LAGRANGE'S MULTIPLIER'S TECHNIQUE FOR NONLINEAR PROGRAMMING PROBLEM

BY

HARUNA MOSHOOD

PGD/MCS/2000/1045

A PROJECT SUBMITTED TO THE MATHEMATICS AND COMPUTER SCIENCE DEPARTMENT, FEDERAL UNIVERSITY OF T ECHNOLOGY, MINNA, NIGERIA.

IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE AWARD OF THE POST GRADUATE DIPLOMA (PGD) IN COMPUTER SCIENCE.

NOVEMBER, 2003

APPROVAL SHEET

This thesis has been read and approved for the partial fulfilment of the Post Graduate Diploma (PGD) in Computer Science .

________________________ ________________________

DR. B. L. ADELEKE DATE
Project Supervisor

________________________ ________________________

MR. L. N. EZEAKO DATE
HOD (Maths and Comp. Sci.)

________________________ ________________________

EXTERNAL SUPERVISOR DATE

DEDICATION

This project work is dedicated to my lovely wife, Engr (Mrs.) Amamat Oluwatoyin Haruna and my little kid Master Abdulnafi'u Y. Haruna.

ACKNOWLWEDGEMENT

First of all, I give thanks to Almighty God for spare my life and for him to have given the wisdom to understand the project and for completing this work despite all the hitches.

My sincere gratitude to **DR. B. L. ADELEKE** who is my project supervisor, for his guidance, constructive criticisms and going through the manuscripts, making corrections and offering useful suggestions throughout the period of making this project and in making it a reality.

I also appreciate the encouragement given to me by the immediate past H.O.D., DR. S.A. Reju, I must not also forget the timely advises given to me always by Mallam Danladi Hakimi. Let me not forget the contributions of all my other lecturers that taught me one course or the other in the course of this peogramme.

Let me also acknowledge the advise and material assistance given to me by my Uncle, Mallam Yusuf Amuda Maigidansanmo. I will like to acknowledge the advise, material assistance and support always given to me by my special and humble friend, Mr. Dauda .O. Isa. I also appreciate the effort of Mrs. Aishatu Abdul for typing my manuscript effectively.

Finally, my gratitude to Almighty Allah for giving me the strength and courage to pick up this project topic, and to be able to conclude the work.

Haruna Moshood Nov; 2003.

TABLE OF CONTENTS

CHAPTER FOUR

CHAPTER FIVE

ABSTRACT

We are proposing to write a project on the Lagrange's Multiplier's technique for Nonlinear Programming Problem in order to find an algorithm/technique that solve the nonlinear programming problems that arise from our day to day activities. It has been observed that most of our daily activities are based on nonlinear programming due to complex nature of decisions to be taken in some aspects of our life, examples are industries, transportation systems, marketing etc. Due to the above, some techniques have to be developed, so as to tackle most of the problems one would encounter in ones daily activities. It is therefore to derive techniques/algorithm that solve nonlinear problems.

Among the methods to be considered in this project are Separable, Quadratic Programming and Lagrange's Multipliers Method. But emphasis will be placed on method(s) that gives best result or best approximation among the methods stated above.

Then from that method that we are going to choose from the above, i.e, Lagrange's Multiplier method (Lagrange's Multiplier Code) will be written and this would solve some nonlinear programming problem encountered in our day to day activities and that is going to be our aim for this project.

CHAPTER ONE

INTRODUCTION TO OPTIMIZATION THEORY

1.1 INTRODUCTION

The problem of finding optima-that is, minima or maxima of real-valued functions plays a central role in Mathematical optimization. We are going to restrict ourselves to case of constrained problems. Here we shall treat the classical Lagrangian multiplier theory and some necessary and sufficient conditions for optima of differentiable functions.

1.2 PROBLEM OF OPTIMIZATION

Optimization is concerned with achieving the best outcome of a given operation while satisfying certain restrictions. Human beings, guide and influenced by their natural surrounding, almost instinctively perform all functions in a manner that economizes energy or minimize discomfort and pain. The motivation is to exploit the available limited resources in a manner that maximizes output or profit. The early inventions of the rocket are clear manifestation of man's desire to maximize moon exploitation.

Physicists, chemists, mathematicians, engineers, economists, operations researchers, managers, and practicing computer scientists are often interested in achieving optimal solutions to their problems. These problems may be to determine designs, programs, trajectories, allocation of resources, approximations of functions. Frequently, different designs or programs, all satisfying the conditions arising from the actual solution are compared, and one is chosen that also as the best in terms of an optimality criterion. Optimization techniques, if properly applied, will automatically examine different designs or plans and select an optimum.

We shall present example 1.1. The WYNDOR GLASS CO produces high quality glass products including windows and glass doors. It has three plants. Aluminium frames and hardware are made in plant A, wood frames are made in plant B, and plant C produce the glass and assembles the products.

Because of declining earnings, top management has decided to revamp the company's product line. Unprofitable products are being discontinued, releasing production capacity to launch two new products having large sales potential:

Product 1: An 8-foot glass door with aluminium framing

Product 2: A 4x6 foot double-hung roof-framed window

Product 1 requires some of the production capacity in plant A and C, but none in plant B, product 2 needs only plants B and C. The marketing division has concluded that the company could sell as much of either product as could be produced by these plants. However, both products would be competing for the same production capacity because in plant C, it is not clear which mix of the two products would be most profitable.

Determine what the production rates should be for the two products in order to maximize their total profit, subject to the restrictions imposed by the limited production capacities available in the three plants.

Table 1.1 summarises the data gathered.

To formulate the mathematical model for this problem, let

x_1 = number of batches of product 1 produced per week

x_2 = number of batches of product 2 produced per week

z = total profit per week from producing these two products

	Production time for Batch, Hours		Production time
	Product		Available per week, Hours
Plant	1	2	
A	1	0	4
B	0	2	12
C	3	2	18
Profit per batch	₦ 3,000	₦ 5,000	

Table 1.1 Data for the Wydor for Glass Co. problem

Thus x_1, and x_2 are the decision variable for the model. Using the bottom row of table 1.1, we obtain

$$Z = 3x_1 + 4x_2$$

Using the data in row 2,3 and 4, then we have, the following:

$$x_1 \leq 4$$
$$2x_2 \leq 12$$
$$3x_1 + 2x_2 \leq 18$$

Plus the restriction $x_1 \geq 0$ and $x_2 \geq 0$. The above model can be rewritten as follows:

Maximize $Z = 3x_1 + 5x_2$

Subject to

$$x_1 \leq 4$$
$$2x_2 \leq 12$$
$$3x_1 + 2x_2 \leq 18$$

and $x_1 \geq 0, x_2 \geq 0$

1.3 CHARACTERISTICS AND TYPES OF MATHEMATICAL MODELS

The problem of optimizing a numerical function of one or more variables when they are constrained in some manner is called a mathematical programming problem, specifically, the purpose of such a problem is to determine the value of n variables $x_1, x_2, \ldots, x_n$ that optimize the function

$$Z = f(x_1, x_2, \ldots x_n) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 1.1$$

Subject to the constraints

$$g_i(x_i, x_2, \ldots x_n)\{\leq = \geq\} b_i i = 1,2,\ldots,m \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 1.2$$

It is usually assumed that the values of the n variables cannot be negative numerically. The nonnegativity restrictions on the variables may be stated as

$$x_j \geq 0 \qquad\qquad j = 1,2,\ldots,n \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad 1.3$$

Also, it is usually desired to determine the optimal value (minimum and maximum) of the function Z in 1.1 which is called the objective function.

The formulation of business and economic questions as mathematical programming problems has resulted in the successful resolution of many complex real-world optimization solutions. Most of the applications of mathematical programming to business and economics involve the maximization of revenues or projects and minimization of costs.

1.3.1 MODEL CLASSIFICATION AND SPECIFIC MODELS OF INTEREST

A real-world optimization problem may be classified in five ways:

1. The functional relations in the problem may be known (deterministic) or uncertain (probabilistic)
2. The function $f(x_1, x_2, \ldots, x_n)$ and $g_i(x_1, x_2, \ldots, x_n), i = 1,2,\ldots,m$ in 1.1 and 1.2 may be linear in $x_1, x_2, \ldots, x_n$; or at least one function in the set may be non linear
3. The functions may be continuously differentiable (smooth) or non differentiable (non smooth)
4. The variables $x_1, x_2, \ldots, x_n$ in the mathematical programming problem may be continuous or may be restricted to integer values
5. The optimization may take place at a fixed point in time (static) or during an interval of time (dynamic)

Most mathematical programming models are deterministic; given $x_1, x_2, \ldots, x_n$, the values of $f, g_1, g_2, \ldots, gm$ are uniquely determined. Most of the current applications of mathematical programming to business and economic problems assume that all model functions are linear there is a very simple reason for this. The simplex method is extremely efficient procedure for solving linear programming problems. When this method is programmed on a computer, it is possible to solve linear problems involving hundreds of variables and thousands of constraints. If one or more of the functions is nonlinear, the problems is always more difficult to solve than linear ones. Thus, even though the real-world problem may be complex and inherently highly nonlinear, successful modeling of it may be possible by using many variables and constraints in a linear formulation.

Most algorithms devised to solve mathematical programming problems require that the functions in the model be continuously, differentiable; thus all functions typically must be smooth. The best known mathematical programming model is linear programming model. All functional in 1.1 and 1.2 are linear in the n-variables $x_1, x_2, \ldots, x_n$. The model may be written as

Optimize $z = f(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} f_j(x_j)$ 1.4

Subject to

$$g_i(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} g_{ij} x_j \{\leq = \geq\} b_i \qquad i = 1,2,\ldots,m$$

where the f_i's, b_i's, and a_{ij}'s are known constants.

The linear programming model has been successfully used to solve a variety of business, economic and scientific problems.

If in the model above, at least one function in the set $f, g_1, g_2, \ldots, gm$ is nonlinear, it is called a nonlinear programming model. We observed that, a nonlinear problem is generally much more difficult to solve than a linear one. Many algorithms have been developed to alleviate this, among which are the following separable programming, Quadratic programming algorithms, Lagrangian multipliers technique etc.

A special case of the general nonlinear programming model, which has received a great deal of attention, is the quadratic programming problem in both chapters 2 and 3. In this model the objective function is quadratic in $x_1, x_2, \ldots, x_n$ and the constraints are linear. Specifically, the model is

$$\text{Optimize } Z = \sum_{j=1}^{n} f_j k_j + \sum_{i=1}^{n} \sum_{j=1}^{n} g_{ij} x_i x_j \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ 1.5$$

$$\text{Subject } \sum_{j=1}^{n} h_{ij} x_j \leq b_i i = 1,2,\ldots,m$$

$$x_j \geq 0 j = 1,2,\ldots,n$$

Where the f_j's, g_{jj}'s and h_{jj}'s are known constants

1.4 FORMULATING AN OPTIMIZATION PROBLEM

An optimization problem is an exercise in mathematical modeling that requires great care in setting up the model. Four steps are involved:

1. Decide the exact objective to be optimized many different objectives are possible.
2. Set up the objective function using as many variables as are required. Try for accuracy rather than compactness. Make sure that all the terms have the same dimensional unit.

3. Set up all the constraints and relationships between the variables
4. If possible, reduce the objective function in step (2) to independent variables
5. The objective function is now ready for solution. If it contains independent variables only, the differentials can be set equal to zero to optimize the expression, or alternatively, tabulation can be made. If the objective function contains dependent variables in addition to independent variables, a Langragian expression can be tried. If this fails, the objective function and its constraints must be optimized, using the skill and ingenuity of the Separable, Quadratic Programmers etc.

Example 1.2

As an illustration of the above, consider the following example. A cheese shop has 20kg of a seasonal fruit mix and 60kg of an expensive cheese with which it will make two cheese spread, delux and regular, that are popular during Christmas week. Each pound of the delux spread consists of 0.2kg of the fruit mix, 0.3kg of the expensive cheese, and 0.5kg of a filler cheese, which is cheap, and in plentiful supply. From past pricing policies, the shop has found that the demand for each spread depends on its price as follows:

$$D_1 = 190 - 25p_1 \text{ and } D_2 = 250 - 50p_2$$

where D denotes demand (in kilograms), P denotes price (in dollars per kg), and the subscripts 1 and 2 refer to the delux and regular spreads, respectively. How many kgs of each spread should the cheese shop prepare, and what prices should it establish, if it wishes to maximize income and he left with no inventory of either spread at the end of Christmas week? (R. Brown).

Solution

Mathematical equivalent of the example

Let x_1 kgs of deluxe spread and x_2 kgs of regular spread be made. If all products can be sold, the objective is

Maximum $Z = p_1x_1 + p_2x_2$

Now, all products will indeed be sold (and none will be left over in inventory if production does not exceed demand, i.e. if $x_1 \leq D_1$ and $x_2 \leq D_2$. This gives the constraints

$$x_1 + 25p_1 \leq 190 \text{ and } x_2 + 50p_2 \leq 250 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 1.7$$

From the availability of fruit mix,

$$0.2x_1 + 0.2x_2 \leq 20 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 1.8$$

and from the availability of expensive cheese,

$$0.8x_1 + 0.3x_2 \leq 60 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 1.9$$

There is no constraint on the filler cheese, since the shop has as much as it needs. Finally, neither production nor price can be negative; so four hidden constraints are $x_1 \geq 0, x_2 \geq 0, p_1 \geq 0$ and $p_2 \geq 0$. Combining these conditions with 1.6 through 1.9, we obtain the mathematical programming problem as follows:

$$\left.\begin{array}{l} \text{Maximize } Z = p_1x_1 + p_2x_2 \\ \text{Subject to : } \begin{array}{l} 0.2x_1 + 0.2x_x \leq 20 \\ 0.8x_1 + 0.3x_2 \leq 60 \\ x_1 + 25p_1 \leq 190 \\ x_2 + 50p_2 \leq 250 \end{array} \end{array}\right] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ 1.10$$

With all variables nonnegative

System 1.10 is a quadratic programming problem in the variables x_1, x_2, p_1, and p_2. It can be simplified if we note that for any fixed position x_1 and x_2 the objective function increases as either p_1 and p_2 increases. Thus for a maximum, p_1 and p_2 must be such that the constraint 1.7 becomes equations; where p_1 and p_2 may be eliminated from the objective function. We then have a quadratic function in x_1 and x_2

$$\left.\begin{array}{l} \text{Maximize } Z = (7.6 - 0.04x_1)x_1 + (5 - 0.02x_2)x_2 \\ \text{Subject to } \begin{array}{l} 0.2x_1 + 0.2x_2 \le 20 \\ 0.8x_1 + 0.3x_2 \le 60 \end{array} \\ \text{With } x_1 \text{ and } x_2 \text{ nonnegative} \end{array}\right] \quad \ldots 1.11$$

1.5 NONLINEAR PROGRAMMING

In this work, emphasis was placed on nonlinear programming than linear programming due to the fact that my work was centred on nonlinear programming problem, both separable and quadratic programming. Algorithms were discussed in chapter 2 while the Lagrangian multiplier's technique was also discussed in chapter 3. Although the simplex method was later utilized in finding solution to both the piecewise linear approximation model and the equivalent linear model of the quadratic programming in chapter 2.

The introduction of nonlinear functions in the mathematical programming problem usually insures more difficulty in solving the problem than if all functions are linear. The primary difficulty introduced by the nonlinear functions in the potential existence of relative or local minima or maxima. The existence of local optima arise due to the nonlinearly of the objective function f(x), the nonlinearity of one or more constraint functions g_i (x), or a combination effect of the nonlinearity in f(x) and in one or more of the constraint functions.

1.6 TYPES OF NONLINEAR OBJECTIVE FUNCTIONS

1.6.1 A NONLINEAR FUNCTION IN ONE VARIABLE

That is, optimizing a nonlinear objective function of a single variable. Note that many of the techniques for solving several variable nonlinear optimization problems actually. To begin, it is convenient to postulate maximization "as the sense of optimization throughout the following discussion. {if the real problem is to minimize an objective function f(x), then you can reformulate the method so as to maximize –f(x)}.

It is assumed that the functions considered possessed continuous first and second derivatives and partial derivative everywhere. Consider a function of a single variable, such as that shown in figure 1.1. A necessary condition for a particular solution x- x* to be either a minimum or maximum is that

$$\frac{df(x)}{dx} = 0 \quad \text{at } x=x^* \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 1.12$$

Thus in figure 1.3.1, there are five solutions satisfying these conditions. To obtain more information about these five so called critical points, it is necessary to examine the second derivative. Thus,

$$\frac{d^2 f(x)}{dx^2} > 0 \quad \text{at } x=x^* \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 1.13$$

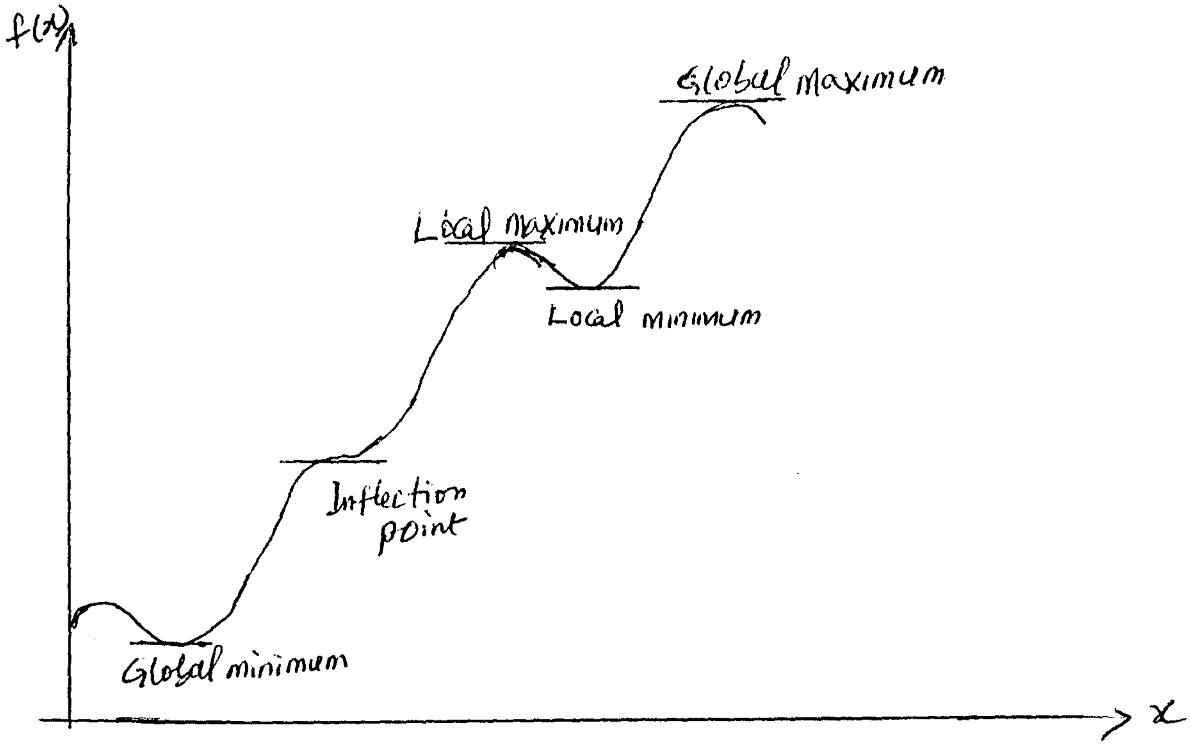


Fig 1.1 A function having several maxima and minima

Then x* must be at least a local minimum (ie $f(x^*) \leq f(x)$ for all x sufficiently close to x*). So x* must be a local minimum if f(x) is strictly convex with neigbourhood of x*. Similarly, a sufficient condition for x* to be a local maximum (given that it satisfies the necessary condition) is that f(x) is strictly concave with a neighbourhood of x* (that is, the second derivatives is negative at x. if the second derivatives is negative at x. If the second derivative is zero, the point may not even be an inflection point and it is necessary to examine higher derivatives.

To first a global minimum (ie a solution x* such that $f(x^*) \leq f(x)$ for all x) it is necessary to compare the local minima and identify the value is less than f(x) as $x \to -\infty$ and as $x \to +\infty$ (or at the endpoints of the function, if it is only defined over a finite interval) then his point is a global minimum.

However, if f(x) is known to be either a convex or concave function, in particular, if f(x) is a convex function, then any solution x*, such that

$$\frac{df(x)}{dx} = 0 \qquad \text{at } x = x^*$$

is known automatically to be a global minimum. In other words this condition is not only a necessary but a sufficient condition for a global minimum of a convex function. If this function is strictly convex, then this solution must be the only global minimum. Similarly, if f(x) is a concave function, then having

$$\frac{df(x)}{dx} = 0 \qquad \text{at } x = x^*$$

becomes both necessary and sufficient condition for x* to be a global maximum. If for any x_1 and x_2 in $I = [-\infty, \infty]$, where $x_1 < x_2$, and for all ρ, $0 \le \rho \le 1, f(x)$ satisfies $\rho f(x_1) + (1-\rho) f(x_2) \ge [\rho x_1 + (1-\rho) x_2]$ convex function 1.14

A function is unimodal wherever it is concave, that is, if for any x_1 and x_2 in I, where $x_1 < x_2$ and for all $\rho, 0 \le \rho \le 1, f(x)$ satisfies $\rho f(x_1) + (1-\rho) f(x_2) \le f[\rho x_1 + (1-\rho) x_2]$ concave function 1.15

1.6.2 A NONLINEAR FUNCTION OF SEVERAL UNCONSTRAINED VARIABLES

That is, maximizing a nonlinear function of several unconstrained variables. There are two motivating reasons for studying this problem. Firstly, an analysis of the multidimensional, unconstrained, nonlinear maximization problem sets the stage for the analysis of constrained models. The algorithmic difficulties to be overcome here are also present in the constrained cases. Secondly, a constrained problem can often be solved by first converting it to an unconstrained problem. We postulate that $f(x_1,x_2,\ldots,x_n)$ is smooth and posses a finite maximum value, occurring at the finite values $x^*_1, x^*_2, \ldots, x^*_n$. Abbreviating a set of values for $x_1,x_2,\ldots,x_n$ by the symbol x, and the expression $f(x_1,x_2,\ldots,x_n)$ by the symbol f(x), these assumptions can be stated more precisely as:

i. For all values of x, f(x) is uniquely defined and finite

ii. For all values of x, every partial derivatives $\partial f / \partial x_j$ is uniquely defined, finite, and continuous, and hence f(x) is continuous

iii. f(x) possesses a finite maximum f*

iv. For any possible value of f(x), say f, there exists an associated finite number M_f such that every $|x_j| \le m_f$ if $f(x) \ge f$

Applying differential calculus, we can state the following. Necessary condition for maximum. Given assumptions (i) through (iii), the function f(x) has a maximum at x* only if $\partial f / \partial x_j = 0$ for j=1,2,…,n

The validity of the result is easy to see. Suppose there is a variable x_j such that $\partial f(x^*) / \partial x_j > 0$. Then f(x) can be increased by increasing x^*_j by a small amount. Analogously, if $\partial f(x^*) / \partial x_j < 0$, then f(x) can be increased by decreasing x^*_j by a small amount. But unfortunately, without imposing further restrictions on the shape of f(x), the necessary condition is not sufficient for a maximum. x* may not maximize f(x) when all $\partial f(x^*) / \partial x_j = 0$. The illustration in Fig 1.2 shows why. The derivative of $\partial f / \partial x_j = 0$ at points a,b,c,d,e as well as at g, which gives the only global maximum.

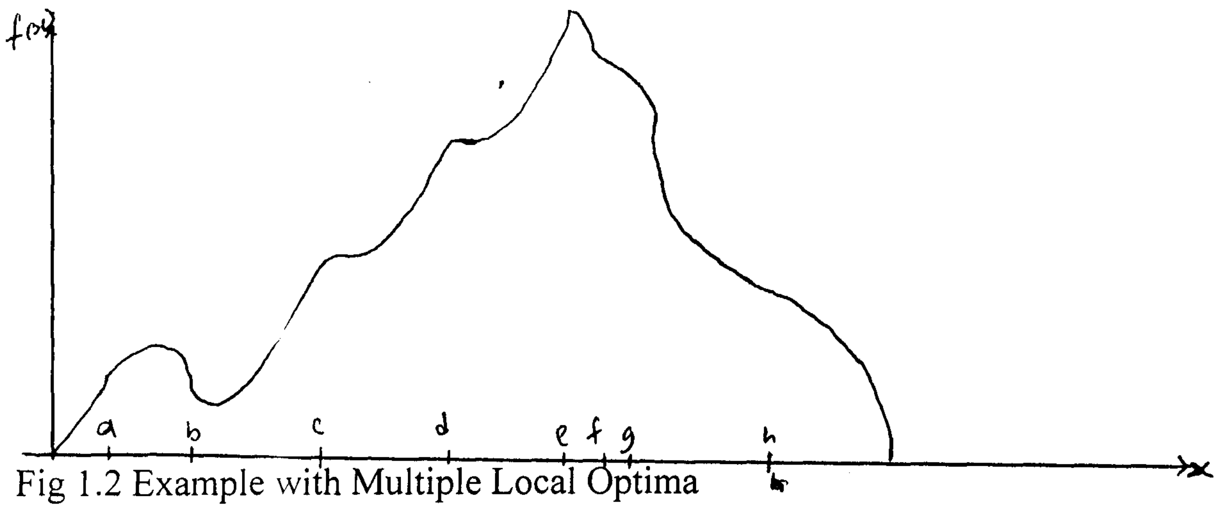


Fig 1.2 Example with Multiple Local Optima

After identifying the critical points that satisfy the condition $\partial f(x_1, x_2, ... x_n) / \partial x_j = 0$ at $= (x_1, x_2, ..., x_n) = (x_1^*, x_2^*, ..., x_n^*)$ for j = 1,2,...,n, each of such point would then be classified as a local minimum or maximum if the function is strictly convex or strictly concave respectively, within a neighbourhood of the point. The global minimum and maximum would be found by comparing the relative minima and maxima and the checking the value of the function as some of the variables approach $-\infty$ or $+\infty$. However, if the function is known to be convex or concave, than a critical point must be a global minimum or a global maximum respectively.

1.6.3 A NONLINEAR FUNCTION OF SEVERAL CONSTRAINED VARIABLES

That is, optimizing a nonlinear function with nonlinear constraints. The aim here is to solve optimization problems containing nonlinear constraints. For the sake of definiteness, suppose the model is states as

Maximize $f(x_1, x_2, ..., x_n)$ 1.16

Subject to $g_i(x_1, x_2, ..., x_n) \leq 0$ i=1,2,...,m............1.17

$x_j \geq 0$ j=1,2,...,n.......................................1.18

1.16 and 1.17 above can be viewed as a canonical statement of a nonlinear programming problem (NPP). Here, the constraints function $g_i(x)$ and objective function f(x) are to be postulated upon as follows:

Definition 1.1 Feasible region.

The assumption on each nonlinear function $g_i(x)$ are given in terms of its shape and smoothness characteristics. To set the stage, a real value function g(x) is defined to be convex if, for any two points $x \neq y$, and for all $\rho, 0 \leq \rho \leq 1$.

$$\rho g(x_1, x_2, \ldots, x_n) + (1-\rho) g(y_1, y_2, \ldots, y_n) \geq g[\rho x_1 + (1-\rho) y_1, \ldots, \rho x_n + (1-\rho) y_n]$$

convex......................1.19

and strictly convex if there is a strictly inequality (>) for $0 < \rho < 1$

(Note that if – g (x) is concave, then g(x) is convex).

A related characteristic of a convex function is that for any two points x and y,

$$g(y) \geq g(x) + \sum_{j=1}^{n} \frac{\partial g(x)}{\partial x_j}(y_j - x_j) \quad \text{convex} \ldots \quad 1.20$$

$g_i(x)$ in 1.20 satisfy the following shape and smoothness assumptions.

i. Each $g_i(x)$ is uniquely defined, finite, and convex for all values of $(x_1, x_2, \ldots x_n)$.

ii. Each $\partial g_i(x)/\partial x_j$ is continuous for all x satisfying the constraints in 120

DEFINITION 1.2: OBJECTIVE FUNCTION:

The function f(x) is also hypothesized to satisfy certain shape and smoothness assumptions (i) through (iv).

i. f(x) is single-valued and finite for each x satisfying the constraints 1.20.

ii. Every partial-derivative $\partial f(x)/\partial x_j$ is a single-valued finite, and continuous at and each x satisfying the constraint 1.20

iii. f(x) possesses a finite maximum f* over all values of a satisfying the constraints 1.20

iv. f(x) is concave over all values of x satisfying the constraints 1.20

It is the purpose of this chapter to develop the basic theory upon which methods devised to solve the nonlinear programming problem are typically based. Among the topics considered are the definitions of local and global optima, the necessary and sufficient conditions introduced into this identification process by nonlinearity.

The final section contains some applications of this material to nonlinear optima.

1.7 LOCAL AND GLOBAL OPTIMA

The concepts of local and global optima play an extremely important role in nonlinear programming.

Definition 1.3 – Global maximum (unconstrained problem), the unconstrained function f(x) is said to take on its global maximum at the point x* if $f(x) \leq f(x^*)$ for all x over which the function f(x) is defined.

Definition 1.4 – Local maximum (unconstrained problem) The unconstrained function f(x) is said to take on a local maximum at the point x^o if constants ϵ and $\delta, 0 < \epsilon < \delta$, exist such that for all x satisfying $0 < |x - x^o| < \epsilon, f(x) \leq f(x^o)$, whose f(x) is defined for all points in some δ-neighbouhood of x^o.

Figure 1.1 illustrates a local and global maximum for a univariate function. Notice from definition 1.3 and 1.4 that a global is also a local maximum. A familiar theorem from differential calculus is now introduced, which states the necessary conditions for a point x^0 to be a local (or global) maximum.

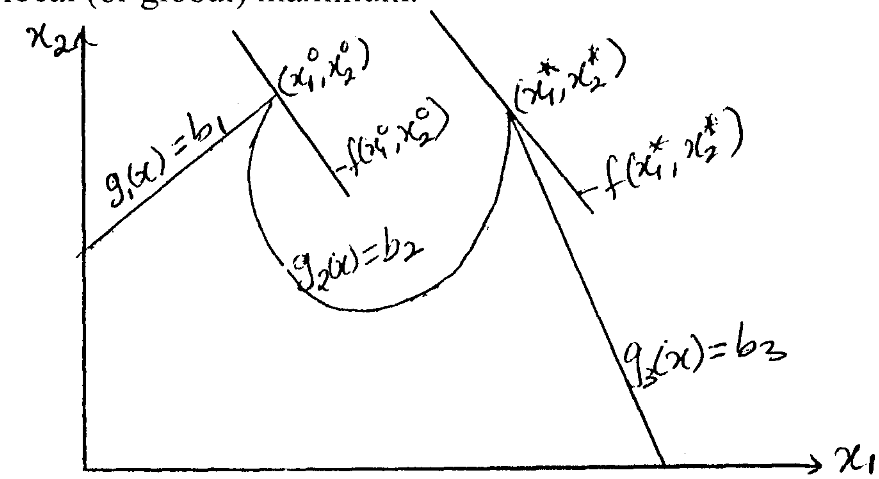


Fig. 1.3 Illustration of Local Optima

Theorem 1.1 f(x) assumes a relative (local) maximum at x^0 then x^0 must be a solution to the set of negations

$$\frac{\partial f(x)}{\partial x_j}0 \qquad j=1,2,\ldots,n$$

proof

Suppose that f(x) assumes a local maximum at x^0 . Then from the definition of a local maximum, an Ebo must exist such that for all points x in a δ-neighbourhood of x^0, $f(x) \le f(x^o)$. In particular, consider a point in the δ-neighbourhood of x^0 of the $x = x^o + he_j$ where $e_j' = [0,0,\ldots,0,1,0,\ldots,0]$ with the 1 placed in the jth position of e_j and $0 < | < | < \epsilon$. Then

$$f(x^o + he_j) \le f(x^o) \qquad j=1,2,\ldots,n \;\ldots\ldots\ldots\ldots\ldots\ldots \; 1.21$$

for all h, $0<|h|<\epsilon$. Dividing 1.21 by h results in the expressions

$$\frac{f(x^o + he_j) - f(x^o)}{h} \le 0 \text{ if h>0 } j=1,2,\dots,n \quad\dots\dots\dots\dots\dots\dots 1.22$$

$$\frac{f(x^o + he_j) - f(x^o)}{h} \ge 0 \text{ if h<0 } j=1,2,\dots,n \quad\dots\dots\dots\dots\dots 1.23$$

On taking the limits of 1.22 and 1.23 as h -> 0, it follows from the definition of partial derivative that's

$$\frac{\partial f(x^o)}{\partial x_j} \le 0 \quad \text{for h -> 0} \qquad h > 0$$

$$\frac{\partial f(x^o)}{\partial x_j} \ge 0 \quad \text{for h -> 0} \qquad h<0$$

Thus

$$\frac{\partial f(x^o)}{\partial x_j} = 0 \qquad j=1,2,..,n \quad\dots\dots\dots\dots\dots 1.24$$

The condition in 1.24 can be conveniently displayed in vector notation in terms of the gradient vector of f(x).

Definition 1.5 the Gradient vector:

The gradient vector of f(x) = f($x_1,x_2,\dots,x_n$), denoted by $\nabla f(x)$, is the nx1 column vector whose components are in the first-order partial derivatives of f(x):

$$\nabla f(x) = \begin{bmatrix} \partial f(x) / \partial x_1 \\ . \\ . \\ . \\ \partial f(x) / \partial x_n \end{bmatrix} \qquad \text{.........................} \ 1.25$$

The condition in 1.24 stated in vector form is $\nabla f(x^o) = 0$

If a point x^o satisfies 1.24, it might not be a maximizing point. Theorem 1.1 provides only the necessary condition for x^o to be maximizing point. In the univariate 1.24 may be satisfied at a minimizing point, a maximizing point, or a point of inflection as illustrated in figure 1.4. In the n-multidimensional case where $x' = [x_1, x_2, ..., x_n]$, the analogy to the univariate case is a minimizing point, maximizing point, or saddle point. A saddle point is the multidimensional analogy to the inflection point in the univariate case. A saddle point for the bivariate case $[x' = (x_1, x_2)]$ is illustrated in Fig. 1.5.

The sufficient condition for x^o to be a maximizing point can be expressed as a property of the Hessian matrix of f(x).

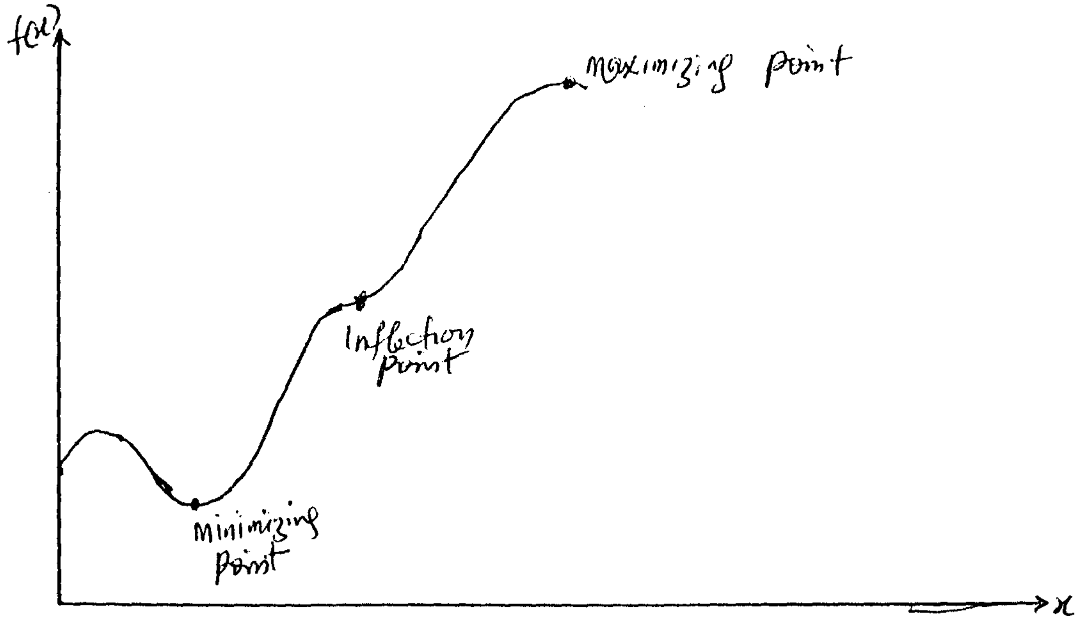


Fig. 1.4 Possible Solution Points to $df(x)/dx_j=0$

Definition 1.6 The Hessian Matrix

The Hessian matrix of $f(x) = f(x_1,x_2,\ldots,x_n)$, denoted by $H(x)$, is the nxn matrix whose elements are the second order partial derivatives of $f(x)$:

$$H(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_2^2} & --- & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_1^2 \partial x_2} & \frac{\partial^2 f(x)}{\partial x_2^2} & --- & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ - & & & \\ - & & & \\ - & & & \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & --- & \frac{\partial^2 f(x)}{\partial x_n^2} \end{bmatrix} \quad \ldots\ldots\ldots 1.26$$

Theorem 1.2. A sufficient condition for $f(x)=f(x_1,x_2,\ldots,x_n)$ to have a local maximum at the point x^o where $\nabla f(x^o) = 0$ is that the Hessian matrix H(x) be negative definite is for any $y' = [y_1, y_2, \ldots, y_n]$, except y=0, y'H(x)y<0

Proof:

This theorem can be proved by applying Taylor's theorem to the function f(x). Taylor's theorem states that for any two points x_1 and $x_2=x_1+h$, there exists a scalar θ, $0 \le \theta \le 1$, such that

$$f(x_2) = f(x_1) + \nabla f'(x_1)h + 0.5h' H[\theta x_1 + (1-\theta)x_2]h \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \; 1.27$$

Applying 1.27 to f(x), where $x_1 = x^o$ and $x_2 = x^o th$, produces the expression

$$f(x^0 th) = f(x^o) + \nabla f'(x^o)h + 0.5h' H[\theta x^o + (1-\theta)(x^o + h)]h$$

since $\nabla f(x^o) = 0$

$$f(x^0 th) = f(x^o) + 0.5h' H[\theta x^o + (1-\theta)(x^o + h)]h$$

or

$$f(x^0 th) - f(x^o) = 0.5h' H[\theta x^o + (1-\theta)(x^o + h)]h \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \; 1.28$$

If the right-hand side of 1.28 is negative for all h in a δ-neighbourhood of x^o, by definition 1.4, x^o must be a local maximum, since $f(x^o th) - f(x^o) \le 0$ if this is the case. The second partial derivatives $\partial^2 f(x^o)/\partial x_i \partial x_j$ will have the same sign as $\partial^2 f[\theta x^o + (1-\theta)(x^o th)]/\partial x_i \partial x_j$ provided that the point $\theta x^o + (1-\theta)(x^o th)$ is in a suitable δ-neighbourhood of x^o. Thus the right hand side of 1.28 is negative only if h'H(x)h<0; i.e. the Hessian matrix evaluated at x^o, $H(x^o)$, must be negative definite to insure that x^o is a maximizing point.

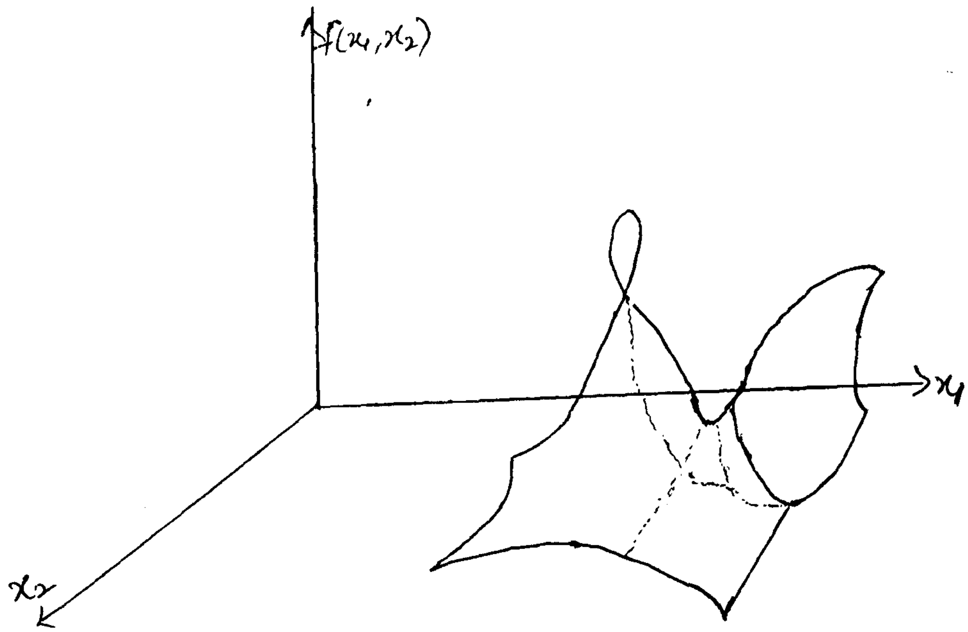


Fig. 1.5 Two Dimensional Saddle Point

We shall present example 1.3, as an illustration. Example 1.3 determine the maximum of:

$f(x) = f(x_1, x_2, x_3) = 16x + 24x_2 - 4x_1^2 - 3x_2^2 - x_3^2$

(R.C. Pfaffenberger & D.A. Walker)

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \frac{\partial f(x)}{\partial x_2} \end{bmatrix} = \begin{bmatrix} -8x_1 + 16 \\ -6x_2 + 24 \\ -2x_3 \end{bmatrix} \text{set} \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

The condition $\nabla f(x)=0$ generate a system of three linear equations three unknowns. The solution to his system is $x' = [x_1, x_2, x_3] = [2,4,0]$.

The Hessian matrix H(x) is now determined.

The Hessian matrix H(x) is now determined.

$$\frac{\partial^2 f(x)}{\partial x_1^2} = -8 \qquad \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} = 0 \qquad \frac{\partial^2 f(x)}{\partial x_1 \partial x_3} = 0$$

$$\frac{\partial^2 f(x)}{\partial x_2^2} = -6 \qquad \frac{\partial^2 f(x)}{\partial x_2 \partial x_2} = 0 \qquad \frac{\partial^2 f(x)}{\partial x_3} = -2$$

Thus the Hessian matrix evaluated at $x^o = [2,4,0]$ is

$$H(x^o) = \begin{bmatrix} -8 & 0 & 0 \\ 0 & -6 & 0 \\ 0 & 0 & -2 \end{bmatrix}$$

The sealer quantity $y'H(x^o)y$ is

$$[y_1 \ \ y_2 \ \ y_3] = \begin{bmatrix} -8 & 0 & 0 \\ 0 & -6 & 0 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

$$= -8y_1^2 - 6y_2^2 - 2y_3^2$$

which is clearly less than zero for any $y' = [y_1 y_2 y_3], y \neq 0$

Thus $x^o = [2,4,0]$ is a maximizing point.

CHAPTER TWO

TYPES OF NONLINEAR PROGRAMMING PROBLEMS

2.1 INTRODUCTION

Linear programming methods developed in the 1950s can be used very effectively in cases where both the constraints and the function to be optimized are linear. Variations of linear programming methods are also available for cases where the function to be optimized is quadratic (quadratic programming) and for cases where the nonlinear constraints can be expressed as pieceuise linear functions (separable programming). Examples of each packages for nonlinear problems are MINOS, GRAMS/MINOS and GINO.

2.2 SOLUTION TECHNIQUES FOR NONLINEAR PROGRAMMING PROBLEM

Here, separable programming problems can be solved by the simplex method, because any such problem can be approximated as closely as desired by a linear programming problem with a larger number of variables.

It is assumed that the objective function f(x) is concave, that each of the constraint function $g_i(x)$ is convex, and that all these functions are separable functions (functions where each term involves just a single variable). We focus here on the special case where the convex and separable $g_i(x)$ are, in fact, linear functions just as for linear programming. Thus only the objective function requires special treatment.

Under the preceeding assumptions, the objective function can be expressed as a sum of concave function of individual variables

$$f(x) = \sum_{j=1}^{n} f_j(x_j)$$

So that each $f_j(x_j)$ has a shape such as the one shown in Fig. 2.1 (either case) over the feasible range of values of values of x_j. Because f(x) represents the measure of performance (say, profit) for all the activities together, $f_j(x_j)$ represents the contribution of profit from activity j when it is conducted at level xj. The condition of f(x) being separable simply implies activity i.e., there are no interactions between the activities (no cross product terms) that affect total profit beyond their independent contributions. The assumption that each $f_j(x_j)$ is concave says that the marginal profitability (slope of the curve) either stays the same or decreases (never increases) as xj is increased.

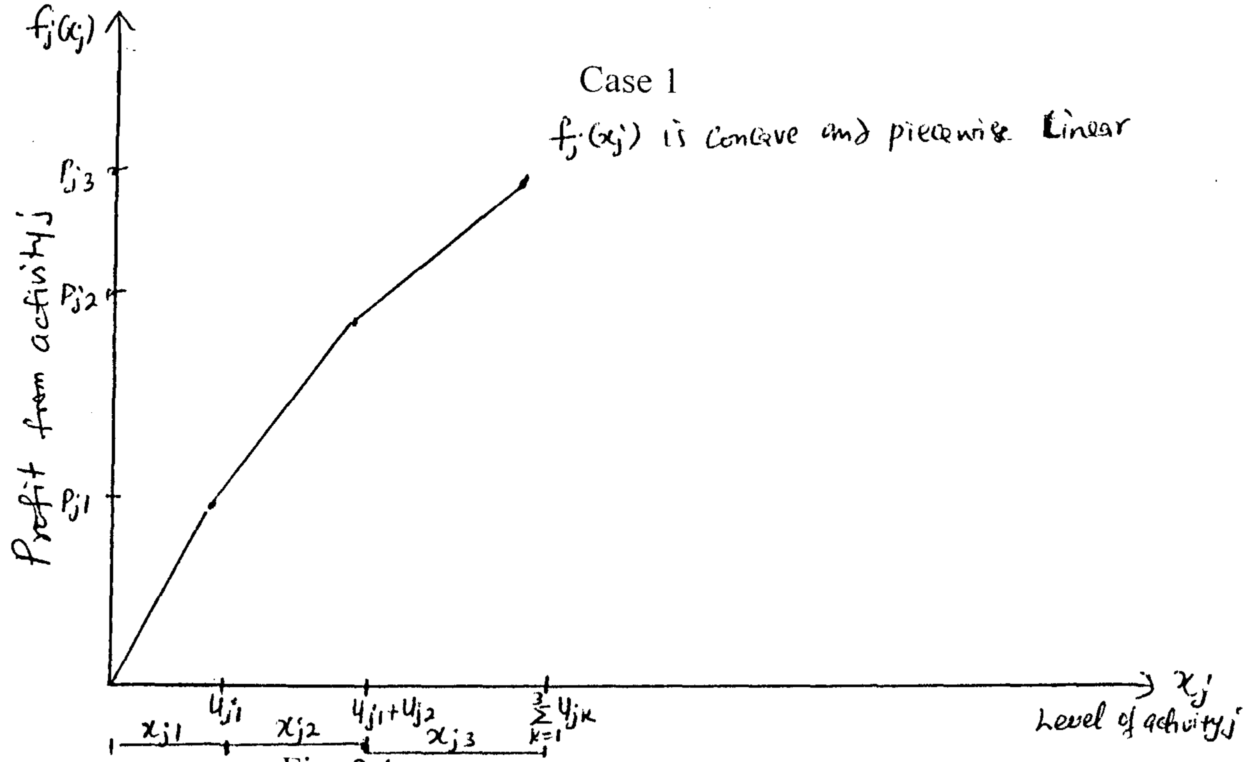


Fig. 2.1

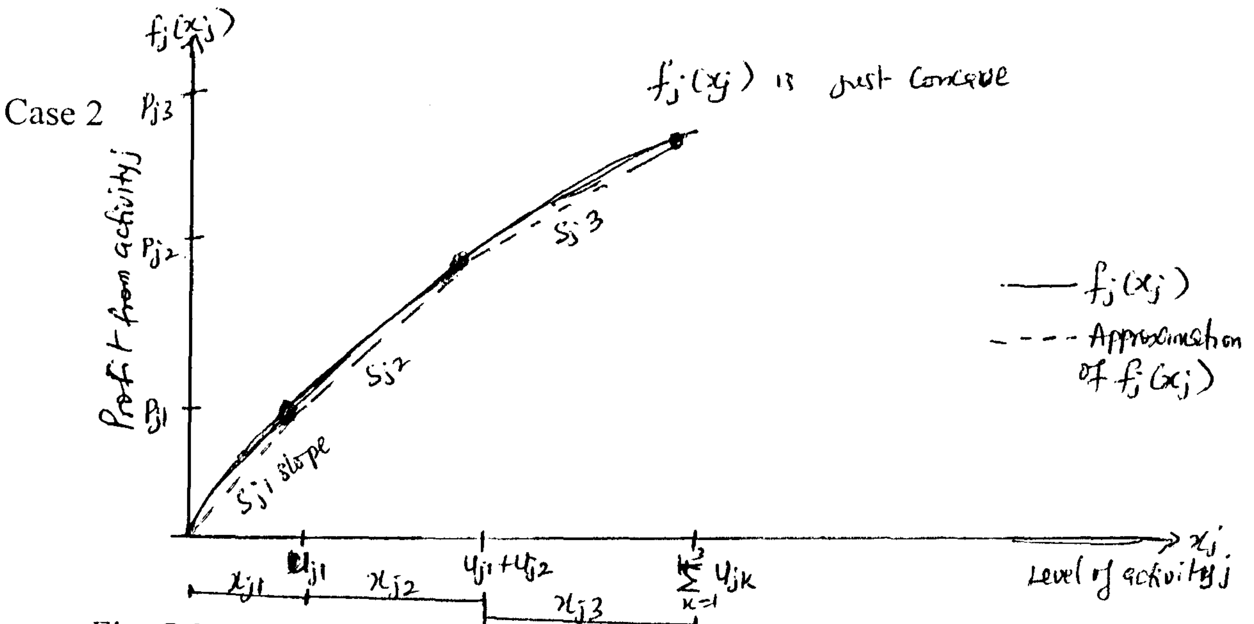


Fig. 2.2.1 Shape of profit curves for separable programming

Concave profit curves occur quite frequently. For example, it may be possible to sell a limited amount of some product at a certain price, then a further amount at a lower price, and perhaps finally a further amount at a still lower price. Similarly, it may be necessary to purchase raw materials from increasingly expensive sources. In another common situation, a more expensive production process must be used (eg over time rather than regular-time work) to increase the production rate beyond a certain point.

These kinds of situation can lead to either type of profit curve shown in Fig. 2.1. In case 1, the slope decreases only at certain breakpoints, so that $f_j(x_j)$ is a piecewise linear function (a sequence of connected line segments). For case 2, the slope may decrease continuously as xj increases, so that $f_j(x_j)$ is a general concave function. Any such function can be approximated as closely as deserved by as needed for separable programming problems. (Figure 2.1 shows an approximating function that consists of just three line segments, but the approximation can be made even better just by introducing additional breakpoints). This approximation is very convenient because a piecewise linear function of a single variable can be rewritten as a linear function of several variables, with one special restriction on the values of these variables, as described next.

2.3 REFORMULATION AS A LINEAR PROGRAMMING PROBLEM

The key to rewriting a piecewise linear function is to use a separate variable for each line segment. To illustrate, consider the piecewise linear function $f_j(x_j)$ shown in Figure 2.1, case 1 (or the approximating piecewise linear function for case 2), which has three line segments over the feasible range of values of xj. Introduce the three new variable x_{j1}, x_{j2}, and x_{j3} and set

$$x_j = x_{j1} + x_{j2} + x_{j3} \qquad 2.1$$

where

$$0 \leq x_{j1} \leq u_{j1}, \qquad 2.2$$

$$0 \leq x_{j2} \leq u_{j2}, \qquad 2.3$$

$$0 \leq x_{j3} \leq u_{j3} \qquad 2.4$$

Then use the slope s_{j1}, s_{j2}, and s_{j3} to rewrite $f_j(x_j)$ as

$$f_j(x_j) = s_{j1} x_{j1} + s_{j2} x_{j2} + s_{j3} x_{j3} \qquad 2.5$$

With special restriction that

$x_{j2} = 0$ whenever $x_{j1} < u_{j1}$,

$x_{j3} = 0$ " $x_{j2} < u_{j2}$,

To see why this special restriction is required, suppose that $x_j = 1$, where $u_{jk} > 1$ (k=1,2,3), so that $f_j(1) = s_{j1}$. Note that

$$x_{j1} + x_{j2} + x_{j3} = 1$$

permits

$x_{j1} = 1$, $x_{j2} = 0$, $x_{j3} = 0$ => $f_j(1) = s_{j1}$,

$x_{j1} = 0$, $x_{j2} = 1$, $x_{j3} = 0$ => $f_j(1) = s_{j2}$,

$x_{j1} = 1$, $x_{j2} = 0$, $x_{j3} = 1$ => $f_j(1) = s_{j3}$,

and soon, where

$$s_{j1} > s_{j2} > s_{j3}.$$

However, the special restriction permits only the first possibility, which is the only one giving the correct value $f_j(1)$.

Unfortunately, the special restriction does not fit into the required format for linear programming constraints, so some piecewise linear functions cannot be rewritten in a linear programming format. However, our $f_j(x_j)$ are assumed to be concave, so $s_{j1} > s_{j2} >\ldots$, so that an algorithm for maximizing f(x) automatically gives the highest priority to using x_{j1} when (in effect) increasing x_j from zero, the next highest priority to using x_{j2} , and so on, without even including the special restriction explicitly in the model. This observation leads to the following key property.

2.4 KEY PROPERTY OF SEPARABLE PROGRAMMING

When f(x) and the $g_i(x)$ satisfy the assumptions of separable programming, and when the resulting piecewise linear functions are rewritten as linear functions, deleting the special restriction gives a linear programming model whose optimal solution automatically satisfies the special restriction.

To write down the complete linear programming model in the above notation, let n_j be the number of line segments in $f_j(x_j)$ (or the piecewise linear function approximating it). So that

$$x_j = \sum_{k=1}^{n_j} x_{jk} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad 2.6$$

would be submitted throughout the original model and

$$f_j(x_j) = \sum_{k=1}^{n_j} \lambda_{jk} x_{jk} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad 2.7$$

would be submitted into the objective function for j=1,2,...n. The resulting model is

$$\text{maximize } z = \sum_{j=1}^{n}\left(\sum_{k=1} \lambda_{jk} x_{jk}\right) \text{ .... } 2.8$$

subject to

$$\sum_{j=1}^{n} a_{ij}\left(\sum_{k=1}^{n_j} x_{jk}\right) \leq b_i \text{ for I=1,2,...,m .... } 2.9$$

$$x_{jk} \leq u_{jk}, \text{ for k=1,2,...,} n_j \text{ : J=1,2,...,n .... } 2.10$$

and

$$x_{jk} \geq 0 \qquad \text{for } k=1,2,\ldots,n_j : j=1,2,\ldots,n \text{ .... } 2.11$$

(The $\sum_{k=1}^{n} x_{jk} \geq 0$ constraints are deleted because they are ensured by the $x_{jk} \geq 0$ constraints) If some original variable x_j has no upper bound, then $u_{jnj} = \infty$, so the constraint involving this quantity will be deleted.

An efficiently way of solving this model is to use the stream lined version of the simplex method for dealing in the upper bound constraints. After obtaining an optimal solution for this model, you then would calculate

$$x_j = \sum_{k=1}^{n_j} x_{jk} ,$$

for j=1,2,...,n in order to identify an optional solution for the original separable programming program (or its piecewise linear approximation).

Example 2.1

Consider the following:

Maximize $x_1^2 - x_1 + x_2$

Subject to $x_1 + x_2^2 \le 4$

$x_1, x_2 \ge 0$

(G.E. Whitehouse, B.L. Wechsler)

Solution

Step 1

$f(x_1,x_2) = f_1(x_1) + f_2(x_2)$

where

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

$g_i = g_{i1}(x_1) + g_{i2}(x_2) + \ldots + g_{in}(x_n)$ i=1,2,...,m

so $g_1(x_1, x_2) = g_{11}(x_1) + g_{12}(x_2)$

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

Step ii

From the original problem we see that both x_1 and x_2 must be ≥ 0

The first constraint indicates that $x_1 \leq 4$ and $x_2 \leq 2$ (The variables do not necessarily need to have the same domain).

In our case, let us partition the domain of each variable into four segments, thus we will have five grid points.

Step III

At $k = 0$, $x_{1k} = x_{10} = 0$

At $k = 1$, $x_{1k} = x_{11} = 1$

Also $f_1(x_{1k}) = x_{1k}^2 - x_{1k}$

At $k = 0$, $x_{10}=0$

Thus $f_1(x_{1k}) = 0$, $k = 0$

Similarly, $f_1(x_{1k}) = 0$, $k = 1$

Use the table below to evaluate the separate functions

Fig. 2.1 Evaluation Table.

k	x_{1k}	x_{2k}	$g_{11}(x_{1k})$	$g_{12}(x_{2k})$	$f_1(x_{1k})$	$f_2(x_{2k})$
0	0	0	0	0	0	0
1	1	.5	1	.25	0	.5
2	2	1	2	1	2	1
3	3	1.5	3	2.25	6	1.5
4	4	2	4	4	12	2

Step iv

The original problem can now be written as follows:

$$\text{Maximize } f(x_1,x_2) \cong \sum_{k=0}^{4} \lambda_{1k} f_{1k} + \sum_{k=0}^{4} \lambda_{2k} f_{ik} =$$

$$0\lambda_{10} + 0\lambda_{11} + 2\lambda_{12} + 6\lambda_{13} + 12\lambda_{14} + 0\lambda_{20} + 5\lambda_{21} + \lambda_{22} + 1.5\lambda_{23} + 2\lambda_{24}$$

subject to

$$g_1(x_1,x_2) \cong \sum_{k=0}^{4} \lambda_{1k} g_{1k} + \sum_{k=0}^{4} \lambda_{2k} g_{2k} =$$

$$0\lambda_{10} + \lambda_{11} + 2\lambda_{12} + 3\lambda_{13} + 4\lambda_{14} + 0\lambda_{20} + .25\lambda_{21} + \lambda_{22} + 2.25\lambda_{23} + 4\lambda_{24} \leq 4$$

$$\sum_{k=0}^{4} \lambda_{1k} = \lambda_{10} + \lambda_{11} + \lambda_{12} + \lambda_{13} + \lambda_{14} = 1$$

$$\sum_{k=0}^{4} \lambda_{2k} = \lambda_{2k} + \lambda_{21} + \lambda_{22} + \lambda_{23} + \lambda_{24} = 1$$

$$\lambda_{jk} \geq \begin{cases} j=1,2 \\ k=0,1,2,3,4 \end{cases}$$

Step v

By introducing a slack variable s, we can write the first constraints as an equality. Proceed by applying simplex method to solve the piecewise linear functions as follows:

Basis	λ_{10}	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{20}	λ_{21}	λ_{22}	λ_{23}	λ_{24}	S	b
S	0	1	2	3	4	0	.25	1	2.25	4	1	4
λ_{10}	1	1	1	1	1	0	0	0	0	0	0	1
λ_{20}	0	0	0	0	0	1	1	1	1	1	0	1
	0	0	2	6	12	0	.5	1	1.5	2	0	0

Initial tableau (Figure 2.2)

By letting λ_{14} enter the basis, λ_{14} will replace S while λ_{10} and λ_{20} remain in the basis. Clearly λ_{10} and λ_{14} are not adjacent points and this is a situation we cannot tolerate. Alternatively, we allow λ_{14} to replace λ_{10}, in that case we would have S, λ_{14} and λ_{20} in the basis, a perfect condition as shown in the next table.

Improved Solution

Basis	λ_{10}	λ_{11}	λ_{12}	λ_{13}	λ_{14}	λ_{20}	λ_{21}	λ_{22}	λ_{23}	λ_{24}	S	b
S	-4	-3	-2	-1	0	0	¼	1	9/4	4	1	0
λ_{14}	1	1	1	1	1	0	0	0	0	0	0	1
λ_{20}	0	0	0	0	0	1	1	1	1	1	0	1
	-12	-12	-10	-6	0	0	.5	1	1.5	2	0	-12

First tableau (Figure 2.3)

The first-tableau yields the solution $\lambda_{1.4}=1$, $\lambda_{20} = 1$, all other variables $= 0$. Either λ_{24}, λ_{22} or λ_{21} would make b_i negative or the adjacency condition would not be met, if any other variable enters. We have therefore found the optional solution to our linear piecewise approximation.

It now only remains to translate our solution into terms of the original variables x_1 and x_2

Step vi

$$x_1 = \sum_{k=0}^{4} \lambda_{1k} x_{1k} = (0)(0) + (0)(1) + (0)(2) + (0)(3) + (1)(4) = 4$$

and

$$x_2 = \sum_{k=0}^{4} \lambda_{2k} x_{2k} = (11)(0) + (0)(.5) + (0)(1.5) + (0)(2) = 0$$

and the evaluation of the objective function yields

$$x_1^2 - x_1 + x_2 = 4^2 - 4 + 0 = 12$$

2.5 QUADRATIC PROGRAMMING TECHNIQUE

Quadratic programming problems again have linear constraints, but the objective function f(x) must be quadratic. Thus, the only difference between them and a linear programming is that some of the terms in the objective function involve the square of a variable or the product of two variables.

We assumed here, that f(x) is concave. So we are going to apply the modified simplex method to solve our problem. The quadratic programming problem differs from the linear programming problem only in that the objective function also includes x_j^2 and $x_i x_j$ $(i \neq j)$ terms. Thus, if we use matrix notation, the problem is to find x so as to

$$\text{Maximize } f(x) = Cx - \tfrac{1}{2} x^T Qx \quad \ldots\ldots \quad 2.12$$

Subject to

$$Ax \leq b \qquad \text{and } x \geq 0 \quad \ldots\ldots \quad 2.13$$

Where C is a row vector, x and b are column vectors, Q and A are matrices, and the superscript T denotes the transpose. The q_{ij} (elements of Q) are given constants such that $q_{ij} = q_{ji}$ (which is the reason for the factor of ½ in the objective function). By performing the indicator vector and matrix multiplication, the objective function then is expressed in terms of these q_{ij}, the c_j (elements of C) and the variables as follows:

$$f(x) = Cx - \frac{1}{2} x^T Qx = \sum_{j=1}^{n} c_j x_j - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j \quad \ldots\ldots \quad 2.14$$

If i=j in this double summation, then $x_i x_j = x_j^2$, so – ½ q_{ij} is the coefficient of x_j^2. If i ≠ j, then $-\frac{1}{2}\left(q_{ij}x_i x_j + q_{ji}x_i x_j\right) = -q_{ij}x_i x_j$, so – q_{ij} is the total coefficient for the product of x_i and x_j,

Consider the following quadratic programming problem given as example 2.2 below:

Example 2.2

Maximum $f(x_1, x_2) = 15x_1 + 30x_2 + 4x_1x_2 - 2x_1^2 - 4x_2^2$

Subject to

$$x_1 + 2x_2 \leq 30$$

And $x_1 \geq 0$, $x_2 \geq 0$

In this case

$$C = [15 \quad 30],\ x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \qquad Q = \begin{bmatrix} 4 & -4 \\ -4 & 8 \end{bmatrix}$$

A = [1 2], b = [30]

Note that

$$x^T Q\, x = [x_1 \ \ x_2] \begin{bmatrix} 4 & -4 \\ -4 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= [(4x_1 - 4x_2) \qquad (-4x_1 + 8x_2)]' \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$= 4x_1^2 - 4x_2x_1 - 4x_1x_2 + 8x_2^2$$

$$= q_{11}x_1^2 + q_{21}x_2x_1 + q_{12}x_1x_2 + q_{22}x_2^2$$

multiplying through by – ½ gives

$$-\frac{1}{2}x'Qx = -2x_1^2 + 4x_1x_2 - 4x_2^2,$$

which is the nonlinear portion of the objective function for this example.

For this project, the modified simplex method will be used to solve this quadratic programming problem.

2.6 THE MODIFIED SIMPLEX METHOD

The modified simplex method exploits the key fact, with the exception of the complementarity constraint, the complementarity simply implies that it is not permissible for both complementarity variables of any pair to be (non degenerate) basic variables (the only variables >0) when (non-degenerate) BF solutions are considered. Therefore the problem reduces to finding an initial BF solution to any linear programming problem that has these constraints, subject to this additional restriction on the identity of the basic variables. (This initial BF solution may be the only feasible solution in this case).

Finding such an initial BF solution relatively straight forward. In the simple case $C^T \leq 0$ (unlikely) and $b \geq 0$, the initial basic variables are elements of y and v (multiple through the first set of equation by -1) so that the desired solution is $x=0$, $u=0$, $y=-C^T$, $v=b$. Otherwise, you need to revise the problem by introducing an artificial variable into each of the equations where $C_j > 0$ (add the variable on the left) or $b_i < 0$ (subtract the variable on the left and then multiply through by -1), in order to use these artificial variables (call them z_i, z_2, and so on) initial basic variables for the revised problem. (Note that this choice of initial basic variables satisfies the complimentarily constraint, because as nonbasic variables $x = 0$ and $u = 0$ automatically).

Next, use phase 1 of the two-phase method to find a BF solution for real problem: i.e., apply the simplex method (with one modification) to the following linear programming problem

$$\text{Minimize } z = \sum_j z_j \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 2.15$$

Subject to the linear programming constraints obtained from the Kuhn Tucker conditions, but with these artificial variables included.

The one modification on the simplex method is the following change in the procedure for selecting an entering basic variable.

2.7 RESTRICTED ENTRY RULE

When you are choosing an entering basic variable, exclude from consideration any nonbasic variable whose complimentary variable already is a basic variable; the choice should be made from the other nonbasic variables according to the usual criterion for the simplex method.

This rule keeps the complimentarily constraint satisfied throughout the course of the algorithm. When an optional solution

$$x^*, u^*, y^*, v^* z_, = 0 \ldots, z_n = 0$$

is obtained for the phase 1 problem, x^* is the desired optional solution for the original quadratic programming problem. Phase 2 of the two-phase method is not needed.

We shall now illustrate this approach using example 2.1. It will be noted that $f(x_1, x_2)$ is strictly concave, that is

$$Q = \begin{bmatrix} 4 & -4 \\ -4 & 8 \end{bmatrix}$$

is positive definite, so the algorithm can be applied.

After the needed artificial variables are introduced, the linear programming problem to be addressed explicitly by the modified simplex method then is

Minimize $z = z_1 + z_2$,

Subject to

$$
\begin{aligned}
4x_1 - 4x_2 + u_1 - y_1 \qquad\qquad & +z_1 = 15 \\
-4x_1 + 8x_2 + 2u_1 \qquad -y_2 \quad & +z_2 = 30 \\
x_1 + 2x_2 \qquad\qquad\qquad & + v_1 = 30
\end{aligned}
$$

and

$$x_1 \geq 0, x_2 \geq 0, u_1 \geq 0, y_1 \geq 0, y_2 \geq 0, v_1 \geq 0, z_1 \geq 0, z_2 \geq 0$$

The addition complementarity's constraint

$$x_1y_1 + x_2y_2 + u_1v_1 = 0$$

Is not included explicitly, because the algorithm automatically enforces this constraint because of the restricted – entry rule. In particular, for each of the three pairs of complementary variables- (x_1, y_1), (x_2, y_2), (u_1, v_1), and (u_2, v_2) whenever one of the two variables already is a basic variable, the other variable is excluded as a candidate for the entering basic variables. Remember that the only non-zero variables are basic variables. Because the initial set of basic variables for the linear programming problem – z_1,z_2,v_1 – gives an initial BF solution that satisfies the complementarity constraint, there is no way that this constraint can be violated by any subsequent BF solution.

Table 2.2.1 – 2.2.4 shows the results of applying the modified simplex method to this problem. The first simplex tableau exhibits the initial system of equations after converting from minimizing z to maximizing –z and algebraically eliminating the initial basic variables from row 4. The three iterations proceed just for the regular simplex method except for eliminating certain candidates for the entering basic variable because of the restricted entry-rule.

	x_1	x_2	u_1	y_1	y_2	v_1	z_1	z_2	b
z_1	4	-4	1	-1	0	0	1	0	15
z_2	-4	8	2	0	-1	0	0	1	30
v_1	1	2	0	0	0	1	0	0	30
z	0	-4	-3	1	1	0	0	0	-45

In the first tableau, u_1 is eliminated as a candidate its complementary variable (v_1) already is a basic variable (but x_2 would have been chosen anyway because –4 < -3).

	x_1	x_2	u_1	y_1	y_2	v_1	z_1	z_2	$\underset{\sim}{b}$
z_1	2	0	2	-1	- ½	0	1	½	30
x_2	-½	1	¼	0	-1/8	0	0	1/8	3 ¾
v_1	2	0	- ½	0	¼	1	0	- ¼	22 ½
z	-2	0	-2	1	½	0	0	½	-30

In the second tableau, both u_1 and y_2 are eliminated as candidates (because v_1 and x_2 are basic variables), so x_1 automatically is chosen as the only candidate with a negative coefficient in row 4 (whereas the regular simplex method would have permitted choosing either x_1 or u_1 because they are tied for the largest negative coefficient).

	x_1	x_2	u_1	y_1	y_2	v_1	z_1	z_2	$\underset{\sim}{b}$
z_1	0	0	5/2	-1	- ¾	-1	1	¾	7 ½
x_2	0	1	1/8	0	-1/6	¼	0	1/16	9 ³/8
x_1	1	0	- ¼	0	1/8	½	0	-1/8	11 ¼
z	0	0	-5/2	1	¾	1	0	¼	-7 ½

In the third tableau, both y_1 and y_2 are eliminated (because x_1 and x_2 basic variables). However, u_1 is not eliminated because v_1 no longer a basic variable, so u_1 is chosen as the entering basic variable in the usual way.

	x_1	x_2	u_1	y_1	y_2	v_1	z_1	z_2	b
z_1	0	0	1	-2/5	-3/10	-2/5	2/5	3/10	3
x_2	0	1	0	1/20	-1/40	3/10	-1/20	1/40	9
x_1	1	0	0	-1/10	1/20	2/5	1/10	-1/20	12
z	0	0	0	0	0	0	1	1	0

The resulting optional solution for this phase 1 problem is $x_1 = 12$, $x_2 = 9$, $u_1 = 3$, with the rest of the variables zero. Therefore, the optional solution for the quadratic programming is $(x_1,x_2)=(12,9)$.

2.8 REMARKS

It was observed that separable and quadratic programmings gave a good account of themselves, and both can be used as search-light for finding the global optimum. It was also observed that the more the variables, the more the number of λ's to be considered in the case of separable programming and the more the work/computational time one needs. The above comments are based on the examples 2.1 and 2.2.

CHAPTER THREE

COMPUTER OPTIMIZATION METHOD

3.1 INTRODUCTION

In calculus we learnt how to obtain the minimum and the maximum of a function by setting derivative equal to zero. Unfortunately minimization or maximization (optimization) problems encountered in industry are not that simple. Usually, optimization should take place while satisfying a number of constraints imposed on the system. In case where the constraints and the function to be optimized are expressed analytically, the Lagrangian method of undetermined multipliers can be used to obtain the optimum solution.

3.2 SOLUTION BY LAGRANGIAN MULTIPLIER'S TECHNIQUE

3.2.1 LAGRANGIAN MULTIPLIERS AND EQUALITY CONSTRAINED PROBLEMS

Before investigating the general nonlinear programming problem, it is necessary to first introduce the method of Lagrangian multipliers for solving the equality – constrained mathematical programming problem. The problem is specified as

Maximum $z = f(x) = f(x_1, x_2, \ldots, x_n)$ 3.1

Subject to $g_i(x) = gi(x_1, x_2, \ldots, x_n) = b_i$ 3.2

The method of Lagrangian multipliers has been introduced into 3.1 and 3.2 as follows:

$$F(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i [b_i - g_i(x)] \quad \ldots\ldots\ldots\ldots\ldots \quad 3.3$$

The necessary conditions for a point $[x^*,\lambda^*]^1 = [x^*_1, x^*_2,\ldots,x^*_n, \lambda^*_1, \lambda^*_2, \lambda^*_m]$ to maximize $F(x, \lambda]$ are, from theorem 1.1,

$$\frac{\partial F(x,\lambda)}{\partial x_j} = \frac{\partial f(x)}{\partial x_j} - \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(x)}{\partial x_j} = 0 \qquad j=1,2,\ldots,n \quad \ldots\ldots\ldots\ldots \quad 3.4$$

$$\frac{\partial F(x,\lambda)}{\partial \lambda_i} = b_i - g_i(x) = 0 \qquad I=1,2,..,m \quad \ldots\ldots\ldots\ldots \quad 3.5$$

In the case where $f(x)$, $g_1(x),\ldots g_m(x)$ are linear in x, if point x^* satisfies (3.4) and 3.5), it is the maximizing point to the corresponding linear programming problem 3.1 and 3.2 where (x), $g_1(x),\ldots g_m(x)$ are linear functions. For the general nonlinear programming problem, it will now be shown why a solution x* to 3.4 and 3.5, which is a local maximum of F(x, λ) in (3.3), is also a local maximum for 3.1 and 3.2. To demonstrate this result, first assume that n=2 and m=1 so that the (3.1) and (3.2) problem is

Maximize $z=f(x_1,x_2)$ 3.6

Subject to $g1(x_1,x_2)=b_1$ 3.7

If the condition of the implicity function theorem are satisfied, it must be possible to write x_1 in terms of x_2 so that $x_1 = \theta_1 (x_1)$. The theorem then guarantees that $\theta_1 (x_1)$ is differentiable. The objective function, can be written using $\theta_1 (x_1)$ as a univariate function in x_1 and the (3.6) and (3.7) problem is equivalent to the unconstrained problem

Maximize $z = f[x_1 = \theta_1 (x_1)]$

The necessary condition for x_1^0 to be a local optimum of $f[x_1 = \theta_1 (x_1)]$ is

$$\frac{df[x_1 = \theta_1 (x_1)]}{d x_1} = 0$$

But recall from differential calculus that the total derivative d/dx_1 of $f(x_1 x_2)$ can be written as

$$\frac{df(x_1,x_2)}{dx_i} = \frac{\partial f(x_1,x_2)}{\partial x_1} + \frac{\partial f(x_1,x_2)}{\partial x_2}\frac{dx_2}{dx_1} \quad \dots\dots\dots\dots\dots\dots \quad 3.8$$

But $x_2 = \theta_1(x_1)$. If $\theta_1(x_1)$ is substituted for x_2 in 3.8 and the total derivative df/dx_1 is evaluated at (x_1^0, x_2^0),

$$\frac{d}{dx_1} f(x_1^0, x_2^0) = \frac{\partial f(x_1^0, x_2^0)}{\partial x_1} + \frac{\partial f(x_1^0, x_2^0)}{\partial x_2} o \frac{d\theta_1(x_1)}{dx_1} = 0 \qquad 3.9$$

since $g_1(x_1,x_2) = b_1$,

$$\frac{d}{dx_1} g_1(x_1, x_2) = \frac{\partial g_1(x_1, x_2)}{\partial x_1} + \frac{\partial g_1(x_1, x_2)}{\partial x_2} \cdot \frac{d\theta_1(x_1)}{dx_1} = 0 \qquad 3.10$$

where $\theta_1(x_1)$ has been substituted for x_1 in the last term. From (3.10)

$$\frac{d\theta_1(x_1)}{dx_1} = - \frac{\partial g_1(x_1, x_2)}{\partial x_1} / \frac{\partial g(x_1, x_2)}{\partial x_2} \qquad 3.11$$

Now substitute the right hand side of 3.11 for $d\theta_1(x_1)/dx_1$ in (3.9) where $d\theta_1(x_1)/dx_1$ is evaluated at (x_1^o, x_2^o). Then

$$\frac{\partial f(x_1^o, x_1^o)}{\partial x_1} - \frac{\partial f(x_1^o, x_1^o)}{\partial x_2} \left[\frac{\partial g_1(x_1^o, x_1^o)}{\partial x_1} \Bigg/ \frac{\partial g_1(x_1^o, x_2^o)}{\partial x_2} \right] = 0 \qquad 3.12$$

and define λ_1 as

$$\lambda_1 = \frac{\partial f(x_1^o, x_2^o)}{\partial x_2} \Bigg/ \frac{\partial g_1(x_1^o, x_2^o)}{\partial x_2}$$

Then (3.12) can be written as

$$\frac{\partial f(x_1^o, x_2^o)}{\partial x_1} - \lambda_1 \frac{\partial g_1(x_1^o, x_2^o)}{\partial x_2} = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 3.13$$

Directly from the definition of λ_1 it follows that

$$\frac{\partial f(x_1^o, x_2^o)}{\partial x_2} - \lambda_1 \frac{\partial g_1(x_1^o, x_2^o)}{\partial x_2} = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 3.14$$

Additionally, (x_1^o, x_2^o) must satisfy

$$g_1(x_1^o, x_2^o) = b_1 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 3.15$$

Therefore, by using the implicit function theorem, it is possible to write the necessary conditions for determining a local maximum to (3.6) and (3.7) in the form (3.14) (3.15).

Now consider the Lagrangian function corresponding to (3.6) and (3.7):

$$F(x,\lambda) = F(x_1, x_2, \lambda_1) = f(x_1, x_2) + [b_1 - g_1(x_1, x_2)]$$

The necessary conditions for maximizing $F(x,\lambda)$ are, from theorem on relative (local) maximum

$$\frac{\partial F(x,\lambda)}{\partial x_1} = \frac{\partial f(x_1, x_2)}{\partial x_1} - \lambda_1 \frac{\partial g_1(x_1, x_2)}{\partial x_1} = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 3.16$$

$$\frac{\partial F(x,\lambda)}{\partial x_2} = \frac{\partial f(x_1, x_2)}{\partial x_2} - \lambda_1 \frac{\partial g_1(x_1, x_2)}{\partial x_2} = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad 3.17$$

$$\frac{\partial F(x,\lambda)}{\partial x_1} = b_1 - g_1(x_1,x_2) = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 3.18$$

The necessary conditions for a point x^o to maximize $F(x,\lambda)$, given by 3.16 – 3.18 are identical to the necessary conditions for the equality-constrained problem in (3.6) and (3.7).

It is possible to extend the above argument from the n=2 case to the general n-variate case to show that the necessary conditions to maximize the Lagrangian function $F(x,\lambda)$ in 3.3 are equality-constrained problem 3.1 and 3.2. Before doing this, it is first necessary to modify the definitions of a local and a global maximum in the presence of constraints.

Definition 3.1

Global maximum (constrained problem). The function f(x) is said to take on its global maximum at the point x* if $f(x) \leq f(x^*)$ for all x (including x*) that belong to the feasible set of points x, where the set x represents the constraint region.

In the equality-constrained problem for example, x belongs to X if x satisfies $g_i(x) = b_i$, $i=1,2,\ldots,m$.

Definition 3.2

Local maximum (constrained problem). The function f(x) is said to take on a local maximum at x^o if x^o belongs to X and there exists an $\in > 0$ such that for every $x \neq x^o$ that belongs to X and is in an $\in$-neighbourhood of x^o , $f(x) \leq f(x^o)$.

Now, suppose that F(x) takes on a local maximum for the equality-constrained set of feasible points, X, at x^o. Furthermore, assume that at x^o the conditions of the implicit function theorem are satisfied so that the rank of G, denoted by r(G), is m. Then for $\hat{x}^{o'} = \left[x^o_{m+1}, x^o_{m+2}, \ldots, x^o_n\right]$, there exist in functions $\theta_i(\hat{x}^o)$, such that

$$x_i = \theta_i(\hat{x}^o) \quad i=1,2,\ldots,m \qquad 3.19$$

Now consider the total differentials of f(x) and $g_i(x)$:

$$df(x) = \sum_{j=1}^{n} \frac{\partial f(x)}{\partial x_j} dx_j \qquad 3.20$$

$$dg_i(x) = \sum_{j=1}^{n} \frac{\partial g_i(x)}{\partial x_j} dx_j \quad i=1,2,\ldots,m \qquad 3.21$$

Since $\partial f(x^o) / \partial x_j = 0$, j=1,2,...,n, if x^o is a stationary point, 3.20 may be written as

$$\sum_{j=1}^{n} \frac{\partial f(x^o)}{\partial x_j} dx_j = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad 3.22$$

Additionally, since $g_i(x)=b_i$, i=1,2,…,m 3.21 may be written as

$$\sum_{j=1}^{n} \frac{\partial g_i(x)}{\partial x_j} dx_j = 0 \qquad i=1,2,\ldots,m \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 3.23$$

From the rules for differentiating compound functions,

$$\frac{\partial h(x)}{\partial x_j} = \sum_{i=1}^{m} \frac{\partial f(x)}{\partial x_i} \frac{\partial \theta_i(x)}{\partial x_j} + \frac{\partial f(x)}{\partial x_j} \qquad j=m+1,m+2,\ldots,n \quad 3.24$$

where $h(x) = f\left(\theta_1(\hat{x}),\ldots,\theta m(\hat{x}^o),\hat{x}^o\right)$. It is now possible to proceed as has been done for the two-dimensional case. Identify an expression for $\partial\theta_i(\hat{x})/\partial x_j$ in terms of the partial derivatives ${\partial g_i(x)}/{\partial x_j}$ and substitute this expression for $\partial\theta_i(\hat{x})/\partial x_j$ in (3.24). However, it is not possible to arrive at the desired result by solving for $\partial\theta_i(\hat{x})/\partial x_j$ directly. Introduce the Lagrangian multiplier λ_i and write

$$df(x) - \sum_{i=1}^{m} \lambda_i dg_i(x) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \qquad 3.25$$

At the point x^o, it follows from (3.22) and (3.23) that (3.25) may b written as

$$\sum_{j=1}^{n}\left[\frac{\partial f(x^o)}{\partial x_j}-\sum_{i=1}^{m}\frac{\partial g_i(x^o)}{\partial x_j}\right]dx_j = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 3.26$$

Since the first m variables can be expressed in terms of the remaining n-m by (3.19), the set of n-m variables may be though of as independent variables in (3.19). This dx_j, j=m+1, m+2,…,n, may be considered as independent variables, and if (3.26) is satisfied, it must follow that

$$\frac{\partial f(x^o)}{\partial x_j}-\sum_{i=1}^{m}\lambda_i\frac{\partial g_i(x^o)}{\partial x_j}\equiv 0 \quad j=m+1, m+2,\ldots,n \ldots. \quad 3.27$$

Now (3.26) can be rewritten excluding the components in the sum for j=m+1, m+2,…,n, since by (3.27) that are zero:

$$\sum_{j=1}^{m}\left[\frac{\partial f(x^o)}{\partial x_j}-\sum_{i=1}^{m}\frac{\partial g_i(x^o)}{\partial x_j}\right]dx_j = 0 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 3.28$$

Since the dx_j, j=1,2,…,m are the dependent variables determined uniquely by dx_j, j=m+1, m+2,…,n in (3.19) coefficients of dx_j, j=1,2,…,m, in (3.28) must be identically zero. Thus

$$\frac{\partial f(x^o)}{\partial x_j}-\sum_{i=1}^{m}\lambda_i\frac{\partial g_i(x^o)}{\partial x_j}\equiv 0 \quad j=1,2,\ldots,m \ldots\ldots\ldots \quad 3.29$$

Combining (3.27) and (3.29), it is seen that the following condition must be satisfied for x^o:

$$\frac{\partial f(x^o)}{\partial x_j} - \sum_{i=1}^{m} \lambda_i \frac{\partial g_i(x^o)}{\partial x_j} = 0 \quad j=1,2,\ldots,m \ldots\ldots\ldots \quad 3.30$$

Additionally,

$$g_i(x^o) - b_i = 0 \qquad i=1,2,\ldots,m \ldots\ldots\ldots \quad 3.31$$

It is seen that these conditions in 3.30 and 3.31 are identical to the necessary conditions for maximizing the Lagrangian function $F(x,\lambda)$ given by (3.4) and (3.5).

In summary, the necessary conditions for x^o to be a local maximum to the equality – constrained problem

Maximum $Z = f(x) = f(x_1, x_2, \ldots x_n)$

Subject to $g_i(x) = b_i \qquad i=1,2,\ldots,m$

Can be generated by defining the Lagrangian function

$$F(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \left[b_i - g_i(x)\right] \ldots\ldots\ldots\ldots\ldots\ldots 3.34$$

These condition for x^o to be maximizing point to (3.34) are

$$\frac{\partial F(x^o,\lambda^o)}{\partial x_j} = \frac{\partial f(x^o)}{\partial x_j} - \sum_{i=1}^{m} \lambda_i^o \frac{\partial g_i(x^o)}{\partial x_j} = 0 \quad j=1,2,\ldots,n \ldots\ldots \quad 3.35$$

$$\frac{\partial F(x^o,\lambda^o)}{\partial x_j} = g_i(x^o) - b_i = 0 \qquad i=1,2,\ldots,m \ \ldots. \qquad 3.36$$

If r(G) = m at x^o, then (3.35) and (3.36) will also be the necessary condition for x^o to be local maximum to (3.32) and (3.33)

We shall present example 3.1, as an illustration:

Example 3.1

Maximum $z = f(x) = f(x_1,x_2,x_3) = x_1x_2x_3$

Subject to $x_1^2 + x_2^2 + x_3^2 = 27$

The Lagrangian function is

$$F(x,\lambda) = x_1x_2x_3 + \lambda_1(27 - x_1^2 - x_2^2 - x_3^2)$$

and the four equations resulting from (3.4) and (3.5) are

$$\left.\begin{array}{l} x_2x_3 - 2\lambda_1x_1 = 0 \\ x_1x_3 - 2\lambda_1x_2 = 0 \\ x_1x_2 - 2\lambda_1x_3 = 0 \\ 27 - x_1^2 - x_2^2 - x_3^2 = 0 \end{array}\right. \quad \text{———} \quad (2)$$

There are eight possible solutions to this set of equations, which are the combination of $x_1 = x_2 = x_3 = \pm 3$ with $\lambda_1 = 3/2, \lambda_1 = -3/2$ in four solutions each.

The maximum of f(x) is not unique; four points in the variable space of x_1,x_2, and x_3 (3,3,3), 3,-3,-3), (-3,3,-3), and (-3,-3,3) will give the maximum value of f(x), which is 27. The other four points (-3,-3,-3), (3,3,-3), (-3,3,3), and (3,-3,3) will give the minimum of f(x) which is -27 on the shell of the sphere delineated by equations (1).

3.3 Behaviour of the functions at the critical point x*. It is necessary to investigate the behaviour of the functions f, $g_1, \ldots, g_m$ at the critical point x* in a more general way than has been presented above. Denote the ∇f and ∇g_1 the column gradient vectors associated with the functions f(x) and $g_1(x)$, where

$$\nabla f^{1} = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}\right]$$

and

$$\nabla g_i^{1} = \left[\frac{\partial g_i}{\partial x_1}, \frac{\partial g_i}{\partial x_2}, \ldots, \frac{\partial g_i}{\partial x_n}\right]$$

and define the (m+1) x n matrix G^o and the m x n matrix G by

$$G^o = \begin{bmatrix} \nabla g'_1 \\ \cdot \\ \cdot \\ \cdot \\ \nabla g'_m \\ \nabla f' \end{bmatrix} \qquad G = \begin{bmatrix} \nabla g'_1 \\ \cdot \\ \cdot \\ \cdot \\ \nabla g'_m \end{bmatrix} \qquad \ldots\ldots \quad 3.37$$

respectively

The Lagrangian function can be written in a more general form:

$$F^{o}(x,\lambda) = \lambda_{o} f(x) + \sum_{i=1}^{m} \lambda_{i}[b_{i} - g_{i}(x)]$$

where λ_0 is either 0 or 1. If f(x) takes on a local maximum (or minimum) at x*, then x* must satisfy

$$\frac{\partial F^{o}(x,\lambda)}{\partial x_{j}} = \lambda_{o}\frac{\partial f}{\partial x_{j}} - \sum_{i=1}^{m} \lambda_{i}\frac{\partial g_{i}(x)}{\partial x_{j}} = 0 \quad j=1,2,\ldots,n \quad .. 3.38$$

and

$$\frac{\partial F^{o}(x,\lambda)}{\partial x_{j}} = b_{i} - g_{i}(x) = 0 \quad i=1,2,\ldots,m \quad \ldots\ldots.. 3.39$$

where $\lambda_o = 0$ or 1.

Notice that if $\lambda_o = 1$, the result are the usual Lagrangian necessary condition given in (3.35) and (3.36). In the special cases where x* will not satisfy (3.35) and (3.36), x* will satisfy (3.38) and (3.39) when $\lambda_o = 0$.

3.3.1 SADDLE POINT

The necessary conditions for (x*, λ*) to be an optimizing point also are necessary for the Lagrangian function to have a saddle point at

(x^*, λ^*). By a saddle point, it is meant that $F(x, \lambda)$ is a maximum with respect to x and minimum with respect to λ:

$$F(x, \lambda^*) \leq F(x^*, \lambda^*) \leq F(x^*, \lambda)$$

and $[x^*, \lambda^*)$ is a global saddle point if

$$\operatorname*{Sup}_{\lambda} F(x, \lambda^*) = F(x^*, \lambda^*) = \operatorname*{Inf}_{\lambda} F(x^*, \lambda)$$

Note that global saddle point will also be sufficient if f(x) is a concave function and the constraints $g_i(x) = b_i$ $i=1,2,3,\ldots,m$ form a convex set.

3.4 COMPUTATIONAL ALGORITHM

The Lagrangian procedure outlines in the development of the Kuhn – Tucker conditions may be directly applied in a computational algorithm that will guarantee the global maximum of the nonlinear programming problem.

The Lagrangian solution method involves the following steps:

Step I

Find the unconstrained maximum of f(x). Frequently, by inspecting the function, it is apparent that the unconstrained maximum will not be feasible, so that this step may be deleted. If this solution is feasible, it will be the global maximum,'and there is no need to proceed to the steps.

Step II

Solve the Lagrangian function based only on the m – s equality constraints $g_i(x) = b_i$, $i = s + 1,\ldots,m$. If this solution satisfies the remaining constraints, it will be the global maximum of f(x) and the process may be stopped.

Step III

Add one of the inequality constraints to the Lagrangian function in Step II treating it as if it were active. Solve this Lagrangian system. If the solution satisfies the remaining s-1 constraints, stop. Otherwise, drop the current inequality constraints fail to yield a feasible solution when treats individually as equality constraints, proceed to step IV.

Step IV

Repeat the process by now adjoining pairs of inequality constraints to the Lagrangian function in step II, treating them as active constraints. Continue until a feasible solution to the s-2 remaining constraints is encountered or all $C_2^s = s!/2!(s-2)!$ pairs have been exhausted. If the latter occurs, proceed to Step V.

Step V

Continue the process taking all C_a^s combinations for a = 3,4,…s until a feasible solution is encountered.

In what follows we shall present example 3.2 as an illustration of Lagrangian algorithm

The problem is

$$\text{Maximum } z = f(x_1,x_2) = -(x_1-11)^2 - 4(x_2-6)^2$$

$$\text{Subject to} \quad \begin{aligned} 2x_1 + x_2 &\le 18 \\ x_1 + 2x_2 &\le 16 \\ x_1, x_2 &\ge 0 \end{aligned}$$

(R. C. Pfaffenberger, D. A. Walker)

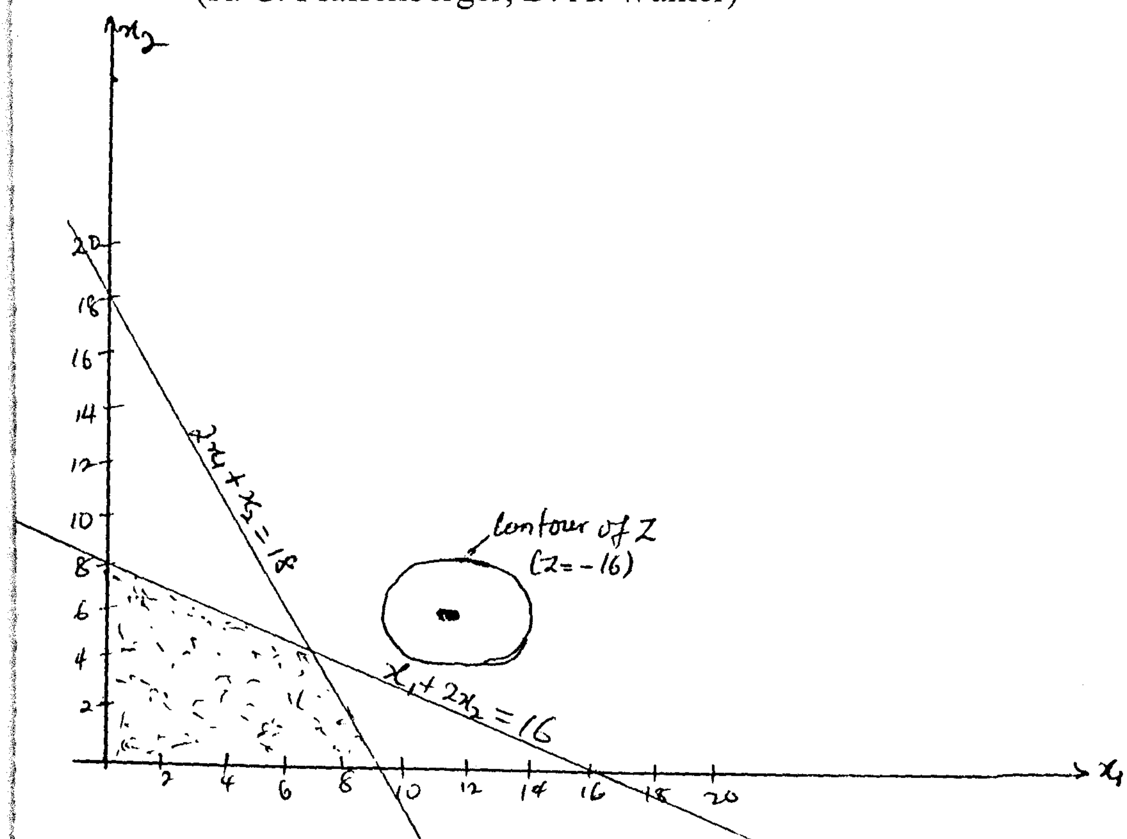


Fig. 3.1 Graphic solution to example 3.1

From the graphic representation of the problem given in fig 3.1, it is apparent that the solution occurs at the intersection of the two lines $2x_1 + x_2 = 18$ and $x_1 + 2x_2 = 16$. The Lagrangian algorithm will now be applied to verify this conjecture and to illustrate the technique.

Step I

The unconstrained global maximum from inspection of $f(x_1,x_2)$ occurs at (11,6), which clearly is not feasible.

Step II

From the Lagrangian function using the constraint $x_1 + 2x_2 = 16$.

$$
\begin{aligned}
&F_1(x,\lambda) = -(x_1 - 11)^2 - 4(x_2 - 6)^2 + \lambda_1(16 - x_1 - 2x_2) \\
&\frac{\partial F_1(x,\lambda)}{\partial x_1} = -2(x_1 - 11) - \lambda_1 = 0 \\
&\frac{\partial F_1(x,\lambda)}{\partial x_2} = -8(x_2 - 6) - 2\lambda_1 = 0 \qquad ..(1) \\
&\frac{\partial F_1(x,\lambda)}{\partial \lambda_1} = 16 - x_1 - 2x_2 = 0
\end{aligned}
$$

The solution to (1) is $x_1 = 7.5$, $x_2 = 4.25$, $\lambda_1 = 7$, and $\lambda_2 = 0$.

Since the point (7.5, 4.25) is not feasible, the process continues.

Step III

Form the Lagrangian function using the constraint $2x_1 + x_2 = 18$.

$$F_2(x,\lambda) = -(x_1 - 11)^2 - 4(x_2 - 6)^2 + \lambda_2(18 - 2x_1 - x_2)$$

$$\left.\begin{aligned}
\frac{\partial F_2(x,\lambda)}{\partial x_1} &= -2(x_1 - 11) - 2\lambda_2 = 0 \\
\frac{\partial F_2(x,\lambda)}{\partial x_2} &= -8(x_2 - 6) - \lambda_2 = 0 \\
\frac{\partial F_2(x,\lambda)}{\partial \lambda_2} &= 18 - 2x_1 - x_2 = 0
\end{aligned}\right] \quad (2)$$

The solution to (2) is $x_1 = 6.3$, $x_2 = 5.4$, λ_1 0, and $\lambda_2 = 4.8$.

Since (6.3, 5.4) also is not feasible, the process continues.

Step IV

Form the Lagrangian function using both constraints $x_1 + 2x_2 = 16$ and $2x_1 + x_2 = 18$.

$$F_3(x,\lambda) = -(x_1 - 11)^2 - 4(x_2 - 6)^2 + \lambda_1(16 - x_1 - 2x_2) + \lambda_2(18 - 2x_1 - x_2)$$

$$\left.\begin{aligned}
\frac{\partial F_3(x,\lambda)}{\partial x_1} &= -2(x_1 - 11) - \lambda_1 - 2\lambda_2 = 0 \\
\frac{\partial F_3(x,\lambda)}{\partial x_2} &= -8(x_2 - 6) - 2\lambda_1 - \lambda_2 = 0 \\
\frac{\partial F_3(x,\lambda)}{\partial \lambda_1} &= 16 - x_1 - 2x_2 = 0 \\
\frac{\partial F_3(x,\lambda)}{\partial \lambda_2} &= 18 - 2x_1 - x_2 = 0
\end{aligned}\right] \quad (3)$$

The solution to (3) is $x_1 = 6.67$, $x_2 = 4.67$, $\lambda_1 = 4.2$ and λ_1=2.23. The point (6.67, 4.67) is the intersection of the two lines and hence is feasible. Therefore, the global maximum occurs at the point

$x^* = (6.67, 4.67)$, and $f(x^*)$ is –25.82.

Notice that the nonnegative constraints have been ignored, although technically they should have been incorporated in the solution process. However, figure 3.1 illustrates that they will not participate in the global maximization of f(x).

In the above problem if it is desired to minimize $f(x_1,x_2)$ rather than maximize this function, the nonnegative constraints obviously would now play an important role. Indeed, by inspecting the minimizing point (x_1,x_2) would be (0,0).

We shall present example 3.3 as an illustration of Lagrangian algorithm

Example 3.3

The problem is

$$\text{Maximum } z = f(x_1,x_2) = -(x_1-4)^2-(x_2-4)^2$$

$$\text{Subject to } \begin{array}{l} x_1 + x_2 \leq 4 \\ x_1^2 + x_2^2 = 4 \\ x_1, x_2 \geq 0 \end{array}$$

(Same as above)

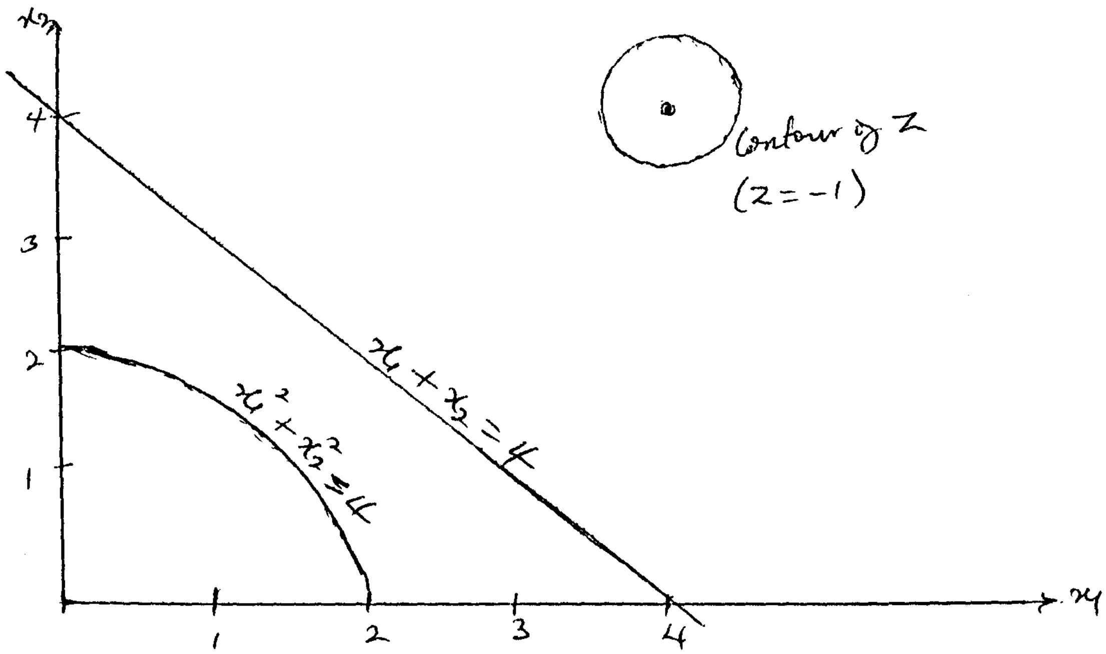


Fig. 3.2 graphic Solution to Example 3.3

The Lagrangian method will be used to solve this problem also.

Step I

The unconstrained maximum from inspection of $f(x_1,x_2)$ occurs at (4,4), which is not feasible.

Step II

From the Lagrangian function using the constraint $x_1 + x_2 = 4$

$$F_1(x,\lambda) = -(x_1-4)^2 - (x_2-4)^2 + \lambda_1(4 - x_1 - x_2)$$

$$\left.\begin{aligned}
\frac{\partial F_1(x,\lambda)}{\partial x_1} &= -2(x_1-4) - \lambda_1 = 0 \\
\frac{\partial F_1(x,\lambda)}{\partial x_2} &= -2(x_2-4) - \lambda_1 = 0 \\
\frac{\partial F_1(x,\lambda)}{\partial \lambda_1} &= 4 - x_1 - x_2 = 0
\end{aligned}\right] \qquad (1)$$

The solution to (1) is $x_1 = x_2 = 2,\ \lambda_1 = 4$, and $\lambda_2 = 0$

Since (2,2) is not feasible, the process continues.

Step III.

From the Lagrangian function using the constraint $x_1^2 + x_2^2 = 4$.

$$F_2(x,\lambda) = -(x_1 - 4)^2 - (x_2 - 4)^2 + \lambda_2(4 - x_1^2 - x_2^2)$$

$$\left.\begin{aligned} \frac{\partial F_2(x,\lambda)}{\partial x_1} &= -2(x_1 - 4) - 2x_1\lambda_2 = 0 \\ \frac{\partial F_2(x,\lambda)}{\partial x_2} &= -2(x_2 - 4) - 2x_2\lambda_2 = 0 \\ \frac{\partial F_2(x,\lambda)}{\partial \lambda_2} &= 4 - x_1^2 - x_2^2 = 0 \end{aligned}\right\} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(2)$$

The solution to (2) is $x_1 = x_2 = \sqrt{2},\ \lambda_1 = 0$, and $\lambda_2 = 1.83$

Since ($\sqrt{2},\ \sqrt{2},$) is feasible, the maximum is f ($\sqrt{2},\ \sqrt{2},$).

From the graphic representation of the problem displayed in Fig. 3.2,
It is clear that this point is the global maximum.

3.5 REMARKS

It was observed that the Lagrangian multiplier's method gives the best optimum value for nonlinear problems and also gives the global optimum value all the time. And it is therefore the best method among the three considered in this research/work. We therefore recommended that a program be written for Largrangian multiplier's method, which would now form the appendix at the end of this project. The output of this appendix (code) will form part of chapter four. The above remarks are based on the three examples considered in this chapter.

CHAPTER FOUR

COMPUTER TECHNIQUE FOR LAGRANGIAN MULTIPLIER'S METHOD

4.1 INTRODUCTION

We considered the output of a computer code (Lagrangian code) in this chapter. Here, we used the code to solve a particular example in chapter 3, that is, Example 3.2 and we are able to see that the code worked perfectly well for this particular example. It can therefore be use for other problems as well. In this chapter, we have psuedocode and flowchart representations, which tend to simplify the working of the Lagrangian code.

4.2 PSUEDOCODE REPRESENTATION

STEP I

INPUT OBJECTIVE FUNCTION F(x) AND CONSTRAINTS gi

i=1, 2, 3, …, m. THEN

STEP II

ADD OBJECTIVE FN f(x) and CONSTRAINTS gi TOGETHER TO

GIVE $F(x,\lambda)$, $x_i = x_1, x_2, \dots x_n$; $\lambda_i, = \lambda_1, \lambda_2, \dots \lambda_m$

STEP III

IF NO CONSTRAINT(S) GIVEN, THEN FIND PARTIAL DERIVATIVES OF OBJECTIVE FN, F(x), w.r.t x, OTHERWISE FIND PARTIAL DERIVATIVES OF F = f + $\sum_{i=1}^{m} \lambda_i g_i$ w.r.t x, λ and EQUATE BOTH THE DIFFERENTIALS TO ZEROS.

STEP IV

CONVERT THE DIFFERENTIALS RESULTING FROM III ABOVE TO MATRIX, AX=B

STEP V

PERFORM ROW OPERATION (GAUSS-JORDAN METHOD) ON IV ABOVE

STEP VI

EVALUATE THE DECISION VARIABLE x_i, i=1,2,…n FROM STEP V ABOVE

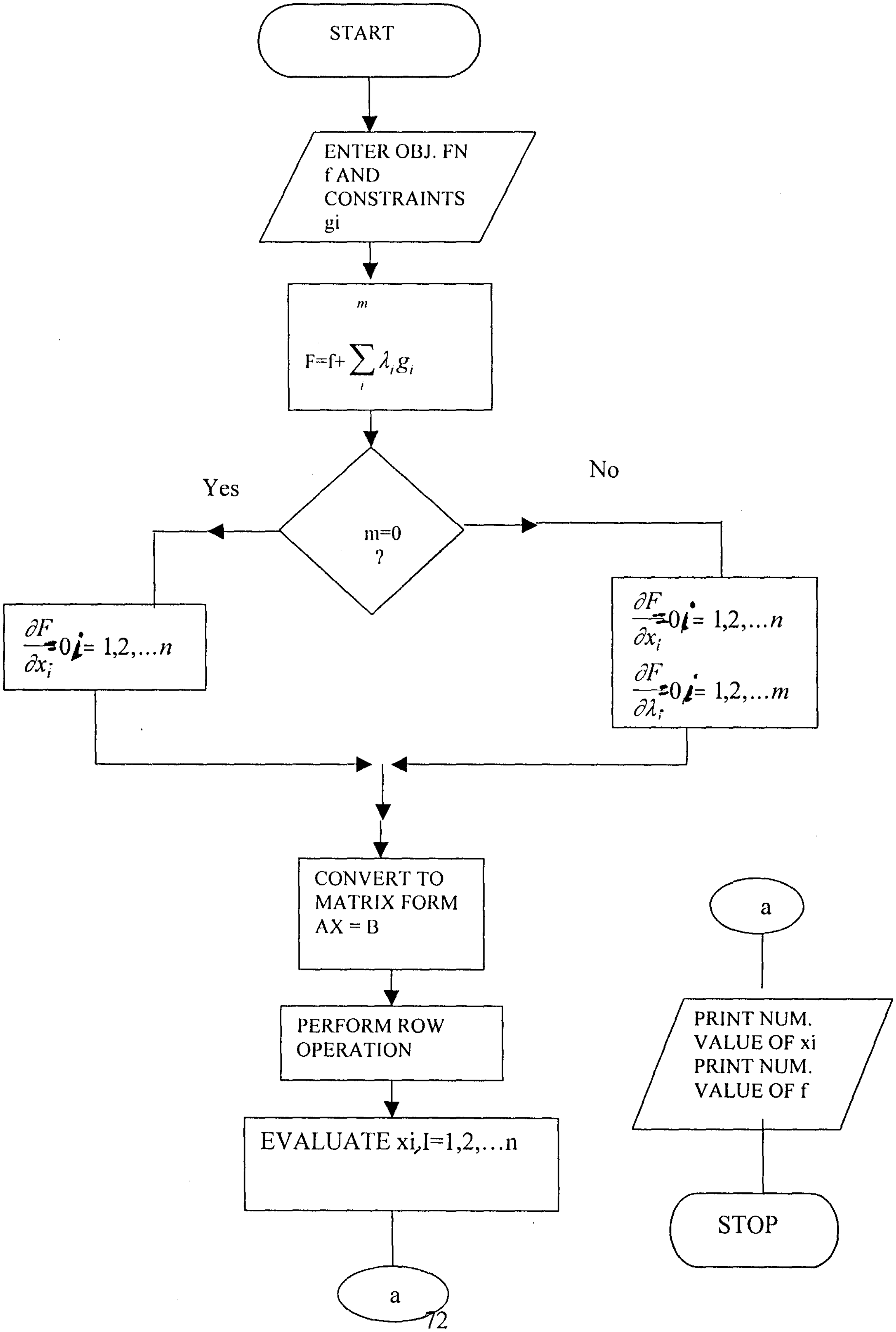
STEP VII

SUBSTITUTE THE DECISION VARIABLES x_i INTO OBJECTIVE FN, f(x)

STEP VIII

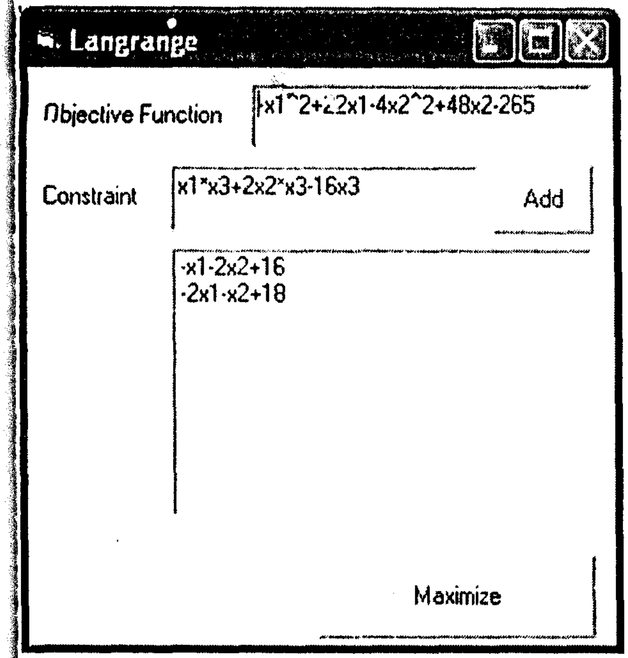
PRINT NUMERICAL VALUES OF THE DECISION VARIABLES x_i, $i=1,2,\ldots,n$ AND ALSO PRINT NUMERICAL VALUE OF OBJECTIVE FN, $f(x)$

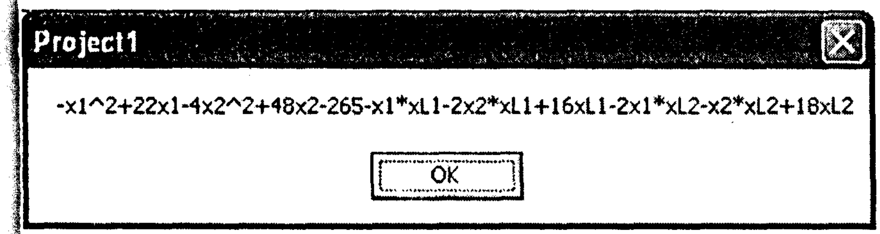
FLOWCHART FOR LAGRANGIAN MULTIPLIER'S METHOD

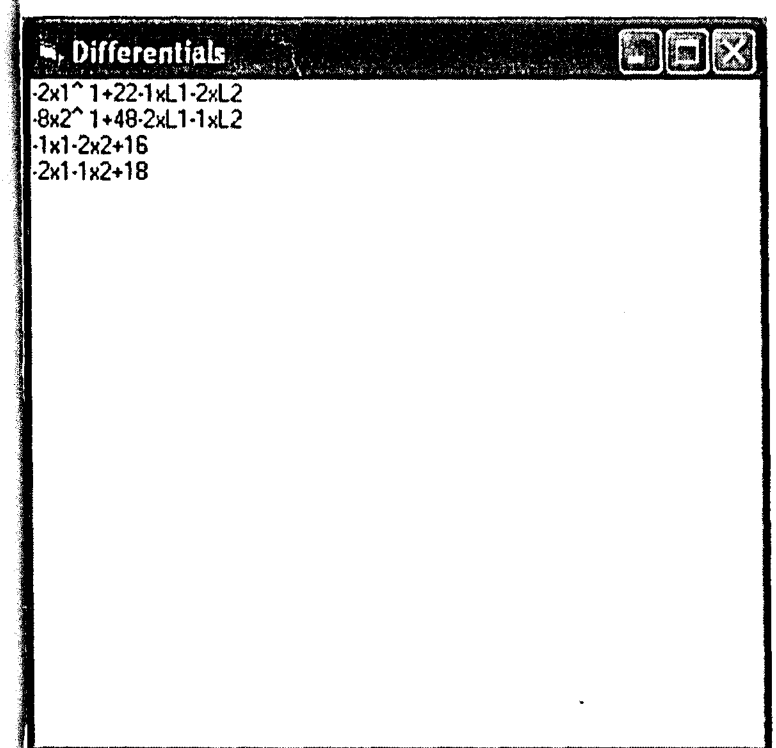


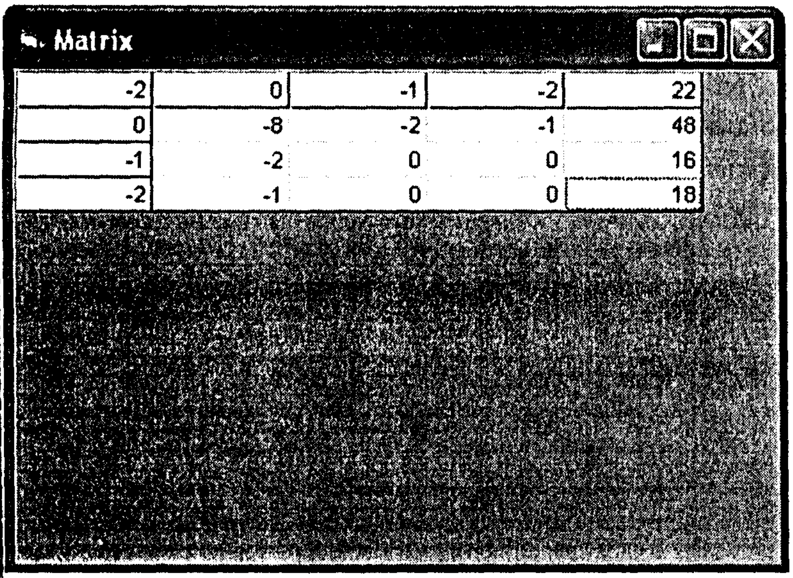
4.3 OUTPUT OF COMPUTER (LAGRANGIAN) CODE

Here, we have solution to the Example 3.2 in chapter 3, using the Lagrangian code in the Appendix of this project and the output is as shown on the next page:

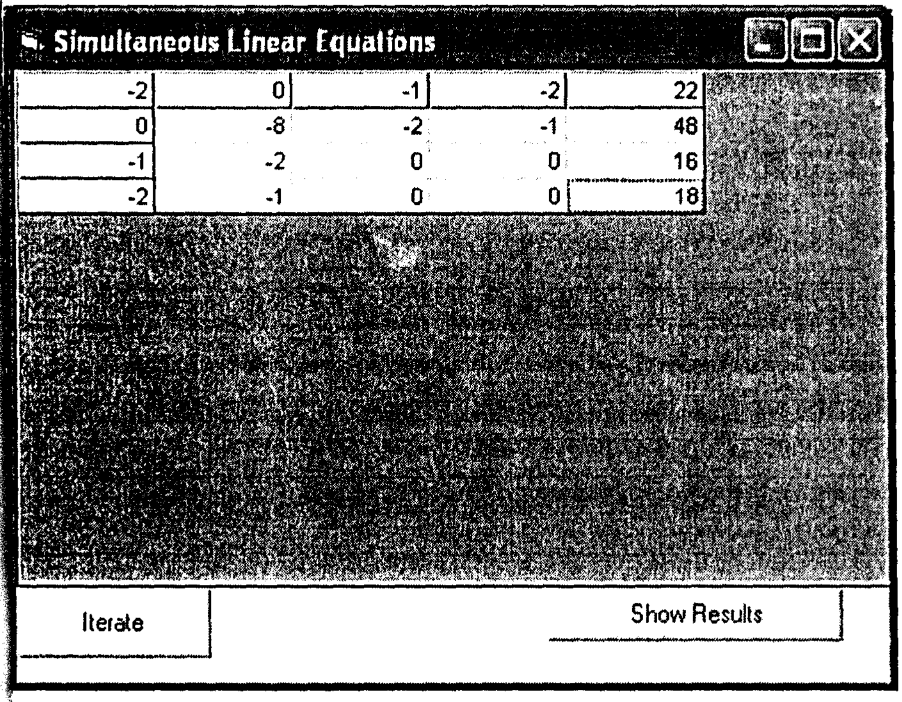

Langrange
Objective Function
-x1^2+22x1-4x2^2+48x2-265
Constraint
x1*x3+2x2*x3-16x3
Add
-x1-2x2+16
-2x1-x2+18
Maximize


Project1
-x1^2+22x1-4x2^2+48x2-265-x1*xL1-2x2*xL1+16xL1-2x1*xL2-x2*xL2+18xL2
OK

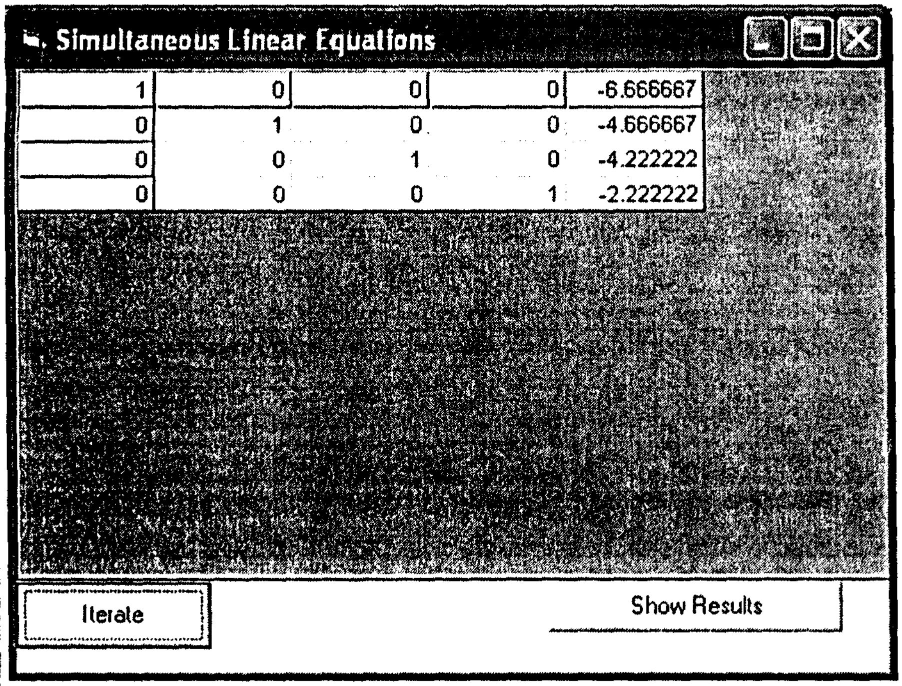

Differentials
-2x1^1+22-1xL1-2xL2
-8x2^1+48-2xL1-1xL2
-1x1-2x2+16
-2x1-1x2+18

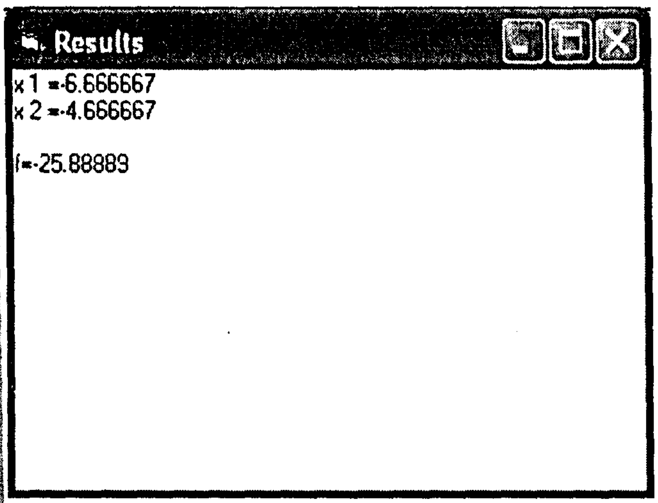


-2	0	-1	-2	22
0	-8	-2	-1	48
-1	-2	0	0	16
-2	-1	0	0	18



-2	0	-1	-2	22
0	-8	-2	-1	48
-1	-2	0	0	16
-2	-1	0	0	18


Simultaneous Linear Equations
1 0 0 0 -6.666667
0 1 0 0 -4.666667
0 0 1 0 -4.222222
0 0 0 1 -2.222222
Iterate
Show Results


Results
x 1 =-6.666667
x 2 =-4.666667
f=-25.88889

REMARKS

The chapter consists of Psuedocode, flowchart and output of the Lagrangian code on Example 3.2. The output of the code on Example 3.2 agreed with the results of Example 3.2 solved manually. The code in this project can solve some nonlinear programming problem (quadratic in nature) with large number of variables and linear constraints.

REFERENCES

Frederic C. Jelen (1983), Cost and Optimization Engineering, second edition, McGraw-Hill Coy, USA.

S. A. Hovanessian, (1976), Computational Mathematics in Engineering, first edition, D.C. Health and Coy.

Roger C. Pfaffenberger, David A. Walker (1976), Mathematics Programming for Economics and Business, first edition, IOWA State University Press, IOWA State.

Robert J. Thierauf, Robert C.Klekamp (1974), Decision Making Through Operations Research, second edition, John Wiley and Sons.

Frederick S. Hillier, Gerald J. Lieberman (1995), Introduction to Operations Research, sixth edition, McGraw-Hill, Singapore.

Harvey M. Wagner (1989), Principles of Operations Research with application to managerial decisions, second edition, Prince Hill, New Delhi.

Gray E. Whitehouse, Ben L. Wechsler (1976), Applied Operations Research, International Edition, John Wiley and sons.

```
.m B As String
.m C As Collection
et C = Objective.Split("+-")
or Each A In C
   If InStr(B, A.Var) = 0 Then
       B = B + " " + A.Var
       GetNoVars = GetNoVars + 1
   End If
ext A
nd Function


e
o






ex
nc

ri
ri
ri
nd

ub
or

ex
et
f
f
nd

ri
or
nd
```

CHAPTER FIVE

5.0 ANALYSIS OF RESULTS, RECOMMENDATION AND CONCLUSION

5.1 INTRODUCTION

This chapter deals with analysis, recommendation and conclusion of the whole project. So the chapter is divided into 3 sections as stated above.

5.2 ANALYSIS

The separable programming and Quadratic programming methods are used to find approximations to Nonlinear programming problems. And we are able to get good results for al the examples considered. But, looking through the procedures to be followed when separable programming method is to be adopted, we are able to see that Linearization of the objective function or constraints (or both) must take place, before the modified simplex method is now used to find both the decision variables and objective value, while in the case of the Quadratic programming technique the Kuhn Tucker technique is adopted in order to get linear programming problem before the modified simplex method is applied on it. In applying modified simplex method, the two-phase method is adopted in order to finds the basic feasible solution.

The Lagrangian's method is both necessary and sufficient conditions for the global optimum to be discovered. The method adopts the uses of partial derivatives of the sum of the objective function and constraints, F(x), with respect to the decision variables, x_i, and the Lagrangian multipliers', λ_i, and the resulting differentials equates to zeros. Therefore the resulting equation is now solve using the Row reduction (Gauss-Jordan) method.

5.3 RECOMMENDATION

The program in this project, is recommended for use only for the Quadratic problem (i.e. Quadratic objective) because provision is not made for the method(s) that solves nonlinear differentials, that may have resulted from the partial derivatives of the objective function plus the constraints F(x). The program (code) in the Appendix can be used to solve all the problems (Examples) in both chapter 2 and 3 as well.

APPENDIX (LAGRANGIAN MULTIPLIER'S CODE)

```
'User Interface Module Form 1
Public Constraints As Collection
Public Objective As Expression
Private Sub AddCommand_Click()
List1.AddItem Text2.Text
End Sub
Private Sub Command1_Click()
Dim Cons As Expression
Set Objective = New Expression
Set Constraints = New Collection
If Text1.Text = Empty Then
    MsgBox "you must provide the objective function to continue",
vbInformation
Else
    Objective.Value = Text1.Text
    'Constraint1.Value = Text2.Text
    Text2.Text = Diff(Objective, "x1").Value 'text2.Text
    For i = 0 To List1.ListCount - 1
        Set Cons = New Expression
        Cons.Value = List1.List(i)
        Constraints.Add Cons
    Next i
    m = List1.ListCount
    N = GetNoVars
    Dimension = m + N
    ReDim Matrix(1 To Dimension, 1 To Dimension + 1)
    ReDim Differentials(1 To Dimension)
    For i = 1 To Dimension
        Set Differentials(i) = New Expression
    Next i
    MsgBox AddUp.Value
    Form2.Show
End If
'Form2.Show
End Sub

Private Sub Text2_Click()
Set Objective = New Expression
Objective.Value = Text2.Text
Text2 = Diff(Objective, "x3").Value
End Sub
Function GetNoVars()
Dim A As SubExp
```

ser Interface Module Form 2

```
ivate Sub Form_Click()
rm3.Show
d Sub

ivate Sub Form_Load()
m A As Expression
t A = AddUp
r i = 1 To N
  Print Diff(A, "x" & LTrim(Str(i))).Value
  Differentials(i).Value = Diff(A, "x" & LTrim(Str(i))).Value
xt i
r i = 1 To m
  Print Diff(A, "xL" & LTrim(Str(i))).Value
  Differentials(i + N).Value = Diff(A, "xL" &
rim(Str(i))).Value
xt i
d Sub
```

ser Interface Module Form 3

```
ivate Sub Form_Load()
m A As SubExp
m C As Collection
id.Rows = Dimension
id.Cols = Dimension + 1
r i = 0 To Dimension - 1
   For j = 0 To Dimension
       Grid.Row = i
       Grid.Col = j
       Grid.Text = 0
       Matrix(i + 1, j + 1) = 0
   Next j
xt i
r i = 1 To Dimension
   Set C = Differentials(i).Split("+-")
   For Each A In C
       'If A.Var <> Empty Then
           Grid.Row = i - 1
           Grid.Col = GetNumber(A.Var) - 1
           Grid.Text = A.Multiplier
           If Grid.Text = Empty Then Grid.Text = 0
           Matrix(i, GetNumber(A.Var)) = A.Multiplier
       'End If
   Next A
ext i
nd Sub

rivate Sub Form_Resize()
rid.Width = Width
rid.Height = Height
nd Sub

ublic Function GetNumber(s As String) As Integer
or i = 1 To Len(s)
   If IsNumeric(Mid(s, i, 1)) Then Exit For
ext i
etNumber = Val(Right(s, Len(s) - i + 1))
f Mid(s, 2, 1) = "L" Then GetNumber = GetNumber + N
f GetNumber = 0 Then GetNumber = N + m + 1
nd Function

rivate Sub Grid_Click()
orm4.Show
nd Sub
```

ser Interface Module Form 4

```
ivate Sub Command1_Click()
erate
date
d Sub

ivate Sub Form_Load()
id.Rows = Dimension
id.Cols = Dimension + 1
date
d Sub

ivate Sub Form_Resize()
id.Width = Width
id.Height = Height - 1200
mmand1.Top = Height - 1180
d Sub

ub Update()
or i = 1 To Dimension
   For j = 1 To Dimension + 1
       Grid.Row = i - 1
       Grid.Col = j - 1
       Grid.Text = Matrix(i, j)
   Next j
ext i
nd Sub
ub Iterate()
row 1 by element 1,1
oom = 1
o
   t = Matrix(doom, doom)
   For i = 1 To Dimension + 1
       Matrix(doom, i) = Matrix(doom, i) / t
   Next i
   '
   For Row = 0 + 1 To Dimension
       If Row <> doom Then
       t = Matrix(Row, doom)
       For Col = 1 To Dimension + 1
           Matrix(Row, Col) = Matrix(Row, Col) - t * Matrix(doom,
ol)
       Next Col
       End If
   Next Row
   doom = doom + 1
oop Until doom = Dimension + 1
```

d Sub

```
ss Module SubExp
ic Value As String
ic Function Var() As String
nStr(Value, "x") = 0 Then
Var = ""
e
Var = Mid(Value, InStr(Value, "x"), InStr(Value + "^", "^") -
r(Value, "x"))
If
Function
lic Function Exponent() As Single
i As Integer
InStr(Value, "^")
i = 0 Then
Exponent = 1
e
Exponent = Right(Value, Len(Value) - i)
If
Function
lic Function Multiplier() As Single
Var <> Empty Then
 Multiplier = Val(Left(Value, InStr(Value, "x") - 1))
 If Not IsNumeric(Left(Value, InStr(Value, "x") - 1)) Then
     If Left(Value, InStr(Value, "x") - 1) = "-" Then
         Multiplier = -1
     Else
         Multiplier = 1
     End If
 End If
eIf InStr(Value, "^") <> 0 Then
 Multiplier = Left(Value, InStr(Value, "^"))
e
 Multiplier = Val(Value)
d If
d Function
```

Standard Module

```
ublic Matrix() As Single
ublic Differentials() As Expression
ublic Dimension As Integer
ublic N As Integer 'no of variables
ublic m As Integer 'no of constraints
ublic Function AddUp() As Expression
im StrTemp As String
trTemp = Form1.Objective.Value
im Con As Expression
 = 1
or Each Con In Form1.Constraints
   If Left(Multiply(Con, "xL" & LTrim(Str(i))), 1) = "+" Or
eft(Multiply(Con, "xL" & LTrim(Str(i))), 1) = "-" Then
       StrTemp = StrTemp & Multiply(Con, "xL" & LTrim(Str(i)))
   Else
       StrTemp = StrTemp & "+" & Multiply(Con, "xL" &
Trim(Str(i)))
   End If
   i = i + 1
ext Con
et AddUp = New Expression
ddUp.Value = StrTemp
nd Function

unction Multiply(m As Expression, L As String) As String
im t As Collection
et t = m.Split("+-")
im S As SubExp
or Each S In t
   If Left(S.Value, 1) = "+" Or Left(S.Value, 1) = "-" Then
       If IsNumeric(S.Value) Then
           Multiply = Multiply & S.Value & L
       Else
           Multiply = Multiply & S.Value & "*" & L
       End If
   Else
       If IsNumeric(S.Value) Then
           Multiply = Multiply & "+" & S.Value & L
       Else
           Multiply = Multiply & "+" & S.Value & "*" & L
       End If
   End If
Next S
End Function
Public Function Diff(E As Expression, Var As String) As Expression
Dim Temp As Collection, t As Collection
```

```
 E1 As Expression
 E1 = New Expression
 sE As SubExp
 D As String
 Temp = New Collection
 Temp = E.Split("+-")
 t = New Collection
 Each sE In Temp
D <> Empty And Sgn(sE.Multiplier) <> -1 Then D = D & "+"
 Left(D, 1) = " " Then
  D = "+" + LTrim(D)

nd If
InStr(sE.Value, Var) Then
 If InStr(sE.Var, "*") Then
     E1.Value = sE.Value
     Set t = E1.Split("*")
     If t(2).Var = Var Then D = D & Str(t(1).Multiplier *
2).Multiplier) & t(1).Var Else D = D & Str(t(1).Multiplier *
2).Multiplier) & t(2).Var
 Else
     If sE.Exponent - 1 <> 0 Then
         D = D & sE.Multiplier * sE.Exponent & sE.Var & "^" &
r(sE.Exponent - 1)
     Else
         D = D & sE.Multiplier * sE.Exponent '& sE.Var ' & "^"
Str(sE.Exponent - 1)
     End If
 End If
se
 If D <> Empty Then
     If Right(D, 1) = "+" Then D = Left(D, Len(D) - 1)
 End If
d If
xt sE
t Diff = New Expression
ff.Value = D
d Function
nction Eliminate(Exp As String, Var As String) As String
= InStr(Exp, Var)
 i = 0 Then
 Eliminate = Exp
se
 Eliminate = Left(Exp, i - 1) + Right(Exp, Len(Exp) - i -
en(Var) + 1)
nd If
nd Function
```

ass Module Expression

```
lic Value As String
lic Function Evaluate(Bindings As Collection) As Single
 subs As Collection
 E As SubExp, B As Binding
 subs = Split("+-")
 Each E In subs
 For Each B In Bindings
     If InStr(E.Value, B.Var) > 0 Then

         Exit For
     End If
 Next B
 Evaluate = Evaluate + Eval(E, B.Value)
xt E
d Function
blic Function Split(Tokens As String) As Collection
m noToks As Integer, i As Integer, Pos As Integer
m SubExp1 As SubExp
m Exp As String
p = Value
Toks = Len(Tokens)
m Toks() As String
Dim Toks(noToks)
r i = 1 To noToks
  Toks(i) = Mid(Tokens, i, 1)
xt i
t Split = New Collection
s = FindToken(Toks, noToks, Exp)
ile Pos <> 0
  Set SubExp1 = New SubExp
  SubExp1.Value = Left(Exp, Pos - 1)
  Exp = Right(Exp, Len(Exp) - Pos + 1)
  Split.Add SubExp1
  Pos = FindToken(Toks, noToks, Exp)
end
et SubExp1 = New SubExp
ubExp1.Value = Exp
plit.Add SubExp1
nd Function

rivate Function FindToken(T() As String, no As Integer, Exp As
tring)
Dim f As Integer
FindToken = InStr(2, Exp, T(1))
For i = 1 To no
   f = InStr(2, Exp, T(i))
```

```
If FindToken = 0 Then FindToken = f
 If f < FindToken And f <> 0 Then FindToken = f
t i
 Function
vate Function Eval(E As SubExp, V As Single) As Single
l = E.Multiplier * V ^ E.Exponent
 Function
lic Function RightSide() As Single
 Temp As Collection, SubE As SubExp
 Temp = Split("+-")
 Each SubE In Temp
 If SubE.Var = Empty Then
     RightSide = -SubE.Multiplier
     Exit For
 End If
xt SubE
d Function
```

CONCLUSION

In concluding this project work, we can say that among all the techniques adopted so far the most effective and reliable one is the Lagrangian multiplier's method, which is quiet straight forward, easy to adopt, and uses less compilation time.

The code in this project is only for Lagrangian multiplier's method, which can be used to solve any problems considered in this project. And the output of Example 3.2 in chapter three is a good illustration of this statement. And in addition, to this, it does give the global optimum value for the objective function.