer to assess the relative importance of the constraints. The resulting configuration could form a useful first step in the design of the actual system. Solutions by non-linear programming method are inadequate: in all cases the solutions are non-integral. The inability to give integer solutions is a serious limitation of the existing non-linear programming methods; and it is hoped that more research effort will be devoted to the development of algorithms that can give integer solutions.

Generation planning involves the investment of large amounts of money. Such vast expenditure warrants thorough analysis to evaluate the effect of the numerous factors bearing on the problem. Mathematical programming could prove an extremely useful tool to the planning engineer, provided that suitable methods for solving large problems of the type involved can be developed. Non-linear programming and dynamic programming formulations have been presented and possible solutions suggested.

12.3. Suggestions for Further Work

The subject matter of this thesis has been wide and varied. Consequently there are many areas which require further investigation. Some of these have been discussed in the main body of the thesis; for example:

(a) Application of geometric programming to the solution of power system problems whose objective functions are expressible as products of the design variables;

(b) establishment of an efficient algorithm for the decomposition of large linear programming problems by dynamic programming;

(c) development of a method of feasible directions that is capable of handling non-linear constraints;

(d) formulation of models for multi-purpose operation of water reservoirs in such a way that uncertainties in stream flow, reservoir storage and drafts, respectively, are allowed for.

(e) development of a dynamic algorithm to solve the model given in Chapter 8;

(f) Application of the decomposition method suggested by Zangwill for the solution of large hydro-thermal operation problems;

(g) further investigations into the capability of the existing unconstrained optimization techniques with regard to large systems involving several hundred variables and constraints.

In addition to the above suggestions, the author feels that further research work should be devoted to the task of establishing fundamental factors pertaining to the validity of a given (mathematical) model.

Furthermore, it is hoped that Zangwill's method of "Penalty Function" (Chapter 5) will soon be used for the solution of non-linear problems of the type considered in this thesis, and that a nonlinear version of the branch and bound algorithm (Chapter 4) will soon be developed and used to solve generation planning problems of the type discussed in Chapter 11.

Contributions of the the Thesis:

Major original contributions of this thesis are summarized below:

(1) A thorough and critical survey of mathematical programming techniques from both theoretical and computational points of view; also, a more unified presentation of the mathematical programming methods than is at present available.

(2) Clarification of the difficulties encountered in the formulation of a mathematical model.

(3) Clarification of the idea of the Systems Design Approach and its relevance to systems optimization in general.

(4) Application of a method of feasible directions to determine the best policy for a reservoir for electrical energy production and irrigation. Also, formulation of a three-dimensional model suitable for solution by a dynamic programming algorithm.

(5) Application of two sequential unconstrained minimization techniques to obtain optimal operating policies

for a variable-head hydrothermal electrical system.

(6) A general formulation of a transmission system design problem. Also, the introduction of an adaptive method of imposing constraints - in the formulation of a linear programming model of a transmission network design problem; this results in a sizeable reduction in the number of constraints.

(7) Formulation of nonlinear programming models of a transmission network design problem and solution of the resulting models by means of a sequential unconstrained optimization technique. Indication of the inability of existing nonlinear programming methods to give integer solutions.

(8) Suggestion of a possible way through which Dale's dynamic programming method for generation planning may be extended so as to take account of the shape of the load duration curve. Also formulation of nonlinear programming models suitable for solution by the existing sequential unconstrained optimization techniques once the latter have been modified to handle large problems of the type encountered in power system planning.

REFERENCES

1. Abadie, J, (ed.), "Nonlinear Programming," (North Holland Publishing Company 1967).

2. Balanski, M.L. "Integer Programming: Methods, Uses and Computation," Management Science Vol.12 (1965) pp. 253-313.

3. Balanski, M.L. and Gomory, R.E. "A Mutual Primal-Dual Simplex Method," Recent Advances in Mathematical Programming, Graves, R.L. and Wolfe, P. (ed.) McGraw-Hill (1963).

4. Balas, E. "An Additive Algorithm for Solving Linear Programs With Zero-One Variables," Operations Research, Vol, 13. No.4. (July-August, 1965) pp.517-546.

5. Beale, E.M.L., "Mathematical Programming in Practice," (Pitman, 1968).

6. Bellman, R.E. and Dreyfus, D.E., "Applied Dynamic Programming," (Princeton University Press, 1962).

7. Berge, C. and Ghouila-Houri, A, "Programming, Games & Transportation Networks," (John Wiley & Sons, 1965).

8. Butkovskii, A.G. "The Necessary and Sufficient Conditions for Optimality of Discrete Control Systems," Automation and Remote Control Vol. 24 pp.963-970 (1963).

9. Burstall, R.M. "Computer Design of Electricity Supply Networks by a Heuristic Method," The Computer Journal Vol.9 pp.263-274 (1966-67).

10. Broyden, C.G., "Quasi-Newton Methods and their Application to Function Minimization," Math. Comp., Vol. 21 (July 1967) p.368.

11. Carroll, C.W. "The Created Response Surface Technique for Optimizing Nonlinear Restrained Systems," Operations Research, Vol. 9 No.2 pp.169-84 (1961).

12. Crary, S.B., "Power System Stability," Vol.1 (John Wiley & Sons, 1950).

13. Cypser, R.J., "Computer Search for Economical Operation of a Hydrothermal Electric System," A.I.E.E. Transactions Vol.73 Pt.III-B. (Oct. 1954) pp.1260-67.

14. Dale, K.M. "Dynamic Programming Approach to the Selection and Timing of Generation-Plant Additions," Proc. I.E.E. Vol.113 (1966), pp.803-811.

15. Dantzig, G.B., "Linear Programming and Extensions," (Princeton University Press, 1963).

16. Davidon, W.C., "Variable Metric Method for Minimization," Argonne National Laboratory Report ANL-5990 Rev., (Nov. 1959).

17. Davis, R.H. and Roberts, P.D., "Method of Conjugate Gradients Applied to Self-Adaptive Digital Control Systems," Proc. I.E.E. Vol.115, No.4. (April 1968), pp.562-571.

18. Dennis, J.B. "Mathematic Programming and Electrical Networks." (M.I.T. Press, 1959).

19. Doig, A.G. and Land, A.H., "An Automatic Method of Solving Descrete Programming Problems," Econometrica, Vol.28 (1960) pp.497-520.

20. Dorn, W.S., "Nonlinear Programming: A Survey," Management Science, Vol.9, No.2 (January, 1963), pp.171-208.

21. Dubovitsky, A.Ya, and Milyutin, A.A. "Some Optimal Problems for Linear Systems," Automation and Remote Control, Vol.24, No.12, (1963).

22. Eckman, D.P. (ed.) "Systems:Research and Design", (John Wiley & Sons, 1961).

23. Falb, P.L. and Athans, M., "Optimal Control," (McGraw-Hill, 1966).

24. Fiacco, A.V. and McCormick, G.P., "Programming Under Nonlinear Constraints by Unconstrained Minimization: A Primal-Dual Method," Research Analysis Corp. RAC-TP-96 (Sept. 1963).

25. Fiacco, A.V. and McCormick, G.P., "The Sequential Unconstrained Minimization Technique for Convex Programming with Equality Constraints," Research Analysis Corp; RAC-TP-155, (1965).

26. Fletcher, R., "Review of Unconstrained Optimization" Proc. Joint Conference on Optimization, University of Keele, (March, 1968).

27. Fletcher, R., "Function Minimization Without Evaluating Derivatives - A Review," Computer Journal Vol.8 (1965) pp.33.

28. Fletcher, R. and Powell, M.J.D., "A Rapidly Convergent Descent Method for Minimization," Computer Journal Vol.6, (1963), pp.163-168.

29. Fletcher, R. and Reeves, C.M., "Functional Minimization by Conjugate Gradients," Computer Journal, Vol.7 (1964).

30. Frisch, R., "The Logarithmic Potential Method for Solving Linear Programming Problems," Memorandum, University Institute of Economics, Oslo, (May, 1955).

31. Fromovitz, S, "Nonlinear Programming with Randomization", Management Science, Vol.11, No.9, (July, 1965).

32. Gass, S.I. and Saaty, T.L. "Parametric Objective Function, II, Generalization," Operations Research Vol.3 (1955), pp.395-401.

33. Geoffrion, A.M., "Integer Programming by Implicit Enumeration and Bala's Method," The RAND Corp., RM-4783-PR., (February 1966).

34. Glover, F., "A Multiphase-Dual Algorithm for Integer Linear Programming," Technical Report No.3, Contract NONR-225 (89), Stanford Univ. (August, 1966).

35. Goldfarb, D. and Lapidus, L., "A Conjugate Gradient Method for Nonlinear Programming," A.I.Ch.Eng. 61st National Meeting, Houston, (February 1967).

36. Gomory, R.E., "On the Relation between Integer and Non-Integer Solutions to Linear Programs," Proc. Nat. Acad. Sci. Vol.53 (1965), pp.260-265.

37. Griffith, R.E. and Stewart, R.A., "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems," Management Science, Vol.7 (1961), pp.379-392

38. Graves, G.W. "A Complete Constructive Algorithm for the General Mixed Linear Programming Problem," Naval Research Logistics Quarterly, Vol.12 No.1 (1965) pp.1-34.

39. Graves, R.L. and Wolfe, P., (ed.) "Recent Advances in Mathematical Programming," (McGraw-Hill, 1963).

40. Haldi, J. and Isaacson, L.M., "An Integer Programming Code," Std. Oil. Company of California, San Francisco, (1965).

41. Hadley, G. "Linear Programming," (Addison-Wesley, 1965).

42. Haldi, J. and Isaacson, L.M., "Linear Integer Programming 1," I.B.M. Share General Program (3335), (Feb. 1965).

43. John, F., "Extremum Problems with Inequalities as Subsidiary Conditions," in Studies and Essarys, Courant Anniversary, (Interscience, N.Y. 1948).

44. Karlin, S. "Mathematical Methods in Game Theory, Programming and Economics," (Addison-Wesley, 1959).

45. Kelly, H.J. and Myers, G.E. "Conjugate Direction Methods for Parameter Optimization," 18th Congress Int. Astronautical Federation, Belgrade, Yugoslavia, (September, 1967).

46. Kelly, Jr., J.E., "The Cutting Plane Method for Solving Convex Programs," Journal of Soc. Indust. Appl. Math. Vol.8 (1960), pp.703-712.

47. Knight, U.G., "The Logical Design of Electrical Networks Using Linear Programming Methods," Proc. I.E.E. Vol.107, Part A. (April, 1960).

48. Knight, U.G., "Optimization Methods in Power System Design and Operation ," Proc. Conference on Digital Computation for Electrical Power Systems, Queen Mary College, London, (September, 1963).

49. Knight, U.G., Ph.D. Thesis, University of London, (1967).

50. Kron, G., "Diakoptics," (Macdonald, 1963).

51. Kuhn, H.W. and Tucker, A.W., "Nonlinear Programming" in Proc. of the Second Berkeley Symposium of Mathematical Statistics and Probability, J. Neyman (ed.) University of California (1951).

52. Lee, R.C.K., "Optimal Estimation, Identification and Control," (M.I.T. Press, 1964).

53. Leitmann, G. (ed.) "Optimization Techniques," (Academic Press, 1962).

54. Little, J.C., et. al., "An Algorithm for the Travelling Salesman Problem," Operations Reserach, Vol. 11 (1963), pp.972-989.

55. Lootsma, F.A., "Logarithmic Programming: A Method of Solving Nonlinear Programming Problems," Phillips Res. Repts. Vol.22, pp.329-344.

56. Maass, A. et. al., "The Design of Water Resource Systems," (Harvard Univ. Press, 1962).

57. Mangasarian, O.L., and Fromovitz, S. "A Maximum Principle in Mathematical Programming" in Mathematical Theory of Control, Balakrishnan, A.V. and Neustadt, L.W. (eds.)

58. Miller, C.E., "The Simplex Method for Local Separable Programming," p.89. In Recent Advances in Mathematical Programming, Graves, R.L. and Wolfe, P. (ed) (McGraw-Hill, 1963).

59. Mitter, S.J. and Liacco, T.E. Dy, "Multilevel Approach to the Control of Interconnected Power Systems," Systems Research Center, Case Institute of Technology, Cleveland, Ohio, U.S.A.

60. Murtagh, B.A. and Sargent, R.W.H. "A Constrained Minimization with Quadratic Convergence," Proc. Joint Conference on Optimization, Univ. Of Keele, (March, 1968).

61. Nenhauser, G.L. "Decomposition of Linear Programs by Dynamic Programming," Naval Research Logistics Quarterly, Vol. 10 (1963).

62. Pearson, J.D. "Duality and a Decomposition Technique," J. SIAM Control, Vol.4 No.1 (1966), pp.164-172.

63. Pearson, J.D. and McCormick, G.D., "Variable Metric Method, Penalty Functions and Unconstrained Optimization," Proc. Joint Conference on Optimization, Univ. of Keele, (March, 1968).

64. Pervozvanskiy, A.A. "Relationship Between the Basic Theorems of Mathematical Programming and the Maximum Principle," Engineering Cybernetics, No.1 (Jan-Feb 1967), pp.6-11.

65. Pontryagin, L.S. et al., "The Mathematical Theory of Optimal Processes," (Interscience, 1962).

66. Propoi, A, I., "The Maximum Principle for Discrete Control Systems," Avtomatika i Telemekhanika, Vol. 26, No.7 (1965).

67. Roberts, S.M., "Dynamic Programming in Chemical Engineering and Process Control," (Academic Press 1964).

68. Rosen, J.B., "The Gradient Projection Method for Nonlinear Programming. Part I - Linear Constraints," J. Soc. Indust. Appl. Math. No.8 (1960), pp.181-217.

69. Rozonoer, L.I. "L.S. Pontryagin's Maximum Principle in the Theory of Optimum Systems - I, II & III" in Optimal and Self-Optimizing Control, Oldenburger, R (ed.) (M.I.T. Press, 1966).

70. Sasson, A.M. "Nonlinear Programming Solutions for the Load Flow, Minimum Loss and Economic Dispatching Problems," Proc. I.E.E.E. Summer Power Conference, Chicago (June, 1968) No.68 TP 673 PWR; also to appear in Trans. I.E.E.E.: Power Systems & Apparatus.

71. Shapiro, J.F., "Dynamic Programming Algorithms for the Integer Programming Problem - I : The Integer Programming Problem Viewed as a Knapsack Type Problem," Operations Research, Vol. 16 No.1 (Jan-Feb. 1968), pp.103-121.

72. Shapiro, J.F. and Wagner, H.M., "A Finite Renewal Algorithm for the Knapsack and Turnpike Models", Operations Research, Vol. 15 (1967), pp.319-341.

73. Singleton, W.T. and Whitfield, E.D. (eds.) "The Human Operator in Complex Systems," (Taylor and Francis, 1967).

74. Stewart III, G.W. "A Modification of Davidon's Minimization Method to Accept Difference Approximations of Derivatives," J. Assoc. Computing Machinery Vol.14 No.1. (Jan, 1967), pp.72-83.

75. Thomas, Jr. H.A. and Revelle, R. "On the Efficient Use of High Aswan Dam for Hydropower and Irrigation," Management Science, Vol.12, No.8 (April 1966), pp. B-296 - B-311.

76. Watters, L.J. "Reduction of Integer Nonlinear Programming Problems to Zero-One Linear Programming Problems," Rand Corp. P-3509 (Dec. 1966).

77. Wilde, D.J. and Beightler, C.S. "Foundations of Optimization," (Prentice-Hall, 1967).

78. Wolfe, P, "Nonlinear Programming Methods," in Recent Advances in Mathematical Programming, Graves, Graves, R.L. and Wolfe, P. (eds.) (McGraw-Hill, 1963). Also in Nonlinear Programming, Abadie, J. (ed.) (North-Holland Pub. Co. 1967).

79. Zadeh, L.A. and Desoer, C.A. "Linear Systems Theory: The State Space Approach," (McGraw-Hill, 1963).

80. Zangwill, W.I. "Nonlinear Programming via Penalty Functions," Management Science, Vol.13 (1967), pp.344-358.

81. Zangwill, W.I. "Minimizing a Function without Calculating Derivatives," Computer Journal, Vol. 10 (1967) pp.293-296.

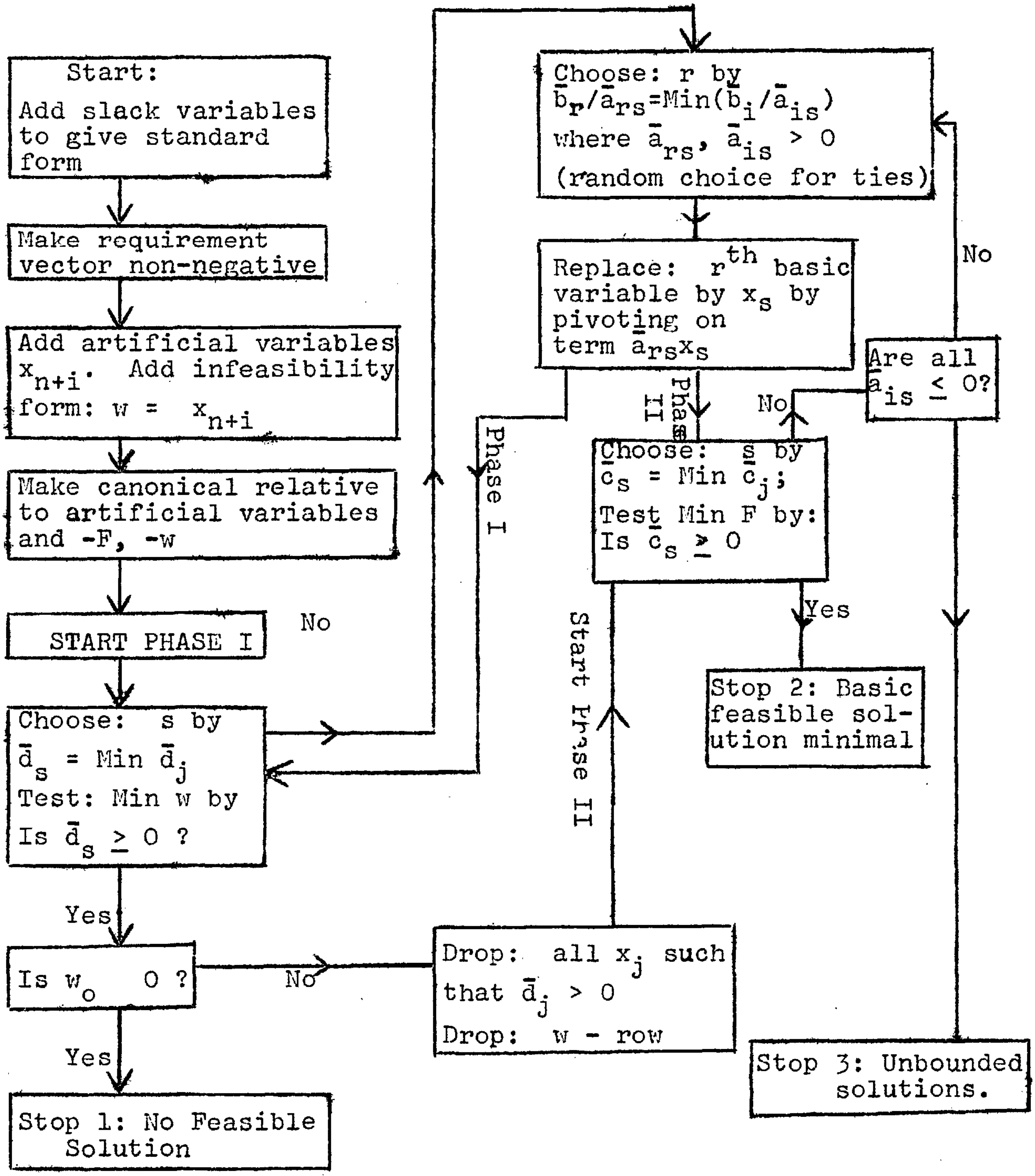
82. Zangwill, W.I. "A Decomposible Nonlinear Programming Approach," Operations Research, Vol.15 (1967) pp.1068-1087.

83. Zenner. C and Duffin,J. "Optimization of Engineering Problems," Westinghouse, Eng. (Sept. 1964), pp. 154-160.

84. Zoutendijk, G. "Nonlinear Programming: A Numerical Survey," J. SIAM Control, Vol.4, No.1 (1966), pp.194-210.

85. Zoutendijk, G. "Methods of Feasible Directions," (Elsevier, Amsterdam, 1960).

APPENDIX A.1.


Start:
Add slack variables to give standard form
Make requirement vector non-negative
Add artificial variables x_{n+i}. Add infeasibility form: $w = x_{n+i}$
Make canonical relative to artificial variables and $-F$, $-w$
START PHASE I
No
Choose: s by $\bar{d}_s = \text{Min } \bar{d}_j$
Test: Min w by
Is $\bar{d}_s \geq 0$?
Yes
Is $w_o \quad 0$?
No
Yes
Stop 1: No Feasible Solution
Drop: all x_j such that $\bar{d}_j > 0$
Drop: w - row
Start Phase II
Phase I
Choose: r by $\bar{b}_r/\bar{a}_{rs} = \text{Min}(\bar{b}_i/\bar{a}_{is})$ where $\bar{a}_{rs}$, $\bar{a}_{is} > 0$ (random choice for ties)
Replace: r^{th} basic variable by x_s by pivoting on term $\bar{a}_{rs}x_s$
Phase II
Choose: s by $\bar{c}_s = \text{Min } \bar{c}_j$;
Test Min F by:
Is $\bar{c}_s \geq 0$
Yes
Stop 2: Basic feasible solution minimal
No
Are all $\bar{a}_{is} \leq 0$?
No
Stop 3: Unbounded solutions.

Flow Diagram of the Simplex method

APPENDIX A.2.

A METHOD OF FEASIBLE DIRECTIONS:

Methods of feasible directions (in conjunction with other methods of non-linear programming) have been discussed in Chapter 5. Detailed steps of the version used to solve the problem discussed in Chapter 8 are given here. Only linear constraints are considered.

Consider the problem:

$$\text{Minimize:a convex function, } F(\bar{x}) \qquad \text{(A2.1)}$$

$$\text{subject to} \quad \Sigma_{j=1}^{n} a_{ij}x_j \geq b_i \quad i = 1,2,\ldots m \qquad \text{(A2.2)}$$

$$x_j \geq 0 \qquad j = 1,\ldots,n \qquad \text{(A2.3)}$$

The computational rules are as follows:

(i) An initial feasible solution $\bar{x}_j^o$ is chosen: i.e. a point satisfying (A2.2) and (A2.3).

(ii) A step-length t (t>0) is chosen such that

$$\bar{x}_j = \bar{x}_j^o + t\bar{s}_j \qquad \text{(A2.4)}$$

satisfies (A2.2) and (A2.3) and such that

$$\sum_{j=1}^{n} a_{ij}s_j \geq 0 \quad i = 1,2,\ldots k \qquad \text{(A2.5)}$$

$$s_j \geq 0 \quad j = 1,2,\ldots m \qquad \text{(A2.6)}$$

$$-1 \leq s_j \leq 1 \quad j = m+1,\ldots,n \qquad \text{(A2.7)}$$

(iii) In order to obtain the largest decrease in $F(\bar{x}_j)$ we

$$\text{Minimize} \quad \sum_{j=1}^{n} \frac{\partial F}{\partial x_j}(x_1^o,\ldots x_n^o)s_j \qquad \text{(A2.8)}$$

subject to (A2.5)-(A2.7). This is a linear programming problem; and the solution s_j^o is a feasible direction. The point $\bar{x}_j$ given by A2.4 yields a smaller value of $F(\bar{x}_j)$, provided that t is sufficiently small.

If the solution $s_j^o \equiv 0$, then x_j^o is the solution to the problem (i.e. minimizes (A2.1) over the given constraint set).

If however, $s_j^o \not\equiv 0$, the parameter t is chosen in the following way:

(a)

$$\hat{t} = \mathrm{Min}\left\{ \begin{array}{ll} (b_i - \sum_{j=1}^{n} a_{ij} x_j^o) / \sum_{j=1}^{n} a_{ij} s_j^o & \text{for } \sum_{j=1}^{n} a_{ij} s_j^o < 0 \\ - x_j^o / s_j^o & \text{for } s_j^o < 0 \end{array} \right. \qquad (A2.9)$$

If $t \leq \hat{t}$ then $\bar{x}_j$ in (A2.4) satisfies the original constraints.

(b) Consider the following function of a single real variable, t,

$$\psi(t) = \sum_{j=1}^{n} s_j^o \left. \frac{\partial F}{\partial x_j} \right|_{x_j^o + t s_j^o} = 0 \qquad (A2.10)$$

Since $F(x_1, x_2, \ldots, x_n)$ is convex, (A2.10) has at most one real solution. Suppose such a solution exists and is denoted by t*. If $t \leq t^*$ then $F(\bar{x}_j)$ will decrease for increasing t.

(c) the optimum choice of t is then

$$t_m = \text{Min}\ (\hat{t}, t^*)$$

and the new feasible solution is given by

$$\bar{x}_j^{(1)} = \bar{x}_j^o + t_m s_j^o \qquad (A2.11)$$

The process is iterated until either:

(i) $s_j^o \equiv 0$ or

(ii) the decrease in the function is sufficiently small.

APPENDIX A.3.

(a) The Fiacco-McCormick Method:

Given a mathematical programming problem:

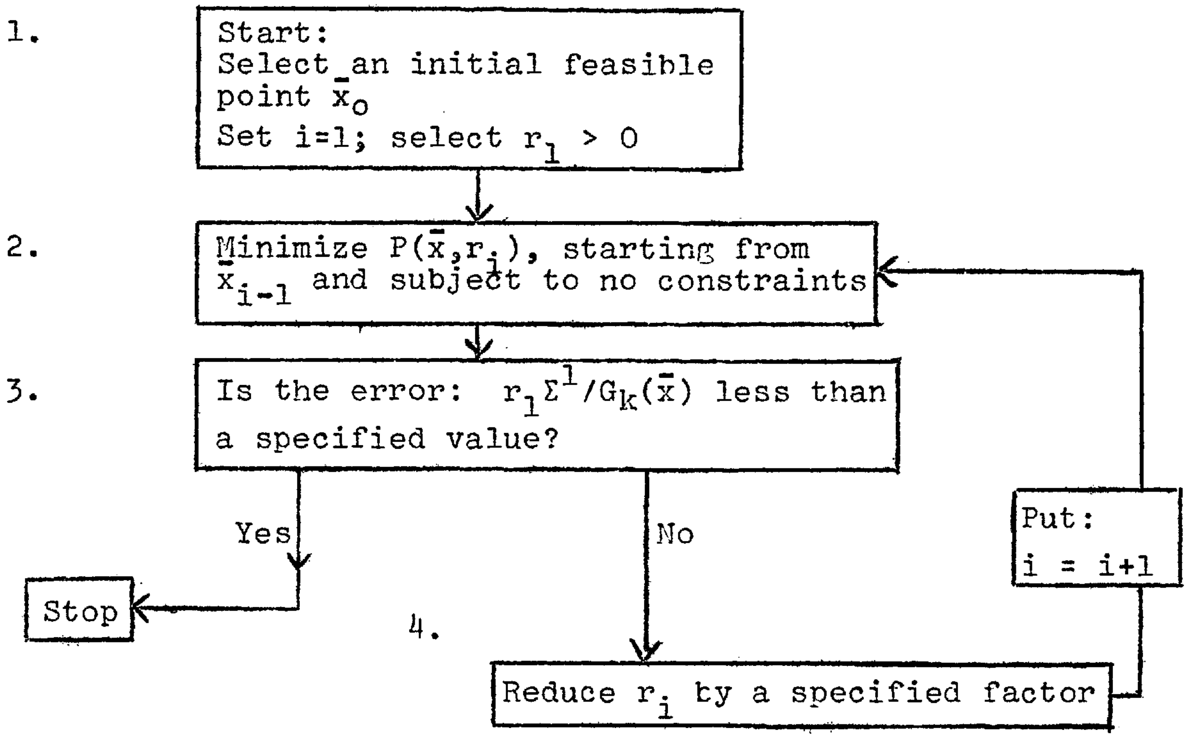
$$\text{Minimize} \quad F(\bar{x}) \tag{A3.1}$$

$$\text{subject to} \quad G_i(\bar{x}) \qquad i = 1,\ldots,m \tag{A3.2}$$

the problem is transformed into an unconstrained one:

$$\text{Minimize} \quad P(\bar{x},r_i) + r \sum_{i=1}^{m} {}^{1}/G_i(\bar{x}) \tag{A3.3}$$

subject to no constraints. A3.3 is then solved as indicated in the flow-diagram below (Fig. A3.1):



1.
Start:
Select an initial feasible point $\bar{x}_0$
Set i=1; select $r_1 > 0$
2.
Minimize $P(\bar{x},r_i)$, starting from $\bar{x}_{i-1}$ and subject to no constraints
3.
Is the error: $r_1\Sigma^1/G_k(\bar{x})$ less than a specified value?
Yes
No
Stop
4.
Reduce r_i by a specified factor
Put:
i = i+1

Fig. A3.1. The Fiacco-McCormick Method.

The actual minimization (Step 2) is carried out by a suitable unconstrained optimization method: e.g. the Fletcher-Powell algorithm given in Appendix A4.

(b) Lootsma's Logarithmic Potential Method:

In this case problem A3.1 and A3.2 is transformed into:

$$\text{Minimize} \quad P(\bar{x},r_i) \quad = \quad F(\bar{x}) - r \sum_{i=1}^{m} \text{In. } G_i(x) \quad (A3.4)$$

The computational steps are similar to those of the Fiacco-McCormick method. The only essential difference is that the initial value of r is estimated by

$$r_i \quad = \quad {}^{\delta_e}/m \qquad i = 1 \qquad (A3.5)$$

where δ_e is the estimate of the error term:

$$\delta_e \quad \doteq \quad r \,\Sigma \text{ In } \; G_i(\bar{x}) \qquad (A3.6)$$

and m is the number of constraints (See Chapter 5 for a more detailed discussion).

APPENDIX A.4.

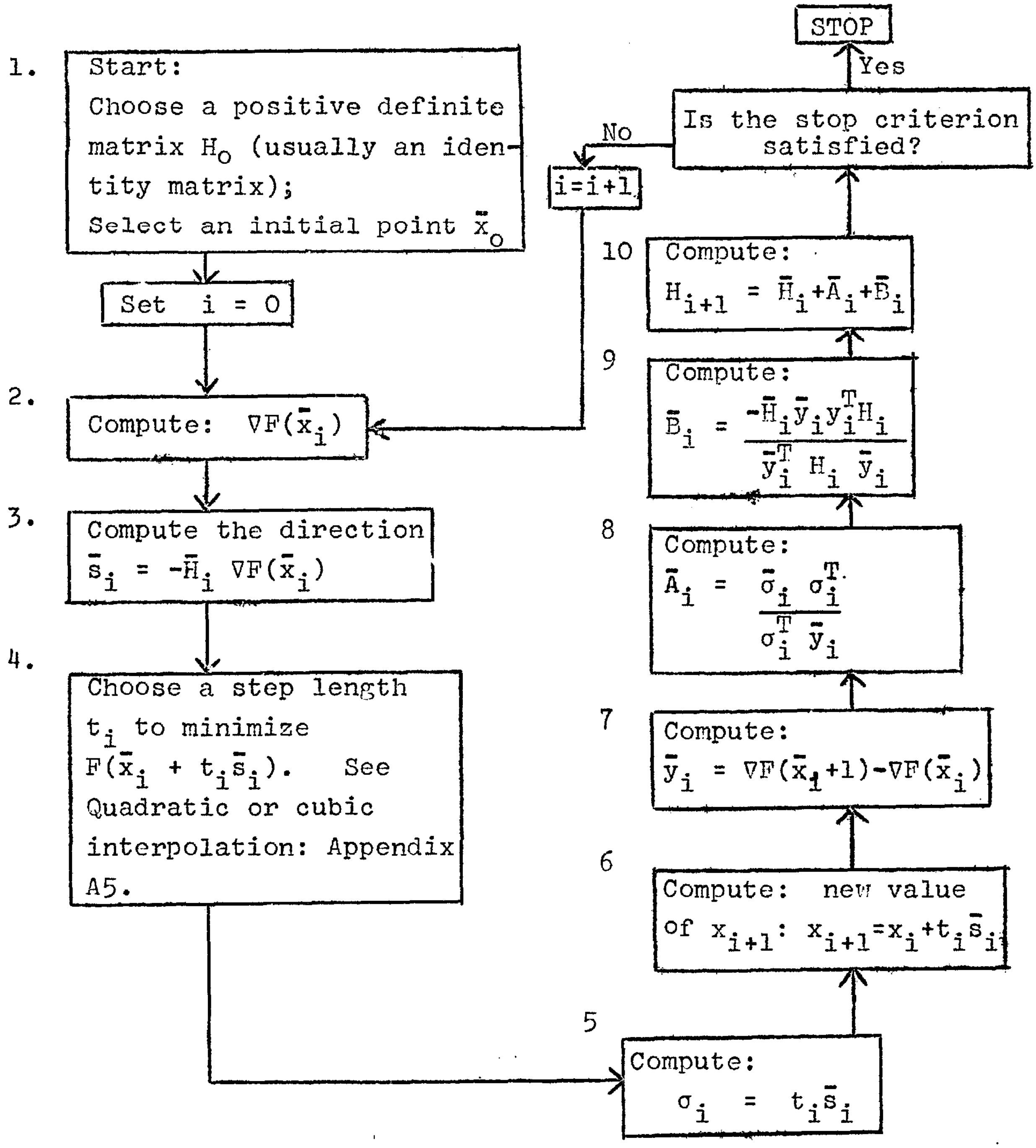

1.
Start:
Choose a positive definite matrix H_o (usually an identity matrix);
Select an initial point $\bar{x}_o$
Set i = 0
2.
Compute: $\nabla F(\bar{x}_i)$
3.
Compute the direction $\bar{s}_i = -\bar{H}_i \nabla F(\bar{x}_i)$
4.
Choose a step length t_i to minimize $F(\bar{x}_i + t_i\bar{s}_i)$. See Quadratic or cubic interpolation: Appendix A5.
5
Compute: $\sigma_i = t_i\bar{s}_i$
6
Compute: new value of x_{i+1}: $x_{i+1}=x_i+t_i\bar{s}_i$
7
Compute: $\bar{y}_i = \nabla F(\bar{x}_i+1)-\nabla F(\bar{x}_i)$
8
Compute: $\bar{A}_i = \frac{\bar{\sigma}_i \sigma_i^T}{\sigma_i^T \bar{y}_i}$
9
Compute: $\bar{B}_i = \frac{-\bar{H}_i\bar{y}_i y_i^T H_i}{\bar{y}_i^T H_i \bar{y}_i}$
10
Compute: $H_{i+1} = \bar{H}_i+\bar{A}_i+\bar{B}_i$
Is the stop criterion satisfied?
Yes
STOP
No
i=i+1

Fig. A4. The Fletcher-Powell Unconstrained Optimization Method

APPENDIX A.5.

CUBIC AND QUADRATIC INTERPOLATIONS

As we have seen in Chapter 5, the direct optimization methods proceed - from a given arbitrary point - by choosing a direction for the next step, and the step-length.

The step-length t is usually chosen as the value of $t > 0$ which minimizes the function

$$g(t) = F(\bar{x}_i + t\bar{s}_i) \qquad (A5.1)$$

A minimum value, t_m, which minimizes A5.1 is obtained by the cubic interpolation procedure outlined in Fig. A5.1. If derivatives of $F(\bar{x}_i + t\bar{s}_i)$ are not available or are difficult to compute, the quadratic interpolation procedure can be used to determine the value of t which minimizes A5.1.

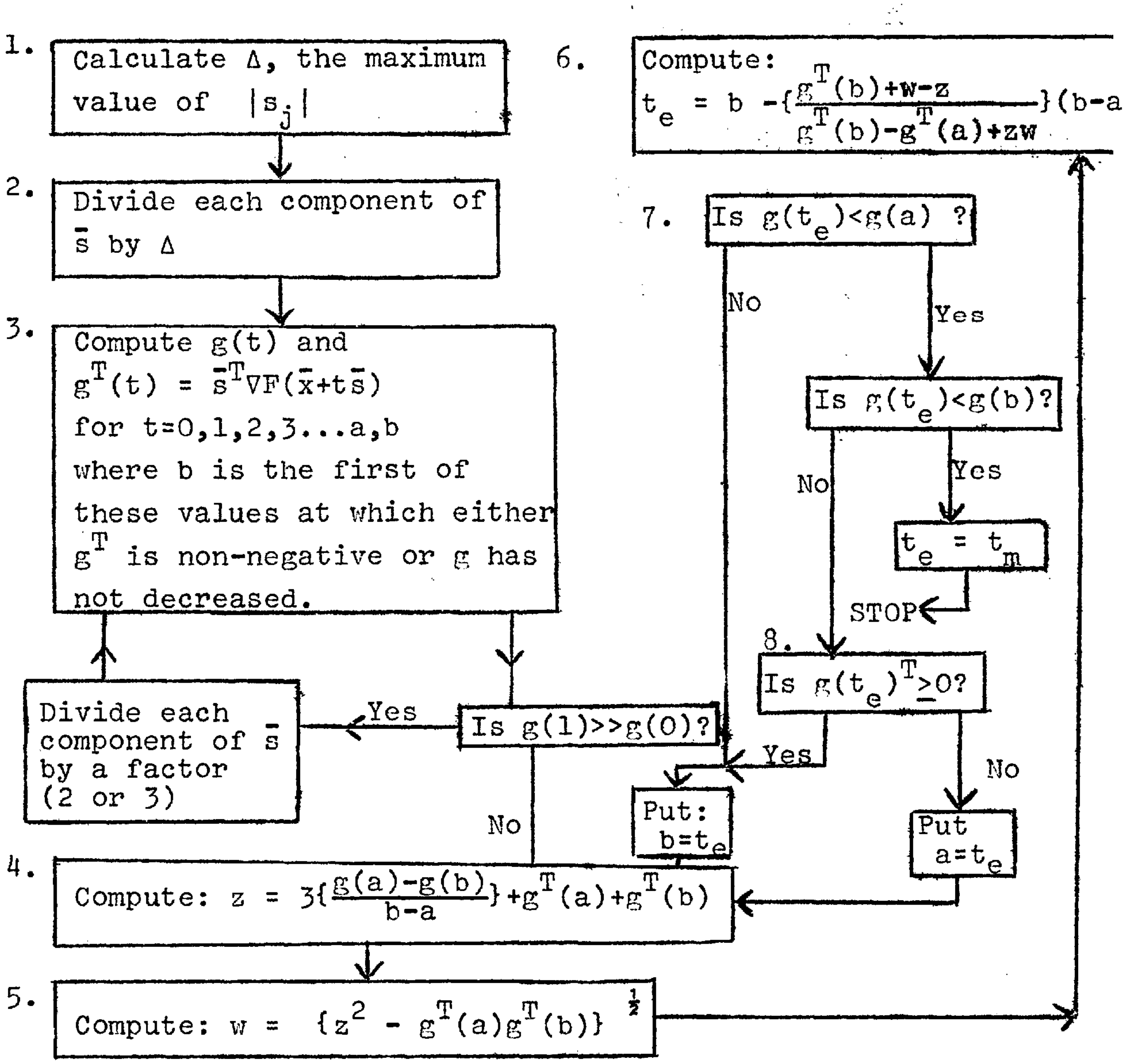

1. Calculate Δ, the maximum value of $|s_j|$
2. Divide each component of $\bar{s}$ by Δ
3. Compute g(t) and $g^T(t) = \bar{s}^T \nabla F(\bar{x}+t\bar{s})$ for t=0,1,2,3...a,b where b is the first of these values at which either g^T is non-negative or g has not decreased.
Is g(1)>>g(0)?
Yes
Divide each component of $\bar{s}$ by a factor (2 or 3)
No
4. Compute: $z = 3\{\frac{g(a)-g(b)}{b-a}\}+g^T(a)+g^T(b)$
5. Compute: $w = \{z^2 - g^T(a)g^T(b)\}^{\frac{1}{2}}$
6. Compute: $t_e = b -\{\frac{g^T(b)+w-z}{g^T(b)-g^T(a)+zw}\}(b-a$
7. Is $g(t_e)<g(a)$?
No
Yes
Is $g(t_e)<g(b)$?
No
Yes
$t_e = t_m$
STOP
8. Is $g(t_e)^T \geq 0$?
Yes
No
Put: $b=t_e$
Put $a=t_e$

Fig. A5.1. Cubic Interpolation

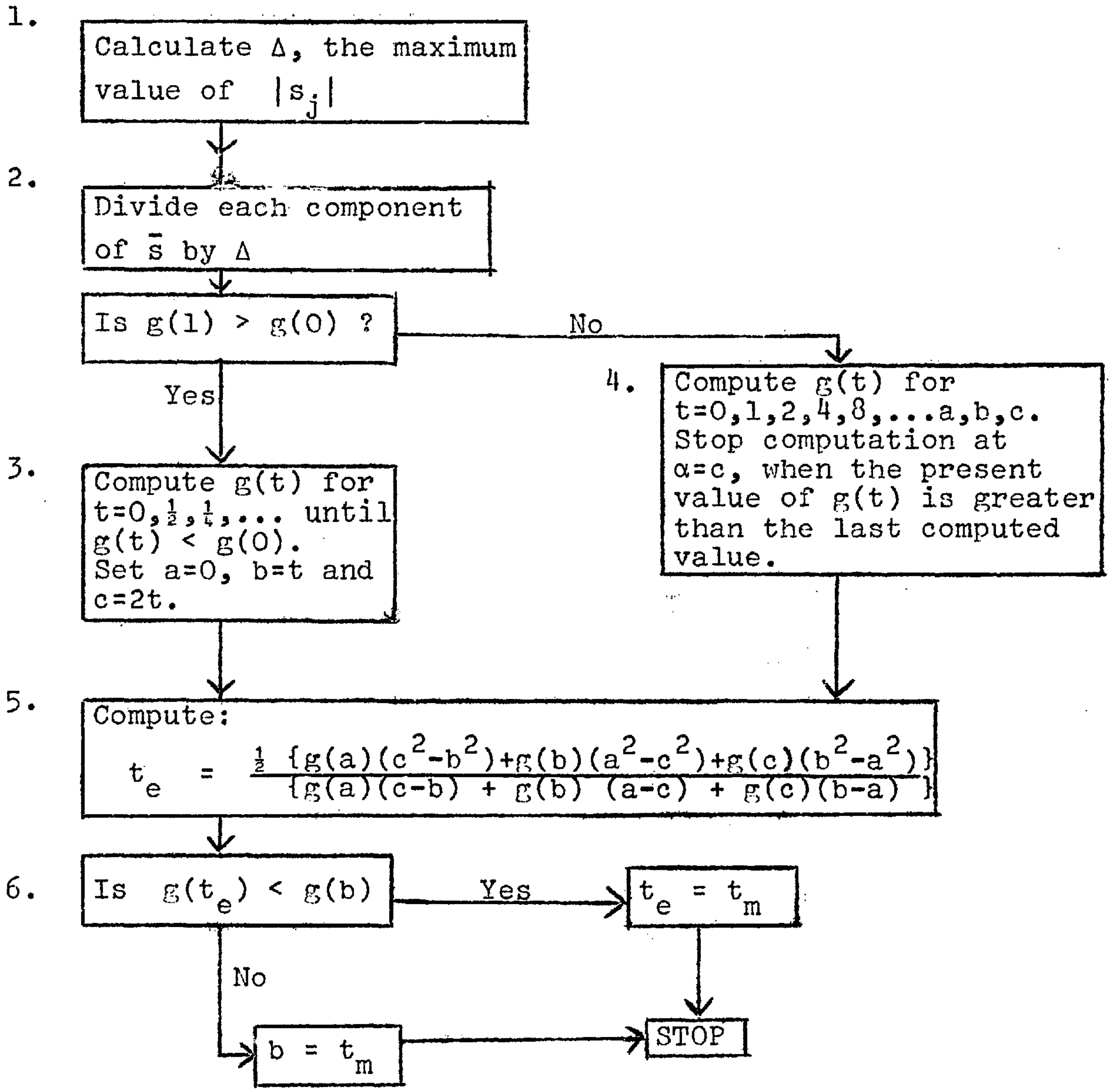

1.
Calculate Δ, the maximum value of $|s_j|$
2.
Divide each component of $\bar{s}$ by Δ
Is g(1) > g(0) ?
No
Yes
3.
Compute g(t) for $t=0,\frac{1}{2},\frac{1}{4},\ldots$ until g(t) < g(0). Set a=0, b=t and c=2t.
4.
Compute g(t) for t=0,1,2,4,8,...a,b,c. Stop computation at α=c, when the present value of g(t) is greater than the last computed value.
5.
Compute:
$t_e = \frac{\frac{1}{2}\{g(a)(c^2-b^2)+g(b)(a^2-c^2)+g(c)(b^2-a^2)\}}{\{g(a)(c-b)+g(b)(a-c)+g(c)(b-a)\}}$
6.
Is $g(t_e) < g(b)$
Yes
$t_e = t_m$
No
$b = t_m$
STOP

Fig. A5.2. Quadratic Interpolation

APPENDIX A.6.

A METHOD OF PREDICTING THE RELIABILITY OF CONTINUOUSLY OPERATING SYSTEMS **

In Chapter 10 we discussed how reliability constraints can be generated for use in the formulation of transmission system mathematical programming model. Details of the method of generating the desired reliability constraints are given in this section.

The main featuresof the method are as follows. The state of a complex system (designed for continuous operation) is defined by identifying the sub-systems which are functioning and those which are undergoing repair. The system is described as UP whenever it is in one of the arbitrarily selected set of system states.

It is assumed that maintainance facilities are always adequate. The formulae derived give the mean durations of system UP-times and DOWN-times in terms of the corresponding quantities for the sub-systems.

A6.1. A General Statement of the Problem

Consider a system S composed of n sub-systems $S_1, \dots, S_n$. At any given time each of the sub-systems is either UP (in working order) or DOWN (for repair). The

** Based on the paper by M. Plotkin and S. Einhorn. I.E.E.E. Journal on Reliability, March 1965, pp.15-22.

system Ṡ is UP or DOWN depending on whether the set of sub-systems does or does not at the given instant include one of certain prescribed combinations of the sub-systems.

The mean duration U_j of UP-time and the mean duration D_j of DOWN-time for the sub-system S_j are known j=1,...,n. The problem is to compute the mean duration U UP-times and the mean duration D DOWN-times for system S.

In our example (Chapter 10) the sub-system, S_j, refers to single transmission line entering a given sub-station; and S is represented by the total number of lines that would be required to maintain adequate security of supply.

A6.2. Definitions

The following notations are used in the discussions:

$$M_{ik} = \begin{cases} U_k & \text{if sub-system } S_k \text{ is UP in system state i} \\ D_k & \text{if sub-system } S_k \text{ is DOWN in system state i} \end{cases}$$

FS = the set of all pairs of subscripts (i,j) such that the system is UP in state i but the failure of a single sub-system, S_ℓ put the system into state j, which is a DOWN state.

$\ell(i,j)$ = the subscript of the sub-system whose failure transforms the system from state i to state j.

g = the set of all subscripts identifying system UP states.

U = mean duration of system UP-time

D = mean duration of system DOWN-time.

A6.3. Transition Rates

The passage of one system state to another is called transition. If the transition from system state i to system state j requires a failure in sub-system $S_\ell(i,j)$, then

$$M_{i\ell}(i,j) = U_\ell(i,j) \qquad (A6.1)$$

$$M_{j\ell}(i,j) = D_\ell(i,j) \qquad (A6.2)$$

$$M_{ik} = M_{jk}, \quad k \neq \ell \qquad (A6.3)$$

The above equations also express what happens in the transition from system state j to system state i. The system states i and j are, in fact, adjacent, differing only in the condition of a single sub-system S_ℓ.

The sum

$$CR_\ell = D_\ell + U_\ell \qquad (A6.4)$$

is the mean duration of a cycle, consisting of a DOWN-time followd by an UP-time for sub-system S_ℓ. There are, on the average

$$1/CR_\ell = 1/(D_\ell + U_\ell) \qquad (A6.5)$$

cycles in sub-system S_ℓ per unit time. If we assume that the failure and repair in sub-system S_ℓ are independent of the events in the other sub-systems, then the expected number of transitions per unit time from system state i to system state j is

$$N_{ij} = \frac{1}{D_\ell(i,j)+U_\ell(i,j)} \left\{\text{Probability that remaining n-1 sub-systems are each in the state required by system state i.}\right\} \qquad (A6.6)$$

i.e.

$$N_{ij} \quad = \quad \frac{1}{D_{\ell}(i,j)+U_{\ell}(i,j)} \times \prod_{k\neq\ell} \left\{\frac{M_{ik}}{D_k+U_k}\right\} \qquad (A6.7)$$

where use has been made of the fact that

$$U_k/(D_k+U_k) \quad \text{and} \quad D_k/(D_k+U_k)$$

are respectively the probabilities that sub-system S_k is UP and DOWN. The probability that sub-system S_k is in a state required by sub-system state i is therefore

$$M_{ik}/(D_k+U_k)$$

Equation (A6.7) may be written

$$N_{ij} \quad = \quad \frac{1}{M_{i\ell}(i,j)} \prod_{k=1}^{n} \left\{\frac{M_{ik}}{D_k+U_k}\right\}$$

$$\equiv \quad \frac{1}{U_{\ell}(i,j)} \prod_{k=1}^{n} \left\{\frac{M_{ik}}{D_k+ U_k}\right\} \qquad (A6.8)$$

A6.4. Mean UP-time Duration:

The mean 'system' UP-time duration is computed as follows. Equation (A6.8) is evaluated for all transitions from i (system UP) to j (system DOWN). The sum

$$\sum_{(i,j)\in FS} N_{ij} \qquad (A6.9)$$

is the expected number of system UP to system DOWN transitions per unit time and its reciprocal

$$\left[\sum_{(i,j)\in FS} N_{ij}\right]^{-1} \qquad (A6.10)$$

is the mean time for the system cycle, consisting of one system DOWN-period followed by a system UP-period.

$$U + D = \left[\sum_{(i,j)\in FS} N_{ij} \right]^{-1} \qquad \text{(A6.11)}$$

From (A6.7) and (A6.8) we note that the probability that the system is in state i at any given instant is

$$P(S_i) = \prod_{k=1}^{n} \left\{ \frac{M_{ik}}{D_k + U_k} \right\} \qquad \text{(A6.12)}$$

Hence the probability that the system is in an UP-condition is

$$\frac{U}{U+D} = \sum_{i\in g} \prod_{k=1}^{n} \left\{ \frac{M_{ik}}{D_k + U_k} \right\} \qquad \text{(A6.13)}$$

Combining (A6.11) and (A6.13) gives

$$U = \left[\sum_{(i,j)\in FS} N_{ij} \right]^{-1} x \sum_{i\in g} \prod_{k=1}^{n} \left\{ \frac{M_{ik}}{D_k + U_k} \right\} \qquad \text{(A6.14)}$$

$$= \left\{ \sum_{(i,j)\in FS} \frac{\prod_{k=1}^{n} M_{ik}}{U_{\ell}(i,j)} \right\}^{-1} \sum_{i\in g} \prod_{k=1}^{n} M_{ik} \qquad \text{(A6.16)}$$

A6.5. Mean DOWN-time Duration

This is obtained from equations (A6.13) and (A6.15) and is denoted by

$$D = \left(\left[\sum_{i\in g} \prod_{k=1}^{n} \left\{ \frac{M_{ik}}{D_k + U_k} \right\} \right]^{-1} -1 \right) \qquad \text{(A6.16)}$$

APPENDIX A.7

PRINCIPAL STEPS OF THE SUGGESTED DYNAMIC PROGRAMMING.

Procedure

In Chapter 11, a possible way of applying the principles of dynamic programming was suggested. A flow diagram of the main steps of the algorithm are included here Fig. A.7. The diagram is essentially similar to that of Dale.[*]

[*] Ref. 14 in the thesis.

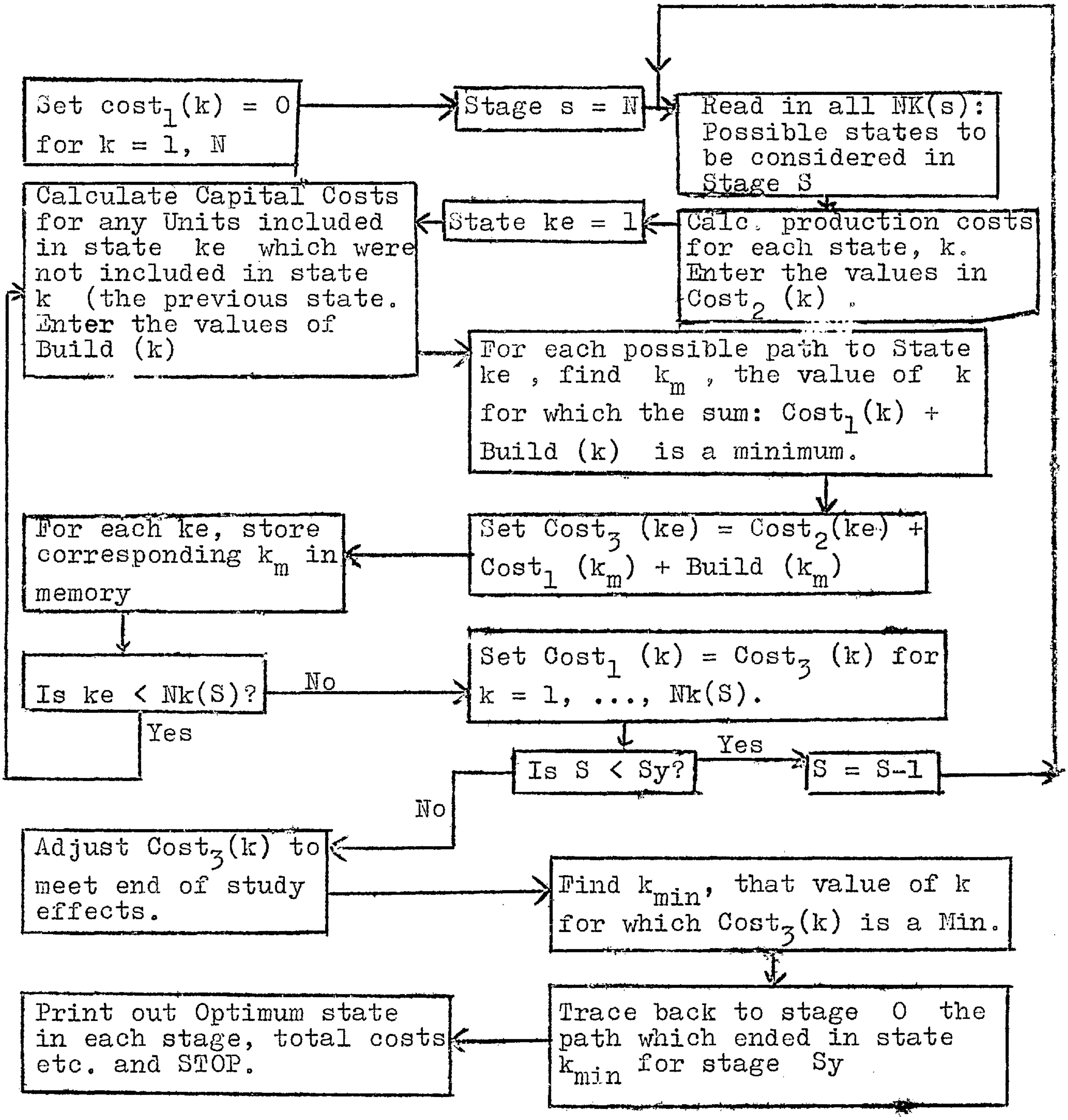

Set cost$_1$(k) = 0 for k = 1, N
Stage s = N
Read in all NK(s): Possible states to be considered in Stage S
Calc. production costs for each state, k. Enter the values in Cost$_2$ (k) .
State ke = 1
Calculate Capital Costs for any Units included in state ke which were not included in state k (the previous state. Enter the values of Build (k)
For each possible path to State ke , find k$_m$, the value of k for which the sum: Cost$_1$(k) + Build (k) is a minimum.
Set Cost$_3$ (ke) = Cost$_2$(ke) + Cost$_1$ (k$_m$) + Build (k$_m$)
For each ke, store corresponding k$_m$ in memory
Is ke < Nk(S)?
No
Yes
Set Cost$_1$ (k) = Cost$_3$ (k) for k = 1, ..., Nk(S).
Is S < Sy?
Yes
S = S-1
No
Adjust Cost$_3$(k) to meet end of study effects.
Find k$_{min}$, that value of k for which Cost$_3$(k) is a Min.
Trace back to stage 0 the path which ended in state k$_{min}$ for stage Sy
Print out Optimum state in each stage, total costs etc. and STOP.

Fig. A.7. A Dynamic Programming Procedure.

$Cost_1(k)$ = Total cumulative cost following the Optimum path from the beginning of the stage (0) to the end of state k in the previous stage.

$Cost_2(k)$ = Total production costs during the stage for the particular make up of existing and new units included in state k .

Build (k) = Cost of any new units that would have to be installed at the beginning of stage, s , in order to go from state k in the previous stage to state (ke) during the current stage.

$Cost_3$ (k) = Total cumulative cost following the optimum path from the beginning of stage 0 to end of state (ke).