DESIGN OF A LOOPED WATER DISTRIBUTION NETWORK USING NON LINEAR PROGRAMMING

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree
of
MASTER OF TECHNOLOGY
in
WATER RESOURCES DEVELOPMENT (CIVIL)

By

ROY PANAGOM PARDEDE

DEPARTMENT OF WATER RESOURCES DEVELOPMENT & MANAGEMENT
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE - 247 667 (INDIA)
JUNE, 2005

[illegible]

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled "Design of a Looped Water Distribution Network using Non Linear Programming", in partial fulfillment of the requirement for the award of the Degree of Master of Technology in Water Resources Development (Civil), submitted in the Department of Water Resources Development and Management (WRDM), Indian Institute of Technology Roorkee, Roorkee is an authentic record of my own work carried out during the period July 2004 to June 2005 under the supervision of Dr. M.L. Kansal, Associate Professor Department Water Resources Development and Management, and Dr. K. Deep, Associate Professor Department of Mathematics, Indian Institute of Technology Roorkee, India

I have not submitted the matter embodied in this thesis for the award of any other degree.

[illegible]

Date : June 17th, 2005
Place : Roorkee

Roy Panagom Pardede
Candidate

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Kusum Deep

Dr. Kusum Deep
Associate Professor,
Department of Mathematics, IIT Roorkee,
India

[illegible]

Dr. M.L. Kansal
Associate Professor,
Department of WRDM, IIT Roorkee,
India

ACKNOWLEDGMENTS

I would like to take this opportunity to express my sincere and profound gratitude to Dr. M.L. Kansal, Associate Professor, Department of Water Resources Development and Management, IIT Roorkee, for his kind help and valuable guidance, encouragement, suggestions at every stage of this dissertation. In spite of his busy schedule, he has been ever ready to help me to solve any difficulty until the completion of this dissertation.

My sincere gratitude also goes to Dr. Kusum Deep, Associate Professor, Department of Mathematics, IIT Roorkee, for her valuable guidance and encouragement throughout this dissertation work.

I express my sincere gratitude also to Dr. S.K. Triparthi, Professor and Head of Department of Water Resources Development and Management, IIT Roorkee for the facility extended.

My sincere thanks to Dr. E. Keedwell from Centre for Water Systems, University of Exeter, UK for his kind help during this work.

Thanks to Department of Public Works - Government of Indonesia, which gave me an opportunity of attending this course.

Lastly, I also express my heartfelt thanks to my mother and all my sisters for their support and encouragement, so that I could finish my course.

[illegible]

Date : June 17th, 2005
Place : Roorkee

Roy Panagom Pardede

SYNOPSIS

Design of water distribution network is become subject of experiment of many researchers. It is because of the distribution network cost takes major part of total network (it is around 70 percent), so the optimal design of water distribution network will bring the optimal design of water supply.

The problem of water distribution network is non linear in nature, due to the cost pipe function and hydraulic law that are formed the problem. The problem can be formulated in Non Linear Programming approach either in D-Q formulation, D-h formulation or in Q-h formulation. Along these three formulations, the Q-h formulation giving a better performance because of optimal result is achieved in small number of iteration.

In this thesis, Non Linear programming approach will be applied, with using Generalised Reduced Gradient (GRG) algorithms. Microsoft Excel Solver is used in the process of iteration. As it is applied for simple networks, Q-h formulation is also used for solving a real type network.

The solutions that are achieved by using Non Liner Programming approach are guarantee only for local optimal solution, and due to this a lot of number of trial solution is needed to ensure the optimal solution. The pipe diameter solutions that are achieved are continuous, so it needs to round up to market size diameter.

The using of Genetic Algorithms in design of water distribution network problem will increase the chance of getting the global optimum solution. It is because of the principles of genetics process in human nature are applied in getting optimal solutions in design of water distribution networks problem. The trial solutions will be simultaneously generated, to get the optimal solution. The superiority of Genetic Algorithms dealing with discrete variable for example market size diameter of pipe, is also becomes an advantage in solving water distribution network design problem. Software package "Water Network Optimiser" that using Genetic Algorithm, will be used in design a real type network, and it gives global optimal solution.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

1.1 WATER DISTRIBUTION SYSTEM

Municipal water distribution systems represent a major portion of the investment in urban infrastructure and a critical component of public works. The goal is to design water distribution systems to deliver potable water over spatially extensive areas in required quantities and under satisfactory pressures. In addition to these goals, cost-effectiveness and reliability in system design are also important.

Municipal water distribution systems are inherently complex because they are:

- large-scale and spatially extensive
- composed of multiple pipe loops to maintain satisfactory levels of redundancy for system reliability
- governed by nonlinear hydraulic equations
- designed with inclusion of complex hydraulic devices such as valves and pumps
- impacted by pumping and energy requirements
- complicated by numerous layout, pipe sizing, and pumping alternatives
- influenced by analysis of tradeoffs between capital investment and operations and maintenance costs during the design process.

Traditional methods of design of municipal water distribution systems are limited because system parameters are often generalized; spatial details such as installation cost are reduced to simplified values expressing average tendencies; and trial and error procedures are followed, invoking questions as to whether the optimum design has been achieved. Even with use of hydraulic network simulation models, design engineers are still faced with a difficult task.

The optimal design of municipal water distribution systems is a challenging optimization problem for the following reasons:

- the system optimization requires an imbedded hydraulic simulation model for pressurized, looped pipe networks
- the discrete decision variables are discrete, since pipe sizes must be selected from commercially available sets [e.g., 8", 10", 12", 15",...]; combinatorial problems involving discrete variables are considered NP-hard in optimization theory
- the optimization problem can be highly nonlinear due to nonlinear hydraulic models and pump characteristic curves
- the optimization problem should be regarded as stochastic due to uncertain demand loadings and system reliability issues
- one way of considering uncertain demands is to include multiple demand loading scenarios in the optimization, which increases problem size and complexity
- pressure constraints must be directly included in the optimization.

The optimal design of municipal water distribution systems involves numerous characteristics which carry significant spatial dependencies. These include:

- topography and its influence on pressure distribution in a pipe network
- street network characteristics, since most water distribution systems are installed in existing and planned road systems
- right of way issues
- congestion problems during installation due to buried utilities
- land use and development issues impacting installation costs, such as increased costs of pipe excavation in commercial districts due to business disruption and the need for traffic rerouting
- spatially distributed soil characteristics impacting excavation costs, such as loose, sandy soils requiring more costly reinforcement of the site.

1.2 STATE OF ART IN WATER DISTRIBUTION MODELS

The current focus in optimal design models is on improving the efficiency and realism of the optimization techniques. With development of "Operation Research", now we able to find the optimise solution for any particular problem through various methods. Various method such as: Linear Programming, Non Linear Programming, Dynamic Programming have also used in water distribution network design. A number of researchers have used Linear programming to optimise a design of a pipe network. Researchers have developed two principal approaches (Alperovits and Shamir 1977; Quindry et al. 1979). Dynamic Programming, that is developed by Richard Bellman in early 1950s, is powerful in solving allocation of water in water distribution problem. The principle of Dynamic Programming is decomposes a multistage decision problem into a sequence of single stage decision problem. As the number of unit that is allocated and number of resources increases, Dynamic Programming become quite complex. Dynamic Programming is rarely used to solve problems of allocation of more than two resources.

A wide variety of techniques have been proposed, with one of the most oft studied being the Linear Programming Gradient (LPG) method and its extensions (Alperovits and Shamir, 1977; Eiger, et al., 1994). However, Bhave and Sonak (1992) claim that the LPG method is inefficient compared with other methods.

Methods based on the use of linear programming (LP) have been developed which are capable of maintaining the constraint on discrete pipe sizes without the need for rounding off solutions. Morgan and Goulter (1985) modified the procedure of Kally (1972) to link a Hardy-Cross network solver with linear programming model. The model is designed to optimize both the layout and design of new systems and expansion of existing systems. It is a highly efficient method, with the main disadvantage being the generation of split pipe solutions (i.e., with some pipe sections requiring two pipe sizes). The latter indeed reduces system costs, but may not be attractive to design engineers.

More recent literature emphasizes reliability issues in water distribution system design, with consideration of the probabilities of satisfying system flow and pressure requirements. Lansey, et al. (1989) employed a chance constrained model to consider uncertainties in demands, pressure head, and pipe roughness. Bao and Mays (1990) applied Monte Carlo simulation methods to measure system reliability. Although reliability-based water distribution system models are useful for analysis of the problem, they may be impractible for designing large-scale systems. The use of multiple demand loading scenarios may be a means of indirectly including system reliability issues at more practical computational expense.

Some approaches attempt to employ efficient combinatorial methods to the optimal design problem. Gessler (1982) linked a network hydraulic simulation model to a filtering subroutine to efficiently enumerate all feasible solutions in pipe network design. This model selects both the optimal design, as well as several near-optimal solutions for tradeoff analysis, and is perhaps the most widely used optimization model.

Other authors have formulated the optimal design problem as a nonlinear programming problem with discrete pipe sizes treated as continuous variables. By considering the link diameters as a continuous variable, several researchers have suggested the optimisation of looped water distribution networks through nonlinear programming ever since Pitchai (1966), Jacoby (1968), and others applied NLP for optimisation of water distribution networks such as Chiplunkar, et al. (1986) employed the Davidon-Fletcher-Powell method to design a water distribution under a single demand loading scenario. Lansey and Mays (1989) coupled the generalized reduced gradient (GRG) algorithm with a water distribution simulation model to optimally size pipe network, pump stations, and tanks. . These approaches differ from one another in the formulation of the problem and/or the method of its solution. The primary disadvantage of these NLP methods is the required rounding off of optimal continuous decision variables to commercially available sizes, which can lead to network infeasibilities as well as raise questions as to optimality of the adjusted solution.

Recent studies have attempted to apply a variety of heuristic programming methods to the optimal design of water distribution systems. These include the application of genetic algorithms (Savic and Walters, 1997) and simulated annealing (Cunha and Sousa, 1999). The advantages of these methods are that they allow full consideration of system nonlinearity and maintain discrete design variables without requiring split pipe solutions.

The disadvantages include:

- cannot guarantee generation of even local optimal solutions, particularly for large-scale systems
- require extensive fine-tuning of algorithmic parameters, which are highly dependent on the individual problem
- can be extremely time consuming computationally
- current applications have not included use of multiple demand loadings because of computational difficulties.

1.3 OBJECTIVES

The aim of dissertation is to explore the application of Non Linear Programming in solving of design of a looped water distribution network problem. The problem of water distribution network is formulated as Non Linear Programming problem and will be solved using a suitable algorithm in Non Linear Programming approach. Due to increasing of complexity of the network will affect the searching of optimal solution, the random search method will also be applied. In this case, Genetic algorithm as a part of evolutionary algorithm will be incorporated in solving water distribution network design problem. These two approaches will be compared to know the better algorithm in solving looped water distribution network design problem.

CHAPTER II

PROBLEM FORMULATIONS

2.1 GENERAL

The design of water distribution systems is often viewed as a least-cost optimization problem with pipe diameters acting as the primary decision variables. However, although the cost of operating a water distribution system can be substantial (arising from maintenance, repair, water treatment, energy costs, and so on), the costs of some items often do not greatly depend on pipe size. In most situations, pipe layout, connectivity, and imposed minimum head constraints at pipe junctions (nodes) are taken as fixed design targets.

Clearly, other elements (such as service reservoirs and pumps) and other possible objectives (reliability, redundancy, flexibility in the face of uncertain future demands, and satisfactory water quality) can be included in the optimization process. But the difficulties of including reservoirs and pumps and quantifying additional objectives for use within the optimization process have focused researchers on determining pipe diameters while maintaining the single objective of least cost. Typically, pumping and storage alternatives are taken as entirely separate approaches that are considered outside of the optimization process. Even this somewhat limited formulation of optimal network design offers a difficult problem to solve (Savic and Walters, 1997). Generally the objective function of the pipe-sizing problem is assumed to be a cost function of pipe diameters and lengths:

$$\underset{x}{\text{minimise}}\, f(x) = \sum_{i=1}^{N} c_i(x_i, l_i) \tag{2.1}$$

where	f	=	objective function to be minimized
	x	=	vector of unknown diameters x_i
	N	=	number of pipes

c_i	=	cost function for pipe i
l_i	=	length of pipe i

The set of constraints associated with this problem consist of continuity and energy loss equations, which can be satisfied by running a standard hydraulic simulation program to evaluate the hydraulics of the solution. Other constraints may include

- The minimum and maximum head constraint at each or selected nodes
- The minimum and maximum velocity in pipes
- The minimum reliability and redundancy constraints
- Other operational constraints, such as balancing reservoirs within 24 hours or any other period, or ensuring at least a minimum turnover of water in storage

The initial process of optimisation problem of water distribution network is started by formulating the objective function. The objective function is to minimise the energy cost of pumped station and cost function of pipeline network (in case of pumped network) and to minimise cost function of pipeline network (in case of gravity network). Pipe cost term has variables such as diameter pipe and pipe length, and energy term has variables i.e. available head at source node and minimum head at each node. The constraints that must be satisfied are continuity of flow at each node, algebraic sum of the head losses in each loop is zero, algebraic sum of the head losses in a path from source to each demand node is not more than the permissible head loss in the path and all pipe lengths, diameter pipe and head loss and or available head are non negative.

2.2. FORMULATIONS

Consider a single source pumped (labelled 0), looped water distribution network having N demand nodes ($j = 1,.., N$), X links ($x = 1,... X$) and C basic circuits or loops ($c = 1,... C$). Since the variation in capital cost of pumps is negligible, we shall consider the present worth of energy charges, PW_e and the network cost C in the objective function, and follow the usual notation.

Since the diameter D, discharge Q and head loss h for a link are interrelated through the link head loss relationship, we can consider any two of them as basic decision variables. Thus, for general Non Linear Programming problem for looped networks, we have:

1. Diameter-discharge, *D-Q* formulation
2. Diameter-head loss, *D-h* formulation, and
3. Discharge-head loss, *Q-h* formulation

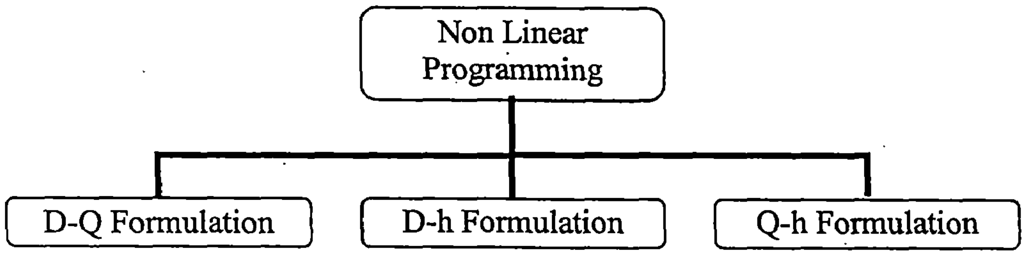


Figure 2-1 Types formulation of Non Linear Programming for Water Distribution Network Design

2.2.1 *D-Q* Formulation

In D-Q formulation, for a single source pumped network, decision variables are HGL at the source H0, the link diameters D_x, $x = 1,\dots X$; and link discharges Q_x, $x = 1, \dots, X$. Thus we have $2X+1$ decision variables. The objective function is:

$$\text{Minimise } C_T = K\,(H_0 - H_c) + \sum_{x=1}^{X} B_x L_x D_x^m \tag{2.2}$$

where :

C_T = cost of water distribution networks plus cost of energy for pump station

$$K = \frac{86000 c_e Q_m h_p F}{\eta}$$

C_e = cost of unit energy, monetary units/kWh

Q_m = mean discharge, m^3/s

H_p = pumping or total head, m

F = Present worth factor (P/A, i %, n)

η = pump efficiency

H_0 = Hydraulic gradient level (HGL) at source node

H_c = Hydraulic gradient level (HGL) at demand node

L_x = link of each link

D_x = diameter of each link

B, m = pipe cost constant

Since the node flow continuity must be satisfied at each demand node, we have N node flow continuity constraints,

$$\sum_{x incident on j} Q_x + q_j = 0 \tag{2.3}$$

where:

Q_x = discharge at link x, m^3/s

q_j = demand node at j, m^3/s

The head loss in links along each loop must be balanced, thus we have C loop head loss constraints,

$$\sum_{x \in c} A L_x Q_x^p D_x^{-r} = 0 \tag{2.4}$$

where:

A = constant depending on the link material, and units of different terms

L_x = length of link x

Q_x = discharge at link x

D_x = diameter pipe of link x

P, r = exponents equal to 2 and 5, respectively, in Darcy Weisbach formula; 1.85 and 4.87, respectively in Hazen William formula and 2 and 5.33 respectively in Manning formula.

To satisfy the HGL constraint at each demand node we have N path head loss constraints,

$$\sum_{x \in P_j} A L_x Q_x^p D_x^{-r} \leq H_0 - H_j^{\min} \tag{2.5}$$

where:

H_0 = Hydraulic gradient level at source node

$H_j^{\min}$ = Minimum Hydraulic gradient level required at node j

In addition, we have the usual non negative constraints for decision variables:

$$H_0 \geq 0\ (\geq H_c),\ D_x \geq 0,\ Q_x \geq 0 \tag{2.6}$$

When the optimisation problem with the non negative constraints is solved, the looped network converts to a branched one. To avoid this, the nonnegative constraints can be replaced by finite value boundary constraints. The link diameters should at least be of the minimum size, $D_{\min}$, corresponding to the D-specified condition. Alternatively, the link discharges must not be less than some specified discharge $Q_{\min}$, corresponding to the Q- specified condition. Thus, the boundary constraints are:

$$H_0 \geq 0\ (\geq H_c),\ D_x \geq D_{\min},\ Q_x \geq Q_{\min} \tag{2.7}$$

2.2.2 *D-h Formulation*

Objective function:

$$\text{Minimise } C_T = K\,(H_0 - H_c) + \sum_{x=1}^{X} B_x L_x D_x^m \tag{2.8}$$

subject to:

$$\sum_{x incident on j} \left[(h_x D_x^r)/(AL_x)\right]^{1/p} + q_j = 0 \tag{2.9}$$

$$\sum_{x \in c} h_x = 0 \tag{2.10}$$

$$\sum_{x \in P_j} h_x \le H_0 - H_j^{\min} \tag{2.11}$$

$$H_0 \ge 0\ (\ge H_c),\ D_x \ge 0 \tag{2.12}$$

or

$$H_0 \ge 0\ (\ge H_c),\ D_x \ge D_{\min},\ [(h_x D_x^r)/(AL_x)]^{1/p} \ge Q_{\min} \tag{2.13}$$

2.2.3 *Q-h Formulation*

$$\text{Minimise } C_T = K\,(H_0 - H_c) + \sum_{x=1}^{X} B_x A_x^{m/r} L_x^{1+m/r} Q_x^{pm/r} h_x^{-m/r} \tag{2.14}$$

subject to:

$$\sum_{x incident on j} Q_x + q_j = 0 \tag{2.15}$$

$$\sum_{x \in c} h_x = 0 \tag{2.16}$$

$$\sum_{x \in P_j} h_x \le H_0 - H_j^{\min} \tag{2.17}$$

$$H_0 \ge 0\ (\ge H_c),\ h_x \ge 0,\ Q_x \ge 0 \tag{2.18}$$

or

$$H_0 \ge H_c,\ h_x \ge AL_x Q_x^p (D^{\min})^{-r},\ \text{or } Q_x \ge Q^{\min} \tag{2.19}$$

Example 2.1

A simple water distribution network is given. It is a two-loop gravity network as shown in Figure 2-2. Node 1 is a source node with HGL of 210 m and nodes 2, …, 7 are demand nodes having demands, shown near the arrow heads, and minimum required HGL values, shown near the nodes. The network has eight links, each 1,000 m long.

The required pressure head and demand at each node is given in Table 2-1.

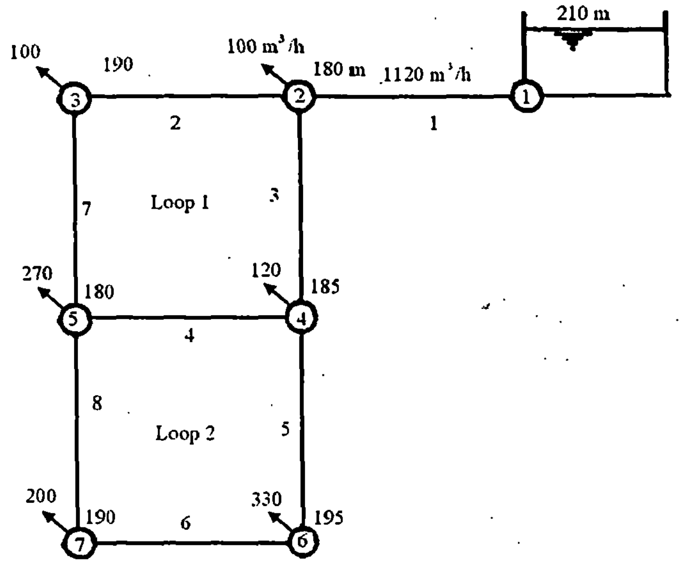


Figure 2-2 Two-loop gravity network

Table 2-1 Required Pressure Head and Demand at each node

Node	Required Pressure head (m)	Demand (m^3/h)
1	210	-
2	180	100
3	190	100
4	185	120
5	180	270
6	195	330
7	190	200

The link cost function is $C = 1.2654\ LD^{1.327}$ in which D is in millimetres, L in metres and C in rupees. Formulate optimisation problem using: (1) D-Q formulation; (2) D-h formulation and (3) Q-h formulation. Use Hazen-Williams head loss formula for headloss calculation, with Hazen Williams coefficient is 130 for all links.

Solution

The cost function can be written as follows:

$C = 1.2654\ (1000)\ (10^3\ D)^{1.327}$

$= 12.1123\ 10^6\ D^{1.327}$

where, C = Cost of pipe per m' (in rupees)

D = diameter of pipe (in meter)

Head loss function, $h = \dfrac{10.7LQ^{1.852}}{C_{HW}^{1.852} D^{4.87}}$

$$= \frac{(10.7)(1000)\left(\dfrac{Q}{3600}\right)^{1.852}}{130^{1.852} D^{4.87}}$$

$$= 3.37356\ \ 10^{-7}\ Q_n^{\ 1.852}\ D_n^{\ -4.87}$$

where, h_n = headloss of link-n (in metre)

Q_n = discharge in link-n (in m^3/s)

D_n = diameter of pipe of link-n (in meter)

1. *D-Q* Formulation

Objective function,

Minimise $C = 12.1123\ 10^6\ (D_1^{1.327} + D_2^{1.327} + D_3^{1.327} + D_4^{1.327} + D_5^{1.327} + D_6^{1.327} + D_7^{1.327} + D_8^{1.327})$

subject to:

Constraints of flow continuity at each node:

$1020 - Q_2 - Q_3 = 0$

$Q_3 - Q_5 - Q_4 - 120 = 0$

$Q_2 - Q_7 = 100$

$Q_4 + Q_8 + Q_7 = 270$

$Q_5 - Q_6 - 330 = 0$

$Q_6 - Q_8 - 200 = 0$

Path head loss constraints:

$3.37356\ 10^{-7}\ Q_1^{1.852} D_1^{-4.87} \le 30$

$3.37356\ 10^{-7}\ (Q_1^{1.852} D_1^{-4.87} + Q_2^{1.852} D_2^{-4.87}) \le 20$

$3.37356\ 10^{-7}\ (Q_1^{1.852} D_1^{-4.87} + Q_3^{1.852} D_3^{-4.87}) \le 25$

$$3.37356\ 10^{-7}\ (Q_1^{1.852}\, D_1^{-4.87} + Q_2^{1.852}\, D_2^{-4.87} + Q_7^{1.852}\, D_7^{-4.87}) \leq 30$$

$$3.37356\ 10^{-7}\ (Q_1^{1.852}\, D_1^{-4.87} + Q_3^{1.852}\, D_3^{-4.87} + Q_4^{1.852}\, D_4^{-4.87}) \leq 30$$

$$3.37356\ 10^{-7}\ (Q_1^{1.852}\, D_1^{-4.87} + Q_3^{1.852}\, D_3^{-4.87} + Q_5^{1.852}\, D_5^{-4.87}) \leq 15$$

$$3.37356\ 10^{-7}\ (Q_1^{1.852}\, D_1^{-4.87} + Q_3^{1.852}\, D_3^{-4.87} + Q_5^{1.852}\, D_5^{-4.87} + Q_6^{1.852} D_6^{-4.87}) \leq 20$$

Constraints of summation of headloss on loop equal zero:

$$3.37356\ 10^{-7}\ (Q_2^{1.852}\ D_2^{-4.87} + Q_7^{1.852}\ D_7^{-4.87}) = 3.37356\ 10^{-7}\ (Q_3^{1.852}\ D_3^{-4.87} + Q_4^{1.852}\ D_4^{-4.87})$$

$$3.37356\ 10^{-7}\ (Q_5^{1.852}\ D_5^{-4.87} + Q_6^{1.852}\ D_6^{-4.87} + Q_8^{1.852}\ D_8^{-4.87}) = 3.37356\ 10^{-7}\ (Q_4^{1.852}\ D_4^{-4.87})$$

Non negativity constraints:

$D_1, D_2, \ldots., D_8 \geq 0$

$Q_1, Q_2, \ldots., Q_8 \geq 0$

2. *D-h* Formulation

Objective function,

$$\text{Minimise } C = 12.1123\ 10^6\ (D_1^{1.327} + D_2^{1.327} + D_3^{1.327} + D_4^{1.327} + D_5^{1.327} + D_6^{1.327} + D_7^{1.327} + D_8^{1.327})$$

subject to:

Constraints of flow continuity at each node:

$$1020 - \left(\frac{h_2 D_2^{4.87}}{3.37356(10^{-7})}\right)^{0.539} - \left(\frac{h_3 D_3^{4.87}}{3.37356(10^{-7})}\right)^{0.539} = 0$$

$$\left(\frac{h_3 D_3^{4.87}}{3.37356(10^{-7})}\right)^{0.539} - \left(\frac{h_5 D_5^{4.87}}{3.37356(10^{-7})}\right)^{0.539} - \left(\frac{h_4 D_4^{4.87}}{3.37356(10^{-7})}\right)^{0.539} = 120$$

$$\left(\frac{h_2 D_2^{4.87}}{3.37356(10^{-7})}\right)^{0.539} - \left(\frac{h_7 D_7^{4.87}}{3.37356(10^{-7})}\right)^{0.539} = 100$$

$$\left(\frac{h_4 D_4^{4.87}}{3.37356(10^{-7})}\right)^{0.539} + \left(\frac{h_8 D_8^{4.87}}{3.37356(10^{-7})}\right)^{0.539} + \left(\frac{h_7 D_7^{4.87}}{3.37356(10^{-7})}\right)^{0.539} = 270$$

$$\left(\frac{h_5 D_5^{4.87}}{3.37356(10^{-7})}\right)^{0.539} - \left(\frac{h_6 D_6^{4.87}}{3.37356(10^{-7})}\right)^{0.539} = 330$$

$$\left(\frac{h_6 D_6^{4.87}}{3.37356(10^{-7})}\right)^{0.539} - \left(\frac{h_8 D_8^{4.87}}{3.37356(10^{-7})}\right)^{0.539} = 200$$

Constraints of summation of headloss on loop equal zero:

$h_2 + h_7 - h_3 - h_4 = 0$

$h_4 = h_5 + h_6 + h_8$

Path headloss constraints:

$h_1 \leq 30$

$h_1 + h_2 \leq 20$

$h_1 + h_3 \leq 25$

$h_1 + h_2 + h_7 \leq 30$

$h_1 + h_3 + h_7 \leq 30$

$h_1 + h_3 + h_5 \leq 15$

$h_1 + h_3 + h_5 + h_6 \leq 20$

$h_1 + h_3 + h_5 + h_6 + h_8 \leq 30$

Non negativity constraints:

$D_1, D_2, \ldots, D_8 \geq 0$

$h_1, h_2, \ldots, h_8 \geq 0$

3. *Q-h* formulation

$C = 12.1123\ 10^6 D^{1.327}$ .... (a)

$$D = \left(\frac{3.37356(10^{-7})Q^{1.852}}{h}\right)^{\frac{1}{4.87}}$$ ..... (b)

Combine a & b, we get:

$$C = 12.1123\ 10^6 \left(\frac{3.37356(10^{-7})Q^{1.852}}{h} \right)^{\frac{1.327}{4.87}}$$

$$= 208,814.2616\ Q^{0.5046}\ h^{-0.2725}$$

So, the objective function:

Minimise $C = 208814.5616\ (1120^{0.5046}\ h_1^{-0.2725} + Q_2^{0.5046} h_2^{-0.2725} + Q_3^{0.5046}$
$h_3^{-0.2725} + Q_4^{0.5046} h_4^{-0.2725} + Q_5^{0.5046} h_5^{-0.2725}$
$+ Q_6^{0.5046} h_6^{-0.2725} + Q_7^{0.5046} h_7^{-0.2725} + Q_8^{0.5046}$
$h_8^{-0.2725})$

subject to:

Node flow continuity constraint:

$1020 - Q_2 - Q_3 = 0$

$Q_3 - Q_5 - Q_4 - 120 = 0$

$Q_2 - Q_7 = 100$

$Q_4 + Q_8 + Q_7 = 270$

$Q_5 - Q_6 - 330 = 0$

$Q_6 - Q_8 - 200 = 0$

Constraints of summation of headloss on loop equal zero:

$h_2 + h_7 - h_3 - h_4 = 0$

$h_4 = h_5 + h_6 + h_8$

Path headloss constraints:

$h_1 \leq 30$

$h_1 + h_2 \leq 20$

$h_1 + h_3 \leq 25$

$h_1 + h_2 + h_7 \leq 30$

$h_1 + h_3 + h_7 \leq 30$

$h_1 + h_3 + h_5 \leq 15$

$h_1 + h_3 + h_5 + h_6 \leq 20$

$h_1 + h_3 + h_5 + h_6 + h_8 \leq 30$

Non negativity constraints:

$Q_1, Q_2, \ldots, Q_8 \geq 0$

$h_1, h_2, \ldots, h_8 \geq 0$

2.3 SUMMARY

The nature of objective function and constraints for formulations in water distribution network design problem is shown in Table 2-2.

Table 2-2 Nature of objective function and constraints

Formulation	*Objective function*	*Constraints*				
		Node flow continuity	*Loop head loss*	*Path head loss*	*Non negativity*	*Q- or D-specified*
D-Q	Non-linear	Linear	Non-linear	Non-linear	Linear	Linear
D-h	Non-linear	Non-linear	Linear	Linear	Linear	Non-linear
Q-h	Non-linear	Linear	Linear	Linear	Linear	Non-linear

If objective function and or constraints are nonlinear, the problem becomes a non linear programming problem. The formulation above indicates that the problem in water distribution network design is a non linear programming in nature. Between of these three formulations, it is easier to solve Q-h formulation than D-Q or D-h formulation, because all the constraints are in linear state, it is better to choose Q-h formulation.

CHAPTER III

TECHNIQUES FOR SOLVING NON LINEAR PROGRAMMING PROBLEMS

3.1 GENERAL

In Non Linear Programming Problem, solution can be obtained from classical optimisation method and from numerical optimisation method. Classical optimisation method proved to be performed well if objective function or constraints are fairly simple in terms of decision variables. In practice, however, the objective function and or constraints would be too complicated to be manipulated for obtaining the optimal solution. In such cases, the numerical approach would be necessary.

3.2 CHARACTERISTICS OF AN NLP PROBLEM

According to the nature of objective function and constraints, the optimisation problem can be divided into 2, i.e. linear and nonlinear programming problem. When the objective function and all the constraints are linear function of the decision variables, the optimisation problem is called a linear programming (LP) problem. When either of objective function or constraints is nonlinear, the optimisation problem is defined as nonlinear programming (NLP) problem.

Mathematically, a non linear programming problem can be expressed as:

$$\text{Optimise } Z = f(x_1,x_2,\ldots,x_n) \tag{3.1}$$

subject to

$$\left.\begin{matrix} g_1(x_1,x_2,\ldots,x_n) \\ g_2(x_1,x_2,\ldots,x_n) \\ \cdot \\ \cdot \\ \cdot \\ g_m(x_1,x_2,\ldots,x_n) \end{matrix}\right\} \begin{matrix} \leq \\ = \\ \geq \end{matrix} \left\{\begin{matrix} b_1 \\ b_2 \\ \cdot \\ \cdot \\ \cdot \\ b_m \end{matrix}\right.$$

where the variables $(x_1,x_2,\ldots,x_n)$ are non linear in nature.

The presence of constraints in an NLP problem creates difficulties in finding the optimum solution. For example, consider some minimisation problem with linear and non linear constraints as shown in Figure 3-1.

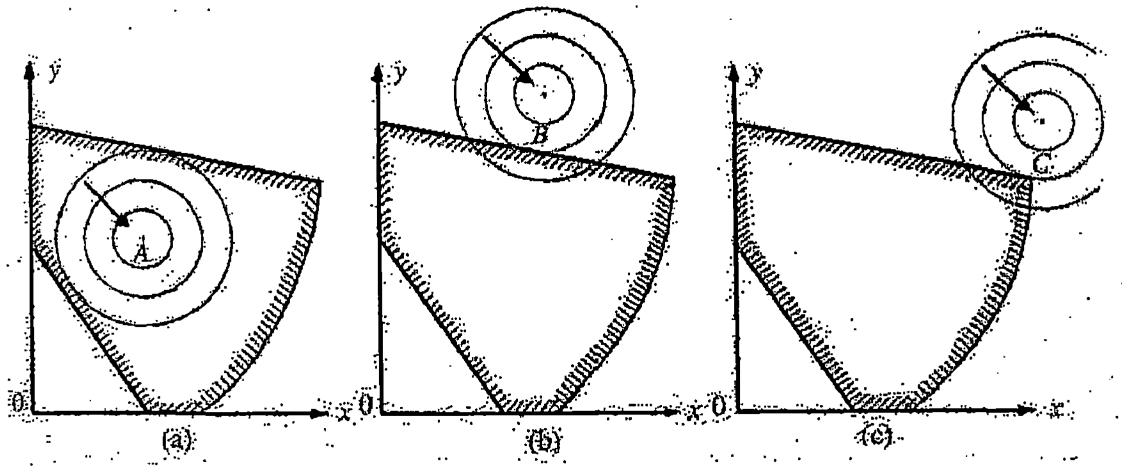


Figure 3-1 Nonlinear programming minimisation problem: (a) inactive constraints; (b) minimum occurring on a linear constraints, and (c) minimum occurring at the point of intersection of two constraints

The feasible region is shown shaded. Different objective function contours are also shown, and the arrows show the direction in which the objective function value decreases. Each problem has one minimum solution. The simplest situation is shown in Figure 3-1(a) where the constraints are inactive so that the optimum solution is the same as the unconstrained one, as shown in point A. However, for practical problems one or more constraints may be active so that the optimum solution would be at point B or at point C as shown in Figure 3-1(b) and Figure 3-1(c) respectively. Therefore for practical problems it is better to start with an assumption that at least some of the constraints would be active and play a role in deciding the optimal solution. However, if we can identify, a priori, inactive constraints from the constraints set, then we can omit them and simplify the NLP problem to that extent.

3.3 CLASSICAL OPTIMISATION

The classical methods of optimisation are useful in finding the optimum solution of continuous and differentiable functions. These methods are analytical and make use the techniques of differential calculus in locating the optimum points. Constraints may be absent or present; accordingly there will be unconstrained optimisation and constrained optimisation. Since some of the practical problems involve objective functions that are not continuous and or differentiable, the classical optimisation techniques have limited scope in practical applications. However, a study of the calculus methods of optimisation forms a basis for developing most of the numerical techniques of optimisation.

3.3.1 Unconstrained Optimisation

1. Single Variable Objective Function

The simplest unconstrained optimisation problem is a single variable objective function for which Theorems 3.1 and 3.2 give the necessary and sufficient conditions, respectively.

Theorem 3.1. If a function is defined in the interval $a \leq x \leq b$, and at $x=x^*$ $(a<x^*<b)$ if the derivative $df(x)/dx=f'(x)$ exists as a finite number at $x=x^*$, then $f'(x^*) = 0$ gives a minimum, maximum or stationary point.

Theorem 3.2. Let $f'(x^*) = f''(x^*) = \ldots = f^{n-1}(x^*)=0$, but $f^n(x^*) \neq 0$. Then $f(x^*)$ is:

(a) A minimum value of $f(x)$ if $f^n(x^*)>0$ and n is even,

(b) A maximum value of $f(x)$ if $f^n(x^*)<0$ and n is even,

(c) Neither a minimum nor a maximum if n is odd.

(Note: the superscript *, denotes an optimum value, either the minimum or the maximum value).

Example 3.1

Determine the maximum and minimum values of the function:

$f(x) = 12x^5 - 45x^4 + 40x^3 + 5$

Solution:

Since $f'(x) = 60\,(x^4 - 3x^3 + 2x^2) = 60\,x^2\,(x-1)\,(x-2)$, the value $f'(x) = 0$ at $x = 0$, $x = 1$, and $x = 2$.

The second derivative is

$f''(x) = 60\,(4x^3 - 9x^2 + 4x)$

At $x = 1$, $f''(x) = -60$ and hence $x = 1$ is a relative maximum. Therefore,

$f_{\max} = f(x{=}1) = 12$

At $x = 2$, $f''(x) = 240$ and hence $x = 2$ is a relative minimum. Therefore,

$f_{\min} = f(x{=}2) = -11$

At $x = 0$, $f''(x) = 0$ and hence we must investigate the next derivative.

$f'''(x) = 60(12x^2 - 18x + 4) = 240$ at $x = 0$

Since $f'''(x) \neq 0$ at $x=0$, $x = 0$ is neither a maximum nor a minimum and it is an inflection point.

2. Multiple Variable Objective Function

If the objective function has several variables, say n, giving objective function

$$Z = f(x_1, x_2, \ldots, x_n) \tag{3.2}$$

then the necessary and sufficient conditions are given by Theorem 4.3 and 4.4 respectively.

Theorem 3.3. If an extreme point (minimum or maximum) exist for a function $f(x_1, x_2, \ldots, x_n)$ and also the first partial derivatives exist at this point, then at this point

$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial x_2} = \ldots. = \frac{\partial f}{\partial x_n} = 0 \tag{3.3}$$

Theorem 3.4. The matrix of second partial derivatives (Hessian matrix) of $f(x_1, x_2, \ldots, x_n)$ evaluated at the extreme points is:

a) Positive definite for minimum solution, and

b) Negative definite for maximum solution.

The Hessian matrix of $f(x_1, x_2, \ldots, x_n)$ having second partial derivatives is given by

$$H_f = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \cdot & \cdot & \cdot & \cdot \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{3.4}$$

One simple test to know whether the matrix is positive definite or negative definite of Hessian matrix, A of order n is by evaluating the determinants of matrix.

$$A = | a_{11} | \tag{3.5}$$

$$A_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \tag{3.6}$$

$$A_3 = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \tag{3.7}$$

$$A_n = \begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{vmatrix} \tag{3.8}$$

The matrix A will be positive definite if and only if all the values A_1, A_2, A_3, ...A_n are positive. The matrix A will be negative definite if and only if the sign of A_j is $(-1)^j$ for $j = 1,2,\ldots n$. If some of the A_j are positive and the remaining A_j are zero, the matrix A will be positive semidefinite.

In case of a function of two variables $f(x,y)$, The Hessian matrix may be neither positive definite nor negative definite at a point (x^*, y^*) at which $(\partial f/\partial x)$ and $(\partial f/\partial y)$ are equal zero. In this case the point (x^*,y^*) is called a saddle point. The characteristic of saddle point is that it gives a relative minimum with respect to one variable while it gives a relative maximum with respect to another variable, as shown in Figure 3-2 for function, $Z = x^2 - y^2$, with saddle point at $x = 0, y = 0$.

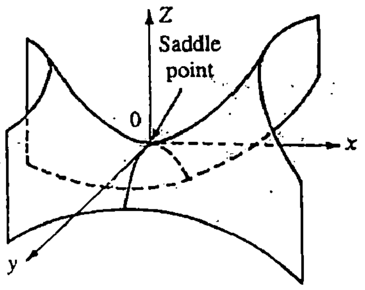


Figure 3-2 Saddle point of function $Z = x^2 - y^2$

Example 3.2

Find the extreme points of the function

$f(x_1,x_2) = x_1^3 + x_2^3 + 2x_1^2 + 4x_2^2 + 6$

Solution:

The necessary conditions for the existence of an extreme point are

$$\frac{\partial f}{\partial x_1} = 3x_1^2 + 4x_1 = x_1(3x_1 + 4) = 0$$

$$\frac{\partial f}{\partial x_2} = 3x_2^2 + 8x_2 = x_2(3x_2 + 8) = 0$$

These equations are satisfied at the points

$(0,0)$, $(0,-\frac{8}{3})$, $(-\frac{4}{3},0)$ and $(-\frac{4}{3}, -\frac{8}{3})$

To find the nature of these extreme points, we have to use the sufficiency conditions.

The second-order partial derivatives of f are given by

$$\frac{\partial^2 f}{\partial x_1^2} = 6x_1 + 4$$

$$\frac{\partial^2 f}{\partial x_2^2} = 6x_2 + 8$$

$$\frac{\partial^2 f}{\partial x_1 x_2} = 0$$

The Hessian matrix of f is given by

$$J = \begin{bmatrix} 6x_1 + 4 & 0 \\ 0 & 6x_2 + 8 \end{bmatrix}$$

If $J_1 = |6x_1 + 4|$ and $J_2 = \begin{vmatrix} 6x_1+4 & 0 \\ 0 & 6x_2+8 \end{vmatrix}$, the values of J_1 and J_2 and the nature of the extreme point are as given below:

Table 3-1 Value and nature of extreme points

Point X	Value of J_1	Value of J_2	Nature of J	Nature of X	F(X)
(0,0)	4	32	Positive definite	Relative minimum	6
$(0,-\frac{8}{3})$	4	-32	Indefinite	Saddle point	$\frac{418}{27}$
$(-\frac{4}{3},0)$	-4	-32	Indefinite	Saddle point	$\frac{194}{27}$
$(-\frac{4}{3},-\frac{8}{3})$	-4	32	Negative definite	Relative maximum	$\frac{50}{3}$

3.3.2 Constrained Optimisation

In constrained optimisation, the objective function of several variables has constraints, which may be equality constraints, or inequality constraints.

1. **Equality constraints.**

The problem is defined with only equality symbol in constraints. Naturally number of constraints is less than number of variables. When number of constraints (m) is more than number of variables (n), the problem is over defined and there is no solution. When $m = n$, the problem has a unique solution. It is only when $m<n$ that the problem has many solutions and the question of optimisation arises. Of the several methods available for solution of this problem, the methods of direct substitution and Lagrange multipliers are discussed herein.

a. **Method of Direct Substitution**

The m equality constraints are simultaneously solved and any m variables are expressed in terms of remaining n-m variables. These expressions are then substituted in the objective function so that the objective function becomes an unconstrained one in n-m variables. This optimisation problem then can be solved by using the method using for unconstrained optimisation problem.

Method of direct substitution appears to be quite simple in theory, but when the constraints are nonlinear as usually is the case in design of water distribution networks, it is not easy to solve *m* equations simultaneously and express *m* variables in terms of remaining *n-m* variables.

Example 3.3

Solve the following optimisation problem by direct substitution.

Minimise $Z = 2x_1 + 9x_1x_2 + 20x_3^2$ (E₁)

subject to

$x_1 - 3x_2 + 2x_3 = 6$ (E₂)

$x_2 + 3x_3 = 4$ (E₃)

Solution:

From equation (E₃),

$x_2 = 4 - 3x_3$ (E₄)

and from equation (E₂)

$x_1 = 6 + 3(4 - 3x_3) - 2x_3$

or $x_1 = 18 - 11x_3$ (E₅)

Substituting the values of x_1 and x_2 in the objective function and simplifying, we get

Minimise $Z = 684 - 904x_3 + 317x_3^2$ (E₆)

Optimisation problem of equation (E₆) is now a single variable, unconstrained one. Therefore, $dZ/dx_3 = 0$ gives

$-904 + 2(317\,x_3) = 0$

or $x_3^* = 1.426$

Substituting the value of x_3^*, we get $x_1^* = 2.315$; $x_2^* = -0.278$ and $Z^* = 39.508$

Since d^2Z/dx_3^2 is positive, the stationary point gives the minimum value. Thus, minimum value of Z, $Z^* = 39.508$

b. Method of Lagrange Multipliers

In Lagrange multiplier method, an additional variable is introduced for each constraint and the original problem is converted to:

$$\text{Optimise } L = f(x_1, x_2, \ldots, x_n) + \lambda_1[g_1(x_1, x_2, \ldots, x_n) - b_1] + \lambda_2[g_2(x_1, x_2, \ldots, x_n) - b_2] + \ldots\ldots + \lambda_m[g_m(x_1, x_2, \ldots, x_n) - b_m] \qquad (3.9)$$

in which L is a Lagrange function of $x_1, x_2, \ldots, x_n$, λ_1, $\lambda_2, \ldots,$ λ_m, with $n+m$ variables.

Equation above can be expressed in a concise form as

$$\text{Optimise } L = f(x_i) + \sum_{j=1}^{m} \lambda_j[g_j(x_i)],\ i = 1, \ldots, n \qquad (3.10)$$

In Lagrange multiplier method the original constrained problem of n variables and m constraints is converted into an unconstrained one with $n + m$ variables. This problem now can be solved by solution method of unconstrained problem. The condition $\partial L/\partial \lambda_j = 0, j = 1, \ldots, m$ will ensure that the constraints are satisfied at the optimum point.

Example 3.4

Find the maximum of the function $f(X) = 2x_1 + x_2 + 10$

subject to

$g(X) = x_1 + 2x_2^2 = 3$

using the Lagrange multiplier method.

Solution:

The Lagrange function is given by:

$L(X, \lambda) = 2x_1 + x_2 + 10 + \lambda\,(3 - x_1 - 2x_2^2)$

The necessary conditions for the solution of the problem are

$$\frac{\partial L}{\partial x_1} = 2 - \lambda = 0$$

$$\frac{\partial L}{\partial x_2} = 1 - 4\lambda x_2 = 0$$

$$\frac{\partial L}{\partial \lambda} = 3 - x_1 - 2x_2^2 = 0$$

The solution is

$$X^* = \begin{Bmatrix} x_1^* \\ x_2^* \end{Bmatrix} = \begin{Bmatrix} 2.97 \\ 0.13 \end{Bmatrix}$$

$\lambda^* = 2.0$

The application of the sufficiently condition yields

$$\begin{vmatrix} L_{11} - z & L_{12} & g_{11} \\ L_{21} & L_{22} - z & g_{12} \\ g_{11} & g_{12} & 0 \end{vmatrix} = 0$$

$$\begin{vmatrix} -z & 0 & -1 \\ 0 & -4\lambda - z & -4x_2 \\ -1 & -4x_2 & 0 \end{vmatrix} = \begin{vmatrix} -z & 0 & -1 \\ 0 & -8 - z & -0.52 \\ -1 & -0.52 & 0 \end{vmatrix} = 0$$

$0.2704\, z + 8 + z = 0$

$z = -6.2972$

Hence X^* will be a maximum of f with $f^* = f(X^*) = 16.07$.

2. Inequality constraints

Inequality constraints are converted to equality constraints by adding non negative slack variables s_j^2 to constraints with less than equal ($\leq$) sign or by subtracting nonnegative surplus variables s_j^2 from constraints with more than equal ($\geq$) sign. The introduction of s_j^2 instead of s_j has avoided further introduction of constraints $s_j \geq 0$. The Lagrange function is now a function of x_i, $i = 1,\ldots, n$; λ_j, $j = 1, \ldots, m$ and s_j^2, $j = 1,\ldots, m$ if all constraints are inequality constraints. For a minimisation problem, $\partial L/\partial \lambda_j = 0$ will ensure that the constraints are satisfied. From $\partial L/\partial s_j = 0$, we have $2\lambda_j s_j = 0, j = 1,\ldots, m$. Thus, we have either $\lambda_j = 0$ or $s_j = 0$. If $\lambda_j = 0$ for a particular value of j, that constraint is inactive, thus, nonbinding at the optimum point, if $s_j = 0$, that constraint is active, thus, binding at the optimum point. The necessary conditions to be satisfied at the constrained minimum point can be expressed as

$$\frac{\partial f}{\partial x_i}+\sum_{j\in J_1}\lambda_j\frac{\partial g_j}{\partial x_i}=0,\ i=1,\ \ldots.,\ n \tag{3.11}$$

where:

$\lambda_j>0, j\in J_1$

In which J_1 is a set of active constraints. These conditions are known as Kuhn-Tucker conditions. These conditions are necessary but not sufficient to ensure a relative minimum. However, for convex programming problems, which have only one minimum, Kuhn-Tucker conditions are necessary and also sufficient to ensure global minimum.

If the set of active constraints is not known, as generally would be the case in practice, the Kuhn-Tucker conditions can be expressed as

$$\frac{\partial f}{\partial x_i}+\sum_{j\in J_1}\lambda_j\frac{\partial g_j}{\partial x_i}=0,\ i=1,\ \ldots.,\ n \tag{3.12}$$

where: $\lambda_j g_j=0,\ g_j\le 0$, and $\lambda_j>0, j=1,\ \ldots.,\ m$

Example 3.5

Consider a problem:

Minimise $f(x_1,x_2)=(x_1-3)^2+(x_2-8)^2$

subject to

$g_1(x_1,x_2)=-x_1^2+x_2\le 2$

$g_2(x_1,x_2)=3x_1+x_2\le 12$

Solve the problem using Kuhn Tucker conditions.

Solution:

The minimisation problem can be written as:

Minimise $Z=f(x_i)=(x_1-3)^2+(x_2-8)^2$ (E1)

subject to:.

$g_1(x_i)=-x_1^2+x_2-2\le 0$ (E2)

$g_2(x_i)=3x_1+x_2-12\le 0$ (E3)

The Kuhn Tucker conditions are :

$$\frac{\partial f}{\partial x_i} + \lambda_1 \frac{\partial g_1}{\partial x_i} + \lambda_2 \frac{\partial g_2}{\partial x_i} = 0,\ i=1,2$$

Thus,

$2(x_1-3) + \lambda_1(-2x_1) + \lambda_2(3) = 0$ (E₄)

$2(x_2-8) + \lambda_1 + \lambda_2 = 0$ (E₅)

$\lambda_j g_j = 0, j = 1,2$

Gives

$\lambda_1 g_1 = \lambda_1(-x_1^2 + x_2 - 2) = 0$ (E₆)

$\lambda_2 g_2 = \lambda_2(3x_1 + x_2 - 12) = 0$ (E₇)

$g_j \le 0, j = 1,2$

Gives

$g_1(x_i) = -x_1^2 + x_2 - 2 \le 0$ (E₈)

$g_2(x_i) = 3x_1 + x_2 - 12 \le 0$ (E₉)

$\lambda_j \ge 0, j = 1,2$

Gives $\lambda_1 \ge 0$ (E₁₀)

and $\lambda_2 \ge 0$ (E₁₁)

Simultaneous solution of equation (E₄), …, (E₇) will give the values of x_1, x_2, $\lambda_1$and λ_2. The acceptable values will be those that also satisfy equation (E₈),…, (E₁₁).

Now, from equation (E₆) and (E₇), we have $\lambda_1 = 0$ or $g_1 = 0$; and $\lambda_2 = 0$, or $g_2 = 0$. This gives four combinations:

(1) $\lambda_1=0$, and $\lambda_2=0$

(2) $\lambda_1=0$ and $3x_1 + x_2 - 12 = 0$

(3) $\lambda_2=0$ and $-x_1^2 + x_2 - 2 = 0$, and

(4) $3x_1 + x_2 - 12 = 0$ and $-x_1^2 + x_2 - 2 = 0$

(1) λ_1=0, and λ_2=0

Substituting these values of λ_1 and λ_2, equation (E_4) and (E_5) give $x_1 = 3$ and $x_2 = 8$. These values of λ_1, λ_2, x_1 and x_2 satisfy equation (E_8), (E_{10}) and (E_{11}) ; but violate equation (E_9).

(2) λ_1=0 and $3x_1 + x_2 - 12 = 0$

Substituting these values of λ_1= 0 and λ_2 =1, give $x_1 = 1.5$ and $x_2 = 7.5$. These values of λ_1, λ_2, x_1 and x_2 satisfy equation (E_9), (E_{10}) and (E_{11}) ; but violate equation (E_8).

(3) λ_2=0 and $-x_1^2 + x_2 - 2 = 0$,

This condition gives $x_2 = x_1^2 + 2$ and leads to two solutions:

(a) λ_1= -0.2136, λ_2=0, x_1=2.4712 and x_2=8.1068; and

(b) λ_1= 10.3666, λ_2=0, x_1=-2.1947 and x_2=6.8167.

Solution (a) satisfies equation (E_8) and (E_{11}) but violates equation (E_9) and (E_{10}). Solution (b) satisfies equation (E_8), ..., (E_{11}) and gives local minimum solution $Z^* = 28.3851$

(4) $3x_1 + x_2 - 12 = 0$ and $-x_1^2 + x_2 - 2 = 0$

This condition gives also two solutions:

(a) x_1= -5, x_2=27, λ_1= 18.5714, and λ_2=-56.5714; and

(b) x_1= 2, x_2=6, λ_1= 1.4286, and λ_2=2.5714.

Solution (a) satisfies equation (E_8), (E_9) and (E_{10}) but violates equation (E_{11}). Solution (b) satisfies equation (E_8), ..., (E_{11}) and gives local minimum solution $Z^* = 5$.

Thus this optimisation problem has two local optimum solutions:

(1) x_1^*= -2.1947 and x_2^*=6.8167, giving $Z^* = 28.3851$; and

(2) x_1^*= 2 and x_2^*=6, giving $Z^* = 5$.

Note that the solutions 1, 2, and 3(a) are infeasible solutions as they violate one or both constraints. Solution 4(a) satisfies both constraints and thus gives a feasible solution. However, the solution is not optimal. Solution 3(b) and 4(b) give local minimum solutions, with solution 4(b) being global minimum.

Classical optimisation methods have following limitations, i.e.

a. The variable must be continuous. Optimality conditions cannot be formulated when the function is non differentiable or consists of discrete sets. It cannot also define optimal solution when the function is piecewise linear.

b. It is not possible, a priori, to distinguish between points giving maxima, minima or saddle points; unless secondary criteria are applied.

c. It is not possible to locate optimal points that occur at points where the optimality conditions are not satisfied such as the boundary points of a range of variables.

d. Solutions to optimality criteria may be unstable when the differential equations are written as difference equations for obtaining solution with a computer. The rate of change of Z with respect to any variable x_i, i.e. $\Delta Z/\Delta x_i$ will be very small near the optimum, $\Delta Z/\Delta x_i \approx 0$. This derivative will then be very sensitive to round off errors that occur in the evaluation of Z. The computation of the optimum value will then be inherently inaccurate estimate of the true optimum.

e. It fails to provide practical means to define the sensitivity of the solution to changes in the values of the variables.

3.4 NUMERICAL METHODS

Since it may not be possible to tackle an NLP problem directly by manipulating the constraints, the NLP solution methods that based on numerical methods of optimisation is applied. In the classical optimisation method, the optimum values of decision variables are achieved and then also the optimal solution. However, in numerical methods, the opposite procedure is followed. First, trial solution is selected and check for optimality. If it is not optimal, it successively improved to obtain the optimal solution.

In optimisation problem of water distribution network, the objective function is to minimise the cost of project. In this case, the minimisation problem can be broadly classified in two categories:

1. Unconstrained minimisation problems and
2. Constrained minimisation problem.

Even though practical design problems are usually constrained, they can be converted to unconstrained so that powerful and convenient methods of unconstrained minimisation can be used. Furthermore, unconstrained minimisation methods provide basic understanding that is very helpful to study constrained minimisation methods. The various methods of solving NLP problems are presented in tree diagram in Figure 3-3.

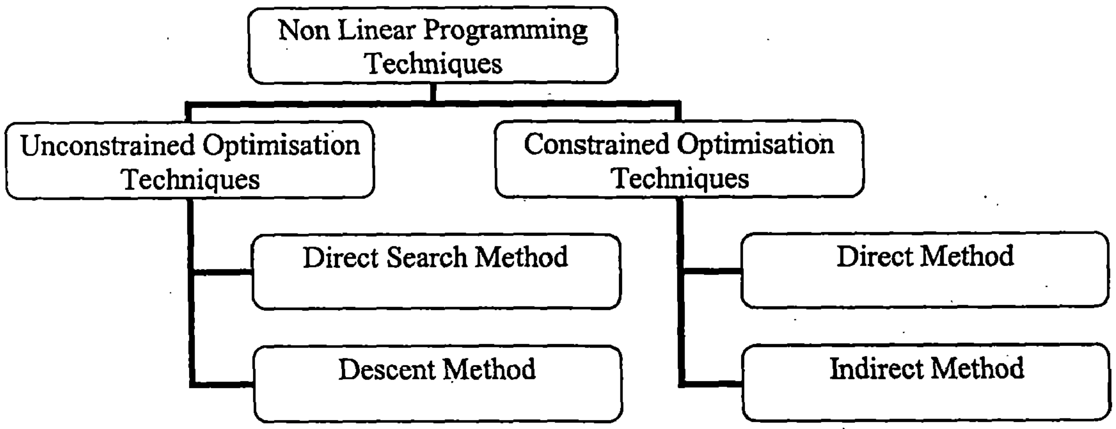

Non Linear Programming Techniques
Unconstrained Optimisation Techniques
Constrained Optimisation Techniques
Direct Search Method
Descent Method
Direct Method
Indirect Method

Figure 3-3 Division of Non Linear programming Techniques

3.4.1. Unconstrained minimisation

Methods available for solving unconstrained minimisation problems can be classified in two categories, i.e. Direct Search method and Descent method

Table 3-2 Unconstrained Minimisation Methods

Direct Search Method	***Descent Method***
Random Search Method	Steepest Descent (Cauchy) Method
Grid Search Method	Fletcher Reeves Method
Univariate Method	Newton's Method
Pattern Search Method	Marquardt Method
- Powell's Method	Quasi-NewtonMethod
- Hooke-Jeeves Method	- Davidon-Fletcher –Powell Method
Rosenbrock's Method	- Broyden-Fletcher-Goldfarb-Shanno
Simplex Method	Method

3.4.1.1 Direct Search Method

Direct Search Methods require only objective function evaluations, and do not need the partial derivatives of the function in the finding the minimum. Therefore, they also called non gradient methods. Some of them will be reviewed as follows:

a. Univariate Method.

In univariate method, only one design variables is changed at a time while other variables are held constant. Thus, from a starting solution of an n-variable problem, any n-1 design variables are held constant, the remaining variables is changed and its improved value is obtained. Similarly, other design variables are also improved sequentially to complete one optimisation iteration. Iterative procedure is continued until the optimal solution of desired accuracy is obtained.

Univariate method is quite simple and can be easily implemented. However, it has a tendency to oscillate with steadily decreasing progress towards the optimum; thus, it does not converge rapidly. In some cases it may not even converge.

b. Pattern Search Method

In univariate method, the minimum is searched along direction parallel to the coordinate axes. However, convergence is slow as the optimum point is approached. In pattern search methods, the directions of search are changed favourably so that convergence characteristics are improved. Hooke and Jeeves method explores moves in different directions to asses the local behaviour of the objective function; and decides the pattern search direction. Powell's method is a widely used direct search method based on conjungate directions. It has quadratic convergence; thus, it rapidly converges to the optimal solution.

c. Rosenbrock's Method of Rotating Coordinates

In this method, the first axis is oriented toward its best local direction and all other axes are made mutually orthogonal and normal to the first one. Since the coordinate system can be suitably rotated it can follow curved and steep valleys, thereby improving convergence.

d. Simplex Method

Simplex method in NLP is based on the concept of simplex, a geometric figure formed by set of n+1 points in n-dimensional space (In two dimensions the simplex is triangle, and in three dimensions it is a tetrahedron). For an n-dimensional problem, n+1 points are selected to form a general simplex. The values of the objective function at the n+1 vertices of the simplex are then compared and the vertex having the maximum value of the objective function in a minimisation problem is replaced by another vertex through operations of reflection, and expansion or contraction, to obtain revised simplex. If H is the point corresponding to the highest value, we can expect vertex R, obtained by reflection of H on the opposite face, to have the smallest value. (Reflection point R lies on the other side of G_n the centroid of all other n points except point H). When $f(R) < f(H)$, to enhance convergence, we may shift point R to a further point N through an operation termed expansion by moving along the direction from G_n to R, using expansion coefficient >1 (ratio of distance between N and G_n and distance between R and G_n). If $f(N) < f(R)$ we select point N, however if $f(N) > f(R)$ the expansion process is not successful and we retain point R to form revised simplex. If the reflection process gives $f(R) > f(H)$ we contract the simplex through an operation termed contraction using contraction coefficient (ratio of distance between N and G_n and distance between R and G_n) lying between 0 and 1. The iterative procedure of revising simplex is continued until convergence is reached and theoretically the simplex collapses into a point, the optimal point. In practice, however, the method is assumed to have converged when the standard deviation of the function at the n+1 vertices of the current simplex is smaller than some preselected small quantity. When the convergence is satisfied, the centroid of the latest simplex is taken as the optimum point.

3.4.1.2. Descent Method

The descent methods require, in addition to objective function, evaluation of the first and possibly higher order derivatives of the objective function. The derivatives provide more information about the function being minimised; therefore, descent methods are more efficient than the direct search methods. The descent methods are also known as gradient methods.

The general procedure of minimisation of unconstrained, and also constrained problem involves following steps:

Step 1. Start with an initial trial point ${}_1x_i$, $i = 1,2,\ldots, n$.

Step 2. Find a suitable direction ${}_kS_i$, i=1,2,…,n (the prefixing subscript k represent iteration number, 1 to start with) points in the general direction of minimum.

Step 3. Find a suitable step length ${}_k\lambda$ to move along the direction ${}_kS_i$, i=1,2,…,n

Step 4. Obtain new approximation ${}_{k+1}x_i$ given by

$${}_{k+1}x_i = {}_kx_i + {}_k\lambda \,.\, {}_kS_i\,,\ i=1,2\ldots,n \tag{3.13}$$

Step 5. Test whether ${}_{k+1}x_i$, i =1,…, n is optimum. If optimum, stop the procedure, otherwise set new k=k+1 and repeat step 2 onwards.

The iterative procedure is valid for unconstrained as well as constrained optimisation problems. In iterative procedure it is necessary to:

1. Select an initial point
2. Select suitable direction
3. Select suitable step length
4. Decide a criterion for termination of the iterative procedure.

The efficiency of the optimisation procedure would depend on the efficiency of selection of these parameters.

1. Selection of initial trial point

Selection of an initial trial point, i.e. selection of an initial feasible solution does not pose any problem in engineering. For example, in water distribution network, larger diameter can be adopted so that head losses are small and water would reach all demand nodes. Even tough the iterative procedure will work from any feasible solution, it would be better if the initial solution were close to the minimum solution so that the number of iterations in the iterative procedure is reduced.

2. Select suitable direction

The direction of search can be decided by several methods. These methods use gradient of the objective function ∇f defined as:

$$\nabla f = \begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \\ .. \\ \partial f / \partial x_n \end{bmatrix} \tag{3.14}$$

The gradient is an n-component vector having an important property that the function increases at the fastest rate along the gradient direction. Thus, the gradient direction is the direction of the steepest ascent. This property is a local property, and the direction of the steepest ascent generally varies from point to point. However, by moving in steps, it is possible to reach local maximum solution.

Negative of the gradient vector, $-\nabla f$ denotes the direction of the steepest descent and it used to determine the search direction in steepest descent method for minimisation of the objective function. Steepest descent method has several merits and limitations; and several approaches have been suggested to improve convergence characteristics of the steepest descent method.

In a constrained minimisation problem, the direction can be found such that:

a) a small move in that direction does not violate any constraint;

b) the value of the objective function can be reduced in that direction.

A direction satisfying property (a) is called a feasible direction; while that satisfying both properties (a) and (b) is called a usable feasible direction. If the trial point ${}_kx_i$, $i=1,\ldots, n$ lies in the interior of the feasible region, the usable direction is given as:

$${}_kS_i = -\nabla f({}_kx_i) \tag{3.15}$$

the direction of the steepest descent. However, when the trial point lies on the boundary of the feasible region, one or more of the constraints are critical. If the critical constraints are linear, the search direction lies in the constraint surface, but if the critical constraints are nonlinear, the search direction satisfying both properties (a) and (b) has to be found by trial and error.

3. Select suitable step length

After deciding the search direction ${}_kS_i$ at any point ${}_kx_i$, we have to determine suitable step length ${}_k\lambda$ to obtain the next point ${}_{k+1}x_i$. There are several ways of computing the step length. One method is to determine optimal step length ${}_k\lambda^*$ which minimises $f({}_kx_i + {}_k\lambda_k S_i)$ such that the new point given by ${}_{k+1}x_i$ lies in the feasible region. Another method is to choose the step length by trial and error so that the objective function reduces withou violating any of the constraints. If the trial step length increases the value of the objective function the step is reduced to half. If the trial step length violates a constraint, the reduced step length can be optimally selected so that the violated constraint is just justified and becomes critical. Since no constraint will become exactly zero while working with a computer, a constraint can be considered active if

$| g_j(x_i) - b_j | \le \varepsilon_g, i = 1,\ldots,n; j=1,\ldots,m$ (3.16)

in which ε_g is a small number of the order of 10^{-2} to 10^{-6}.

4. Decide a criterion for termination of the iterative procedure.

The iterative procedure can be stopped when it converges to the optimal solution. The optimality of the solution can be checked by testing the Kuhn-Tucker conditions. Alternatively, we can perturb the optimal design variable x_i^*, $i=1,\ldots,n$; by changing each of the design variables, one at a time, by a small amount, positive as well as negative, and verify that Z^* does not decrease in a minimisation problem without violating any of the constraints.

In the iterative procedure we can check convergence to the optimal (minimum) solution by testing

$$\frac{{}_k f(x_i) - {}_{k+1} f(x_i)}{{}_k f(x_i)} \le \varepsilon_f, \; i=1,\ldots,n \qquad (3.17)$$

in which ε_f is a predetermined, small functional change value of the order of 10^{-2} to 10^{-6}. Another convergence criterion can be

$| {}_k x_j - {}_{k+1} x_j | \le \varepsilon_x, i = 1,\ldots,n$ (3.18)

in which ε_x is a predetermined small value of the order of 10^{-2} to 10^{-6}.

During the iteration procedure, if all given constraints become active at a point, the solution at this point is optimal and the iterative procedure is stopped.

Several variations of the general descent method have been suggested for minimisation of unconstrained problems. The steepest descent method uses the negative of the gradient vector as a suitable direction. Conjugate gradient method (Fletcher-Reeves method), Quasi Newton method and variable metric method (Davidon-Fletcher-Powell method) are some variations of descent method with improved convergence characteristics.

3.4.2 Constrained Minimisation

Method available for constrained minimisation problems can be classified into two categories:

(1) Direct method and

(2) Indirect Method

In Direct Method, the constraints are handled explicitly while in most of indirect methods, the constrained minimisation is changed to an unconstrained one.

Table 3-3 Constrained Optimisation Techniques

Direct Methods	**Indirect Methods**
Random Search Method	Transformation of variables technique
Heuristic Search Method	Sequential unconstrained minimisation techniques
- Complex method	- Interior penalty function method
Objective and constraint approximation Methods	- Augmented Lagrange multiplier method
- Sequential Linear Programming Method	
- Sequential Quadratic Programming Method	
Method of Feasible Directions	
- Zoutendijk's Method	
- Rosen's gradient projection method	
Generalised Reduced Gradient Method	

3.4.2.1 Direct Method

In Direct Method, there are four categories that available:

1) Heuristic Search Method.
2) Objective and Constraint Approximation Methods
3) Method of Feasible Direction
4) Generalised Reduced Gradient Method

Those methods are described as follows:

1) Heuristic Search Method.

Heuristic Search Method are mostly based on intuition and do not have much theoretical support. However, they are simple and applicable to specific problems.

An example of Heuristic method is Complex method that is suggested by Box (1965). This method is an extension of simplex method of unconstrained optimisation. It does not require the derivative of the objective function and constraints, and thus is comparatively very simple. However, it cannot handle nonlinear equality constraints but can handle side constraints in which the decision variables are restricted by lower and upper bounds.

In simplex method for n-dimensional unconstrained minimisation problem a simplex (geometrical figure) with n+1vertces is generated; while in complex method for n-dimensional constrained minimisation problem, a complex geometrical figure with $k \geq n+1$ vertices is generated, each vertex representing a feasible solution satisfying all constraints. The values of the objective function at all k vertices are obtained, compared and the vertex H with the value of the function is located. It is then reflected in the opposite face to locate reflection point R. If R is a feasible point and if $f(R)<f(H)$, point R replaces point H to obtain revised complex. If at R, $f(R)>f(H)$ or any of the constraints is violated, point R is the remaining k-1 vertices so that point N gives a feasible solution and $f(N) < f(H)$. If such a point N cannot be located, instead of point H, a point with next largest value of the function is selected for the reflection procedure. The procedure of finding point H with highest value of the function in the current complex and replacing it by point R or point N to get revised complex is continued until convergence is achieved. The convergence is said to be achieved when either the complex shrinks to a specified small size or the standard deviation of the function value becomes sufficiently small..

2) Objective and Constraint Approximation Methods

There are two examples of objective and constraint approximation methods, i.e.:

a) Sequential linier programming method

b) Sequential quadratic programming method

Those methods are described as follows:

2.a Sequential linier programming method

Sequential Linear Programming (SLP) method also known as Cutting Plane method, was originally presented by Cheney and Goldstein (1959). It linearises the constraints at selected point through Taylor's series. These linearised constraints, which approximate the feasible region by linearised envelopes, are then used to solve the LP problem. The iterative procedure is continued to find a sufficiently accurate solution. This method is efficient and can use the available LP algorithms. However, all the optimum solutions of the approximating LP problems lie in the infeasible region. Thus, the final optimum solution depends upon the tolerance limit and may require adjustment in practice.

The SLP algorithm can be stated as follows:

1. Start with an initial point and set the iteration number as i=1. The point X_1 need not be feasible.
2. Linearise the objective and constraint function about the point X_i as

   $f(X) \approx f(X_i) + \nabla f(X_i)^T (X\text{-}X_i)$ (3.19)

   $g_j(X) \approx g(X_i) + \nabla g_j(X_i)^T (X\text{-}X_i)$ (3.20)

   $h_k(X) \approx h_k(X_i) + \nabla h_k(X_i)^T (X\text{-}X_i)$ (3.21)
3. Formulate the approximating linear programming problem as

   Minimise $f(X_i) + \nabla f_i^T (X\text{-}X_i)$ (3.22)

   subject to

   $g(X_i) + \nabla g_j(X_i)^T (X\text{-}X_i) \le 0,\ j=1,2,\ldots,m$

   $h_k(X_i) + \nabla h_k(X_i)^T (X\text{-}X_i) \le 0,\ k=1,2,\ldots,p$
4. Solve the approximating LP problem to obtain the solution vector X_{i+1}
5. Evaluate the original constraints at X_{i+1}, that is , find

   $g_j(X_{i+1}), j=1,2,..,f$ and $h_k(X_{i+1}), k=1,2,\ldots,p$

   If $g_j(X_{i+1}) \le \varepsilon$ for $j=1,2,..,m$ and $|h_k(X_{i+1})| \le \varepsilon$, $k=1,2,..,p$, where ε is a prescribed small positive tolerance, all the original constraints can be assumed to have been satisfied. Hence stop the procedure by taking

   $X_{\text{opt}} \approx X_{i+1}$

   If $g_j(X_{i+1}) > \varepsilon$ for some j, or $|h_k(X_{i+1})| > \varepsilon$ for some k, find the most violated constraint, for example, as

   $g_k(X_{i+1}) = \max_j [\, g_j(X_{i+1})]$ (3.23)

   Relinearise the constraint $g_k(X) \le 0$ about the point X_{i+1} as

   $g_k(X) \approx g_k(X_{i+1}) + \nabla g_k(X_{i+1})^T (X\text{-}X_{i+1}) \le 0$ (3.24) and add this as the $(m+1)^{\text{th}}$ inequality constraints to the previous LP problem.

6. Set the new iteration number as $i=i+1$, the total number of constraints in the new approximating LP problems as $f+1$ inequalities and p equalities, and go to step 4.

The sequential linear programming method has several advantages:

1. It is an efficient technique for solving convex programming problems with nearly linear objective and constraint functions
2. Each of the approximating problems will be a LP problem and hence can be solved quite efficiently. More over, any two consecutive approximating LP problems differ by only one constraint, and hence the dual simplex method can be used to solve the sequence of approximating LP problems much more efficiently.
3. The method can easily be extended to solve integer programming problems. In this case, one integer LP problem has to be solved in each stage.

Example 3.6

Minimise $f(x_1,x_2) = x_1 - x_2$

subject to

$g_1(x_1,x_2) = 3x_1^2 - 2x_1x_2 + x_2^2 - 1 \le 0$

using Sequential Linear Programming Method. Take the convergence limit as $\varepsilon = 0.02$.

Note: Since the constraint boundary represent an ellipse, the problem is convex programming problem. From graphical representation, the optimum solution of the problem can be identified as $x_1^* = 0$, $x_2^* = 1$ and $f_{\min} = -1$.

Solution:

Step 1,2,3: Although we can start the solution from any initial point X_1, to avoid the possible unbounded solution, we first take the bounds on x_1 and x_2 as $-2 \le x_1 \le 2$ and $-2 \le x_2 \le 2$ and solve the following LP problem:

Minimise $f = x_1 - x_2$ (E$_1$)

Subject to

$-2 \le x_1 \le 2$

$-2 \leq x_2 \leq 2$

The solution of this problem can be obtained as

$$X = \begin{bmatrix} -2 \\ 2 \end{bmatrix} \text{ with } f(X) = -4$$

Step 4: Since we have solved one LP problem, we can take

$$X_{i+1} = X_2 = \begin{Bmatrix} -2 \\ 2 \end{Bmatrix}$$

Step5: Since $g_1(X_2) = 23 > \varepsilon$, we linearise $g_1(X)$ about point X_2 as

$$g_1(X) \approx g_1(X_2) + \nabla g_1 (X_2)^T (X - X_2) \leq 0 \qquad (E_2)$$

as

$$g_1(X_2)=23, \left.\frac{\partial g_1}{\partial x_1}\right|_{x_2} = (6x_1 - 2x_2)|_{x_2} = -16$$

$$\left.\frac{\partial g_1}{\partial x_2}\right|_{x_2} = (-2x_1 + 2x_2)|_{x_2} = 8$$

So the above equation becomes

$g_1(X) = -16x_1 + 8x_2 - 25 \leq 0$

By adding this constraint to the previous LP problem, the new LP problem becomes:

Minimise $f = x_1 - x_2$ $\qquad (E_3)$

subject to:

$-2 \leq x_1 \leq 2$

$-2 \leq x_2 \leq 2$

$-16x_1 + 8x_2 - 25 \leq 0$

Step 6: Set the iteration number as $i = 2$ and go to step 4

Step 4: Solve the approximating LP problem stated in equation (E_3) and obtain the solution

$$X_3 = \begin{Bmatrix} -0.5625 \\ 2.0 \end{Bmatrix} \text{ with } f_3 = f(X_3) = -2.5625$$

This procedure is continued until the specified convergence criterion, $g_1(X_i) \leq \varepsilon$, in step 5 is satisfied. The computation result are summarised in Table 3-4.

Table 3-4 Results for Example 3.6

Iteration Number	New Linearised Constraint Considered	Solution of the Approximating LP Problem X_{i+1}	$f(X_{i+1})$	$g(X_{i+1})$
1	$-2 \le x_1 \le 2$ and $-2 \le x_2 \le 2$	(-2.0,2.0)	-4.00000	23.00000
2	$-16.0x_1+8.0x_2-25 \le 0$	(-0.56250,2.00000)	-2.56250	6.19922
3	$-7.375x_1+5.12x_2-8.19922 \le 0$	(0.27870,2.00000)	-1.72193	2.11978
4	$-2.33157x_1+3.44386x_2-4.11958 \le 0$	(-0.52970,0.83759)	-1.36730	1.43067
5	$-4.85341x_1+2.73459x_2-3.43067 \le 0$	(-0.05314,1.16024)	-1.21338	0.47793
6	$-2.63930x_1+2.42675x_2-2.47792 \le 0$	(0.42655,1.48490)	-1.05845	0.48419
7	$-0.41071x_1+2.11690x_2-2.48420 \le 0$	(0.17058,1.20660)	-1.03603	0.13154
8	$-1.38975x_1+2.07205x_2-2.13155 \le 0$	(0.01829,1.04098)	-1.02269	0.04656
9	$-1.97223x_1+2.04538x_2-2.04657 \le 0$	(-0.16626,0.84027)	-1.00653	0.06838
10	$-2.67809x_1+2.01305x_2-2.06838 \le 0$	(-0.07348,0.92972)	-1.00321	0.01723

2.b Sequential quadratic programming method

This method has a theoretical basis that is related to:

(a) the solution of a set of nonlinear equations using Newton's method, and

(b) the derivation of simultaneous nonlinear equations using Kuhn-Tucker conditions to the Lagrangian of the constrained optimisation problem.

(1) Derivation

Consider a nonlinear optimisation problem with only equality constraints as:

Find X which minimises $f(X)$ (3.25)

subject to

$h_k(X) = 0, \quad k = 1,2,\dots,p.$

The Lagrange function, $L(X,\lambda)$, corresponding to the problem of Equation (3.25) is given by

$$L = f(X) + \sum_{k=1}^{p} \lambda_k h_k \tag{3.26}$$

where λ_k is the Lagrange multiplier for the kth equality. The Kuhn-Tucker necessary conditions can be stated as

$\nabla L = 0$ or $\nabla f + \sum_{k=1}^{p} \lambda_k h_k = 0$ or $\nabla f + [A]^{\mathrm{T}}\lambda = 0$ (3.27)

$h_k(X) = 0, \quad k = 1,2,\ldots,p$ (3.28)

where $[A]$ is an n x p matrix whose kth column denotes the gradient of the function h_k. Equation (3.27) and (3.28) represent a set of n+p nonlinear equations in n+p unknowns (x_i, i = 1,..,n and λ_k , k =1,.., p). These nonlinear equations can be solved using Newton's method. For convenience Equation (3.27) and (3.28) can be rewrite as

$F(Y) = 0$ (3.29)

where

$$F = \begin{Bmatrix} \nabla L \\ h \end{Bmatrix}_{(n+p)x1}, \quad Y = \begin{Bmatrix} X \\ \lambda \end{Bmatrix}_{(n+p)x1}, \quad 0 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}_{(n+p)x1} \qquad (3.30)$$

According to Newton's method, the solution of Equation (3.29) can be found iteratively as

$Y_{j+1} = Y_j + \Delta Y_j$ (3.31)

with

$[\nabla F]_j^T \Delta Y_j = -F(Y_j)$ (3.32)

where Y_j is the solution at the start of jth iteration and ΔY_j is the change in Y_j necessary to generate the improved solution, Y_{j+1} , $[\nabla F]_j = [\nabla F(Y_j)]$ is the (n +p) x (n+p) Jacobian matrix of the nonlinear equations whose ith column denotes the gradient of the function $F_i(Y)$ with respect to the vector Y. By substituting Equation (3.29) and (3.30) into Equation (3.32), we obtain

$$\begin{bmatrix} [\nabla^2 L] & [H] \\ [H]^T & [0] \end{bmatrix}_j \begin{Bmatrix} \Delta X \\ \Delta \lambda \end{Bmatrix}_j = - \begin{Bmatrix} \nabla L \\ h \end{Bmatrix}_j \qquad (3.33)$$

$\Delta X_j = X_{j+1} - X_j$ (3.34)

$\Delta \lambda_j = \lambda_{j+1} - \lambda_j$ (3.35)

where:

$[\nabla^2 L]_{nxn}$ denotes the Hessian matrix of the Lagrange function. The first set of equation in Equation (3.33) can be written separately as

$$[\nabla^2 L]_j \, \Delta X_j + [H]_j \, \Delta\lambda_j = -\nabla L_j \tag{3.36}$$

Using Equation (3.35) for $\Delta\lambda_j$ and Equation (3.27) for ∇L_j, Equation (3.36) can be expressed as

$$[\nabla^2 L]_j \, \Delta X_j + [H]_j \, (\lambda_{j+1} - \lambda_j) = -\nabla f_j - [H]_j^T \lambda_j \tag{3.37}$$

which can be simplified to obtain

$$[\nabla^2 L]_j \, \Delta X_j + [H]_j \, \lambda_{j+1} = -\nabla f_j \tag{3.38}$$

Equation (3.38) and second set of equations in (3.33) can now be combined as

$$\begin{bmatrix} [\nabla^2 L] & [H] \\ [H]^T & [0] \end{bmatrix}_j \begin{Bmatrix} \Delta X_j \\ \lambda_{j+1} \end{Bmatrix} = - \begin{Bmatrix} \nabla f_j \\ h_j \end{Bmatrix} \tag{3.39}$$

Equations (3.39) can be solved to find the change in the design vector ΔX_j and the new values of the Lagrangian multipliers λ_{j+1}. The iterative process indicated by Equation (3.39) can be continued until convergence is achieved.

Now consider the following quadratic programming problem:

Find ΔX that minimises the quadratic objective function

$$Q = \nabla f^T \Delta X + \tfrac{1}{2} \Delta X^T [\nabla^2 L] \Delta X \tag{3.40}$$

subject to the linear equality constraints

$$h_k + \nabla h_k^T \Delta X = 0, \quad k = 1,2, \ldots, p \text{ or } h + [H]^T \Delta X = 0 \tag{3.41}$$

The Lagrange function, L, corresponding to the problem of Equation (3.40) and (3.41) is given by

$$L = \nabla f^T \Delta X + \tfrac{1}{2} \Delta X^T [\nabla^2 L] \Delta X + \sum_{k=1}^{p} \lambda_k (h_k + \nabla h_k^T \Delta X) \tag{3.42}$$

where λ_k is the Lagrange multiplier associated with the kth equality constraint.

The Kuhn-Tucker necessary condition can be stated as

$$\nabla f + [\nabla^2 L] \Delta X + [H] \lambda = 0 \tag{3.43}$$

$$h_k + \nabla h_k^T \Delta X = 0, \; k = 1,2,\ldots,p \tag{3.44}$$

Equation (3.43) and (3.44) can be identified to be same as Equation (3.39) in matrix form. This shows that the original problem of equation (3.25) can be solved iteratively by solving the quadratic programming problem defined by Equation

(3.40) and (3.41). In fact, when inequality constraints are added to the original problem, the quadratic programming problem of Equation (3.40) and (3.41) becomes:

Find X that minimises $Q = \nabla f^T \Delta X + \frac{1}{2} \Delta X^T [\nabla^2 L] \Delta X$ (3.45)

subject to:

$g_j + \nabla g_j^T \Delta X \leq 0, \; j = 1,2,\ldots, m$

$h_k + \nabla h_k^T \Delta X = 0, \quad k = 1,2,\ldots, p$

with Lagrange function given by

$$L = f(X) + \sum_{j=1}^{m} \lambda_k g_j(X) + \sum_{k=1}^{p} \lambda_{m+k} h_k(X) \tag{3.46}$$

(2) Solution Procedure

As in case of Newton's method of unconstrained minimisation, the solution vector ΔX in Equation (3.45) is treated as the search direction, *S*, and the quadratic programming subproblem (in terms of the design vector *S*) is restated as:

Find *S* which minimises $Q(S) = \nabla f(X)^T S + \frac{1}{2} S^T [H] S$ (3.47)

subject to:

$\beta_j g_j(X) + \nabla g_j(X)^T S \leq 0, \; j = 1,2,\ldots, m$

$\beta h_k + \nabla h_k(X)^T S = 0, \quad k = 1,2,\ldots, p$

where [*H*] is a positive definite matrix that is taken initially as the identity matrix and is updated in subsequent iterations so as to converge to the Hessian matrix of the Lagrange function of Equation (3.46), and β_j and β are constants used to ensure that the linearised constraints do not cut off the feasible space completely. Typical values of these constants are given by

$$\beta \approx 0.9; \; \beta_j = \begin{cases} 1 & \text{if } g_j(X) \leq 0 \\ \beta & \text{if } g_j(X) \geq 0 \end{cases} \tag{3.48}$$

The subproblem of Equation (3.47) is a Quadratic programming problem and can be solved by any available methods. Once the search direction, S is found by solving the problem in Equation (3.47), the design vector is updated as

$$X_{j+1} = X_j + \alpha^* S \tag{3.49}$$

where α* is the optimal step length along the direction S found by minimising the function (using exterior penalty function approach):

$$\phi = f(X) + \sum_{j=1}^{m} \lambda_k (\max[0, g_j(X)]) + \sum_{k=1}^{p} \lambda_{m+k} |h_k(X)| \tag{3.50}$$

with

$$\lambda_j = \begin{cases} |\lambda_j|, j = 1,2,.., m+p \text{ in first iteration} \\ \text{Max } \{|\lambda_j|, 0.5(\lambda_j', |\lambda_j|)\} \text{ in subsequent iteration} \end{cases} \tag{3.51}$$

and $\lambda_j' = \lambda_j$ of the previous iteration. The one dimensional step length α* can be found by any of methods related to one dimensional optimisation.

Once X_{j+1} is found from Equation (3.47) for next iteration the Hessian matrix $[H]$ is updated to improve the quadratic approximation in Equation (3.98). Usually, a modified BFGS formula, given below, is used for this purpose

$$[H_{i+1}] = [H_i] - \frac{[H_i]P_i P_i^T [H_i]}{P_i^T [H_i] P_i} + \frac{\gamma\gamma^T}{P_i^T P_i} \tag{3.52}$$

$$P_i = X_{i+1} - X_i \tag{3.53}$$

$$\gamma = \theta.Q_i + (1-\theta)[H_i]P_i \tag{3.54}$$

$$Q_i = \nabla_x L(X_{i+1}, \lambda_{i+1}) - \nabla_x L(X_i, \lambda_i) \tag{3.55}$$

The value of θ will be:

a) 1.0 if $P_i^T Q_i \geq 0.2 P_i^T [H_i] P_i$, or (3.56)

b) $\dfrac{0.8 P_i^T [H_i] P_i}{P_i^T [H_i] P_i - P_i^T Q_i}$ if $P_i^T Q_i < 0.2 P_i^T [H_i] P_i$

where L is given by Equation (3.46) and the constants 0.2 and 0.8 in Equation (3.56) can be changed, based on numerical experience.

Example 3.7

Find the solution of the problem

Minimise $f(X) = 0.1\,x_1 + 0.05773$ (E₁)

subject to

$g_1(X) = \frac{0.6}{x_1} + \frac{0.3464}{x_2} - 0.1 \le 0$ (E₂)

$g_2(X) = 6 - x_1 \le 0$ (E₃)

$g_3(X) = 7 - x_2 \le 0$ (E₄)

using the sequential quadratic programming technique.

Solution:

Let the starting point be $X_1 = (11.8765\ \ 7.0)^T$ with $g_1(X_1) = g_3(X_1) = 0$, $g_2(X_1) = -5.8765$, and $f(X_1) = 1.5917$. The gradients of the objective and constraint functions at X_1 are given by

$$\nabla f(X_1) = \begin{Bmatrix} 0.1 \\ 0.05733 \end{Bmatrix}$$

$$\nabla g_1(X_1) = \begin{Bmatrix} \dfrac{-0.6}{x_1^2} \\ \dfrac{-0.3464}{x_2^2} \end{Bmatrix}_{X_1} = \begin{Bmatrix} -0.004254 \\ -0.007069 \end{Bmatrix}$$

$$\nabla g_2(X_1) = \begin{Bmatrix} -1 \\ 0 \end{Bmatrix}, \quad \nabla g_3(X_1) = \begin{Bmatrix} 0 \\ -1 \end{Bmatrix}$$

We assuming the matrix $[H_1]$ to be the identity matrix and hence the objective function of Equation (3.47) becomes

$Q(S) = 0.1\, s_1 + 0.05773 s_2 + 0.5 s_1^2 + 0.5 s_2^2$ (E₅)

Equation (3.48) gives $\beta_1 = \beta_3 =$ since $g_1 = g_3 = 0$ and $\beta_2 = 1.0$ since $g_2 < 0$, hence the constraint of Equation (3.47) can be expressed as

$g_1 = -0.004254\, s_1 - 0.007069\, s_2 \le 0$ (E₆)

$g_2 = -5.8765 - s_1 \le 0$ (E₇)

$g_3 = -s_2 \le 0$ (E₈)

We solve this quadratic programming problem directly with the use of the Kuhn – Tucker conditions. The Kuhn Tucker conditions are given by

$\frac{\partial Q}{\partial s_1} + \sum_{j=1}^{3} \lambda_j \frac{\partial g_j}{\partial s_1} = 0$ (E₉)

$$\frac{\partial Q}{\partial s_2}+\sum_{j=1}^{3}\lambda_j\frac{\partial g_j}{\partial s_2}=0 \qquad (E_{10})$$

$$\lambda_j g_j = 0 \quad ,j=1,2,3 \qquad (E_{11})$$

$$g_j \leq 0 \quad ,j=1,2,3 \qquad (E_{12})$$

$$g_j \geq 0 \quad ,j=1,2,3 \qquad (E_{13})$$

Equations (E_9) and (E_{10}) can be expressed, in this case, as

$$0.1+s_1-0.004254\,\lambda_1-\lambda_2=0 \qquad (E_{14})$$

$$0.05773+s_2-0.007069\,\lambda_1-\lambda_3=0 \qquad (E_{15})$$

By considering all possibilities of active constraints, we find that the optimum solution of the quadratic programming problem is given by

$s_1{}^* = -0.04791$, $s_2{}^* = 0.02883$, $\lambda_1{}^* = 12.2450$, $\lambda_2{}^* = 0$, $\lambda_3{}^* = 0$

The new design vector, X, can be expressed as

$$X = X_1 + \alpha S = \begin{Bmatrix} 11.8765-0.04791\alpha \\ 7.0+0.02883\alpha \end{Bmatrix}$$

where α can be found by minimising the function ϕ in Equation (3.50)

$$\phi = 0.1(11.8765-0.04791\alpha)+0.05773\,(7.0+0.02883\alpha)$$
$$+\,12.2450\left(\frac{0.6}{11.8765-0.04791\alpha}+\frac{0.3464}{7.0+0.02883\alpha}-0.1\right)$$

By using quadratic interpolation technique (unrestricted search method can also be used for simplicity), we find that ϕ attains its minimum value of 1.48 at $\alpha^* = 64.93$, which corresponds to the new design vector

$$X_2 = \begin{Bmatrix} 8.7657 \\ 8.8719 \end{Bmatrix}$$

with $f(X_2) = 1.38874$ and $g_1(X_2) = 0.0074932$ (violated slightly). Next we update the matrix $[H]$ using Equation (3.52) with

$$L = 0.1x_1 + 0.05773x_2 + 12.2450\left(\frac{0.6}{x_1}+\frac{0.3464}{x_2}-0.1\right)$$

$$\nabla_x L = \begin{Bmatrix} \dfrac{\partial L}{\partial x_1} \\ \dfrac{\partial L}{\partial x_2} \end{Bmatrix} \text{ with } \frac{\partial L}{\partial x_1} = 0.1 - \frac{7.3470}{x_1^{\,2}}4$$

and $\frac{\partial L}{\partial x_2}$=0.05773 - $\frac{4.2417}{x_2^{\,2}}$

$$P_1 = X_2 - X_1 = \begin{Bmatrix} -3.1108 \\ 1.8719 \end{Bmatrix}$$

$$Q_1 = \nabla_x L(X_1) = \begin{Bmatrix} 0.00438 \\ 0.00384 \end{Bmatrix} - \begin{Bmatrix} 0.04791 \\ -0.02883 \end{Bmatrix} = \begin{Bmatrix} -0.04353 \\ 0.03267 \end{Bmatrix}$$

$P_1^T\,[H_1]\,P_1 = 13.1811,\ P_1^T Q_1 = 0.19656$.

This indicates that $P_1^T Q_1 < 0.2\ P_1^T\,[H_1]\,P_1$, and hence θ is computed using Equation (3.56), as

$$\theta = \frac{(0.8)(13.1811)}{13.1811 - 0.19656} = 0.81211$$

$$\gamma = \theta Q_1 + (1-\theta)\,[H_1]\,P_1 = \begin{Bmatrix} 0.54914 \\ -0.32518 \end{Bmatrix}$$

Hence

$$[H_2] = \begin{bmatrix} 0.2887 & 0.4283 \\ 0.4283 & 0.7422 \end{bmatrix}$$

We can now start another iteration by defining a new quadratic programming problem using Equation (3.47) and continue the procedure until the optimum solution is found. Note that the objective function reduced from value of 1.5917 to 1.38874 in one iteration when X changed from X_1 to X_2.

3) Method of Feasible Direction

The methods of feasible directions are based on selecting usable feasible direction and determining the proper step length. The methods that adopt this concept is Zoutendijk's method and Rosen's gradient projection method..

3.1 Zoutendijk's method

Algorithm of Zoutendijk's method is stated as follow:

Step 1:

Start with an initial feasible point X_1 and small numbers ε_1, ε_2 and ε_3 to test the convergence of the method. Evaluate $f(X_1)$ and $g_j(X_1), j = 1,2,\ldots,m$. Set the iteration number as $i = 1$.

Step 2:

If $g_j(X_i) < 0, j = 1,2,\ldots,m$ (i.e. X_i is an interior feasible point), set the current search direction as

$S_i = -\nabla f(X_i)$ (3.57)

Normalise S_i in a suitable manner and go to step 5. If at least one $g_j(X_i) = 0$, go to step 3.

Step 3:

Find a usable feasible direction S by solving the direction-finding problem:

Minimise $-\alpha$ (3.58a)

subject to:

$S^T \nabla g_j(X_i) + \theta_j \alpha \leq 0, j = 1,2,..,p$ (3.58b)

$S^T \nabla f + \alpha \leq 0$ (3.58c)

$-1 \leq s_i \leq 1, i = 1,2,..,n$ (3.58d)

where s_i is the ith component of S, the first p constraints have been assumed to be active at the point X_i (the constraint can always be renumbered to satisfy this requirement), and the values of all θ_j can be taken as unity. Here α can be taken as additional design variable.

Step 4:

If the value of α^* found in step 3 is very nearly equal to zero, that is, if $\alpha^* \leq \varepsilon_1$, terminate the computation by taking $X_{\text{opt}} \approx X_i$. If $\alpha^* > \varepsilon_1$, go to step 5 by taking $S_i = S$.

Step 5:

Find a suitable step length λ_i along the direction S_i and obtain a new point X_{i+1} as

$X_{i+1} = X_i + \lambda_i \mathrm{S_i}$ (3.59)

There are 2 methods of finding the step length λ_i., i.e. to determine an optimal step length (λ_i) that minimise $f(X_i + \lambda S_i)$ such that the new point X_{i+1} lies on feasible region. Another method is to choose the step length (λ_i) by trial and error so that it satisfies the relations

$f(X_i + \lambda_i S_i) \leq f(X_i)$ (3.60)

$g_j(X_i + \lambda_i S_i) \leq 0, j = 1,2,..,m$ (3.61)

Step 6:

Evaluate the objective function $f(X_{i+1})$

Step 7:

Test for the convergence of the method. If

$$\left|\frac{f(X_i) - f(X_{i+i})}{f(X_i)}\right| \leq \varepsilon_2 \text{ and } \|X_i - X_{i+1}\| \leq \varepsilon_3 \quad (3.62)$$

Terminate the iteration by taking $X_{\text{opt}} \approx X_{i+1}$. Otherwise, go to step 8.

Step 8:

Set the new iteration number as $i \approx i + 1$, and repeat from step 2 onward.

Example 3.8

Minimise $f(x_1,x_2) = x_1^2 + x_2^2 - 4x_1 - 4x_2 + 8$

subject to:

$g_1(x_1, x_2) = x_1 + 2\,x_1 - 4 \leq 0$

with the starting point $X_1 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$. Take $\varepsilon_1 = 0.001$, $\varepsilon_2 = 0.001$, and $\varepsilon_2 = 0.01$

Solution:

Step 1:

At $X_1 = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$:

$f(X_1) = 8$ and $g_1(X_1) = -4$

Iteration 1

Step 2:

Since $g_1(X_1) < 0$, we take the search direction as

$$S_1 = -\nabla f(X_1) = -\begin{Bmatrix} \partial f/\partial x_1 \\ \partial f/\partial x_2 \end{Bmatrix}_{X_1} = \begin{Bmatrix} 4 \\ 4 \end{Bmatrix}$$

This can be normalised to obtain $S_1 = \begin{Bmatrix} 1 \\ 1 \end{Bmatrix}$

Step 5:

To find the new point X_2, we have to find a suitable step length along S_1. For this, minimise $f(X_1+\lambda S_1)$ with respect to λ. Here

$f(X_1+\lambda S_1) = f(0+\lambda\,, 0+\lambda) = 2\lambda^2 - 8\lambda + 8$

$\frac{df}{d\lambda} = 0$ at $\lambda = 2$

Thus the new point is given by $X_2 = \begin{Bmatrix} 2 \\ 2 \end{Bmatrix}$ and $g_1(X_2) = 2$. As the constraint is violated, the step size has to be connected.

As $g_1 = g_1|_{\lambda=0} = -4$ and $g_1'' = g_1|_{\lambda=2} = 2$, linear interpolation gives the new step length as

$$\lambda_n = -\frac{g_1'}{g_1'' - g_1'}\lambda = \frac{4}{3}$$

This gives $g_1|_{\lambda=\lambda n} = 0$ and hence $X_2 = \begin{Bmatrix} 4/3 \\ 4/3 \end{Bmatrix}$

Step 6: $f(X_2) = \frac{8}{9}$

Step 7: Here

$$\left|\frac{f(X_1) - f(X_2)}{f(X_1)}\right| = \left|\frac{8 - \frac{8}{9}}{8}\right| = \frac{8}{9} > \varepsilon_2$$

$\|X_1 - X_2\| = [(0-\frac{4}{3})^2 + (0-\frac{4}{3})^2]^{1/2} = 1.887 > \varepsilon_2$

And hence the convergence criteria are not satisfied.

Iteration 2

Step 2: As $g_1=0$ at X_2, we proceed to find a usable feasible direction.

Step 3: The direction-finding problem can stated as

Minimise $f = -\alpha$

subject to :

$t_1 + 2t_2 + \alpha + y_1 = 3$

$-\frac{4}{3}t_1 - \frac{4}{3}t_2 + \alpha + y_2 = -\frac{8}{3}$

$t_1 + y_3 = 2$

$t_2 + y_4 = 2$

$t_1 \geq 0$

$t_2 \geq 0$

$\alpha \geq 0$

where y_1 to y_4 are the nonnegativity slack variable.

Since an initial basic feasible solution is not readily available, we introduce an artificial variable $y_5 \geq 0$ into the second constraint equation. By adding the infeasibility form $w = y_5$, the LP problem can be solved to obtain the solution:

$t_1^* = 2$

$t_2^* = \frac{3}{10}$

$\alpha^* = \frac{4}{10}$

$y_4^* = \frac{17}{10}$

$y_1^* = y_2^* = y_3^* = 0$

$-f_{\min} = -\alpha^* = -\frac{4}{10}$

As $\alpha^* > 0$, the usable feasible direction is given by

$$S = \begin{Bmatrix} s_1 \\ s_2 \end{Bmatrix} = \begin{Bmatrix} t_1^* - 1 \\ t_2^* - 1 \end{Bmatrix} = \begin{Bmatrix} 1.0 \\ -0.7 \end{Bmatrix}$$

Step 4: Since $\alpha^* > \varepsilon_1$, we go to the next step.

Step 5: We have to move along the direction $S_2 = \begin{Bmatrix} 1.0 \\ -0.7 \end{Bmatrix}$ from the point $X_2 = \begin{Bmatrix} 1.333 \\ 1.333 \end{Bmatrix}$. To find the minimising step length, we minimise

$$f(X_2 + \lambda S_2) = f(1.333 + \lambda, 1.333 - 0.7\lambda)$$
$$= 1.49\lambda^2 - 0.4\lambda + 0.889$$

As $df/d\lambda = 2.98 - 0.4 = 0$ at $\lambda = 0.134$, the new point is given by

$$X_3 = X_2 + \lambda S_2 = \begin{Bmatrix} 1.333 \\ 1.333 \end{Bmatrix} + 0.134 \begin{Bmatrix} 1.0 \\ -0.7 \end{Bmatrix} = \begin{Bmatrix} 1.467 \\ 1.239 \end{Bmatrix}$$

At this point, the constraint is satisfied since $g_1(X_3) = -0.055$. Since point X_3 lies in the interior of the feasible domain, we go to step 2.

The procedure is continued until the optimum point $X^* = \begin{Bmatrix} 1.6 \\ 1.2 \end{Bmatrix}$ and $f_{\min} = 0.8$ are obtained.

3.2 Rosen's gradient projection method

Another method of feasible direction is Rosen's Gradient Projection Method. This method does not require the solution of an auxiliary linear optimisation variable (α) to find the usable feasible direction. It uses the projection of the negative of the objective function gradient onto the constraints that are currently active. Although the method has been described by Rosen for general nonlinear programming problem, its effectiveness is confined primarily to problems in which the constraints are all linear.

The algorithm of Rosen's Method is given as follows:

Step 1:

Start with an initial point X_1. The point X_1 has to be feasible, that is,

$g_j(X_1) \leq 0, \quad j=1,2,\ldots,m$ (3.63)

Step 2:

Set the iteration number as $i=1$

Step 3:

If X_i is an interior feasible point [i.e. if $g_j(X_1) < 0$ for $j = 1,2,..m$), set the direction of search as $S_i = -\nabla f(X_1)$, normalise the search direction as

$$S_i = \frac{-\nabla f(X_i)}{\|\nabla f(X_i)\|} \quad (3.64)$$

And go to step 5. However, if $g_j(X_1) = 0$ for $j = j_1, j_2, \ldots, j_p$ go to step 4.

Step 4:

Calculate the projection matrix P_i as

$P_i = I - N_p(N_p^T N_p)^{-1} N_p^T$ (3.65) where

$N_p = [\nabla g_{j1}(X_i)\ \nabla g_{j2}(X_i) \ldots \nabla g_{jp}(X_i)]$ (3.66)

and find the normalised search direction S_i as

$$S_i = \frac{-P_i \nabla f(X_i)}{\|P_i \nabla f(X_i)\|} \tag{3.67}$$

Step 5:

Test whether or not $S_i = 0$. If $S_i \neq 0$, go to step 6. If $S_i = 0$, compute the vector λ at X_i as

$\lambda = -(N_p^T N_p)^{-1} N_p^T \nabla f(X_i)$ (3.68)

If all the components of the vector λ are nonnegative, take $X_{\text{opt}} = X_i$ and stop the iterative procedure. If some of the components of λ are negative, find the component λ_q that has the most negative value and form the new matrix N_p as

$N_p = [\nabla g_{j1}\ \nabla g_{j2} \ldots \nabla g_{jp}]$ (3.69)

And go to step 3.

Step 6:

If $S_i \neq 0$, find the maximum step length λ_M that is permissible without violating any of the constraints as $\lambda_M = \min(\lambda_k)$, $\lambda_k > 0$ and k is any integer among 1 to m other than $j_1, j_2, \ldots, j_p$. Also find the value of $df/d\lambda(\lambda_m)$ is zero or negative, take the step length as $\lambda_i = \lambda_M$. On the other hand, if $df/d\lambda(\lambda_m)$ is positive, find the minimising step length λ_i^* either by interpolation or by any other methods, and take $\lambda_i = \lambda_i^*$.

Step 7:

Find the new approximation to the minimum as

$X_{i+1} = X_i + \lambda_i \mathrm{S_i}$ (3.70)

If $\lambda_i = \lambda_M$ or if $\lambda_M \leq \lambda_i^*$, some new constraints (one or more) become active at X_{i+1} and hence generate the new matrix N_p to include the gradients of all active constraints evaluated at X_{i+1}. Set the new iteration number as $i=i+1$, and go to step 4. If $\lambda_i = \lambda_i^*$ and $\lambda_i^* < \lambda_M$, no new constraint will be active at X_{i+1} and hence the matrix N_p remains unaltered. Set the new value of i as $i=i+1$, and go to step 3.

Example 3.9

Minimise $f(x_1,x_2) = x_1^2 + x_2^2 - 2x_1 - 4x_2$

subject to:

$g_1(x_1,x_2) = x_1 + 4x_2 - 5 \le 0$

$g_2(x_1,x_2) = 2x_1 + 3x_2 - 6 \le 0$

$g_3(x_1,x_2) = -x_1 \le 0$

$g_4(x_1,x_2) = -x_2 \le 0$

starting from the point $X_1 = \begin{Bmatrix} 1.0 \\ 1.0 \end{Bmatrix}$.

Solution:

Iteration i=1

Step 3:

Since $g_j(X_1) = 0$ for $j = 1$, we have $p = 1$ and $j_1 = 1$

Step 4: As $N_1 = [\nabla g_1(X_1)] = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$, the projection matrix is given by

$$P_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 \\ 4 \end{bmatrix} \left[[1 \;\; 4] \begin{bmatrix} 1 \\ 4 \end{bmatrix} \right]^{-1} [1 \;\; 4]$$

$$= \frac{1}{17} \begin{bmatrix} 16 & -4 \\ -4 & 1 \end{bmatrix}$$

The search direction S_1 is given by:

$$S_1 = -\frac{1}{17} \begin{bmatrix} 16 & -4 \\ -4 & 1 \end{bmatrix} \begin{Bmatrix} 0 \\ -2 \end{Bmatrix} = \begin{Bmatrix} -\frac{8}{17} \\ \frac{2}{17} \end{Bmatrix} = \begin{Bmatrix} -0.4707 \\ 0.1177 \end{Bmatrix}$$

As

$$\nabla f(X_i) = \begin{Bmatrix} 2x_1 - 2 \\ 2x_2 - 4 \end{Bmatrix}_{X_1} = \begin{Bmatrix} 0 \\ -2 \end{Bmatrix}$$

The normalised search direction can be obtained as

$$S_1 = \frac{1}{[(-0.4707)^2 + (0.1177)^2]^{1/2}} \begin{Bmatrix} -0.4707 \\ 0.1177 \end{Bmatrix} = \begin{Bmatrix} -0.9701 \\ 0.2425 \end{Bmatrix}$$

Step 5:

Since $S_1 \neq 0$, we go to step 6.

Step 6:

To find the step length λ_M, we set

$$X = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix} = X_1 + \lambda S$$

$$= \begin{Bmatrix} 1.0 - 0.9701\lambda \\ 1.0 + 0.2425\lambda \end{Bmatrix}$$

For $j = 2$:

$g_2(x_1,x_2) = (2.0 - 1.9402\,\lambda) + (3.0+0.7275\lambda) - 6.0 = 0$ at $\lambda = \lambda_2 = -0.8245$

For $j = 3$:

$g_3(x_1,x_2) = -(1.0 - 0.9701\,\lambda) = 0$ at $\lambda = \lambda_3 = -1.03$

For $j = 4$:

$g_4(x_1,x_2) = -(1.0 - 0.2425\,\lambda) = 0$ at $\lambda = \lambda_4 = -4.124$

Therefore,

$\lambda_M = \lambda_3 = 1.03$

Also

$f(X) = f(\lambda) = (1.0 - 0.9701\lambda)^2 + (1.0 + 0.2425\lambda)^2 - 2(1.0 - 0.9701\lambda) - 4(1.0 + 0.2425\lambda)$

$= 0.9998\,\lambda^2 - 0.4850\lambda - 4.0$

$$\frac{df}{d\lambda} = 1.9996\,\lambda - 0.4850$$

$$\frac{df}{d\lambda}(\lambda_M) = 1.9996\,(1.03) - 0.4850 = 1.5746$$

As $df/d\lambda\ (\lambda_M) > 0$, we compute the minimising step length λ_1^* by setting $df/d\lambda = 0$. This gives

$$\lambda_1 = \lambda_1^* = \frac{0.4850}{1.9996} = 0.2425$$

Step 7: We obtain the new point X_2 as

$$X_2 = X_1 + \lambda_1 S_1 = \begin{Bmatrix} 1.0 \\ 1.0 \end{Bmatrix} + 0.2425 \begin{Bmatrix} -0.9701 \\ 0.2425 \end{Bmatrix} = \begin{Bmatrix} 0.7647 \\ 1.0588 \end{Bmatrix}$$

Since $\lambda_1 = \lambda_1{}^*$ and $\lambda_1{}^* < \lambda_M$, no new constraint has become active at X_2 and hence the matrix N_1 remains unaltered.

Iteration $i = 2$

Step 3:

Since $g_1(X_2) = 0$, we set $p = 1, j_1 = 1$ and go to step 4.

Step 4: $N_1 = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$, the projection matrix is given by

$$P_2 = \frac{1}{17}\begin{bmatrix} 16 & -4 \\ -4 & 1 \end{bmatrix}$$

$$\nabla f(X_i) = \begin{Bmatrix} 2x_1 - 2 \\ 2x_2 - 4 \end{Bmatrix}_{x_1} = \begin{Bmatrix} 1.5294 - 2.0 \\ 2.1176 - 4.0 \end{Bmatrix} = \begin{Bmatrix} -0.4706 \\ -1.8824 \end{Bmatrix}$$

The search direction S_2 is given by:

$$S_2 = -P_2 \nabla f(X_2) = -\frac{1}{17}\begin{bmatrix} 16 & -4 \\ -4 & 1 \end{bmatrix}\begin{Bmatrix} 0.4706 \\ 1.8824 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}$$

Step 5:

Since $S_2 = 0$, we compute the vector λ at X_2 as

$$\lambda = -(N_1{}^T N_1)^{-1} N_1{}^T \nabla f(X_2)$$

$$= -\frac{1}{17}[1 \quad 4]\begin{Bmatrix} 0.4706 \\ 1.8824 \end{Bmatrix} = 0.4707 > 0$$

The non negative value of λ indicates that we have reached the optimum point and hence that

$$X_{\text{opt}} = X_2 = \begin{Bmatrix} 0.7647 \\ 1.0588 \end{Bmatrix} \text{ with } f_{\text{opt}} = -4.059$$

(4) Generalised Reduced Gradient Method

This method is an extension of the reduced gradient method that was presented originally for solving problems with linear constraints only.

Consider the non linear programming problem:

$$\text{Minimise } f(X) \tag{3.71}$$

subject to

$h_j(X) \le 0, j = 1,2,\ldots, m$ (3.72)

$l_k(X) = 0, k = 1,2, \ldots, l$ (3.73)

$x_i^{(l)} \le x_i \le x_i^{(u)}, \quad i = 1,2,\ldots, n$ (3.74)

By adding a non negative slack variable to each of the inequality constraints, the problem can be stated as

Minimise $f(X)$ (3.75)

subject to

$h_j(X) + x_{n+j} = 0, \ j = 1,2,\ldots, m$ (3.76)

$l_k(X) = 0, k = 1,2, \ldots, l$ (3.77)

$x_i^{(l)} \le x_i \le x_i^{(u)}, \quad i = 1,2,\ldots, n$ (3.78)

$x_{n+j} \ge 0, \ j = 1,2, \ldots., m$ (3.79)

with $n+m$ variables. The problem can be rewritten in a general form as :

Minimise $f(X)$ (3.80)

subject to

$g_j(X) \le 0, j = 1,2,\ldots, m+l$ (3.81)

$x_i^{(l)} \le x_i \le x_i^{(u)}, \quad i = 1,2,\ldots, n+m$ (3.82)

where the lower and upper bounds on the slack variables, x_i are taken as 0 and a large number (infinity), respectively.

The GRG method is based on the idea of elimination of variables using the equality constraints. Thus, theoretically, one variable can be reduced from set of $n+m$ variables for each of the $m+l$ equality constraints. It is convenient to divide the $n+m$ design variables arbitrarily into two sets as

$$X = \begin{Bmatrix} Y \\ Z \end{Bmatrix} \qquad (3.83)$$

where :

$$Y = \begin{Bmatrix} y_1 \\ y_2 \\ \dots \\ y_{n-l} \end{Bmatrix} = \text{design or independent variables} \tag{3.84}$$

$$Z = \begin{Bmatrix} z_1 \\ z_2 \\ \dots \\ z_{m+l} \end{Bmatrix} = \text{state or dependent variables} \tag{3.85}$$

and where the design variables are completely independent and the state variables are dependent on the design variables used to satisfy the constraints $g_j(X) \le 0, j = 1,2,\dots, m+l$.

Consider the first variations of the objective and constraint functions:

$$\mathrm{d}f(X) = \sum_{i=1}^{n-l} \frac{df}{dy_i} dy_i + \sum_{i=1}^{m+l} \frac{df}{dz_i} dz_i = \nabla_Y^T f dY + \nabla_Z^T f dZ \tag{3.86}$$

or

$$\mathrm{d}g = [C]\,\mathrm{d}Y + [D]\,\mathrm{d}Z \tag{3.87}$$

where

$$\nabla_Y f = \begin{Bmatrix} \frac{\partial f}{\partial y_1} \\ \frac{\partial f}{\partial y_2} \\ \dots\dots \\ \frac{\partial f}{\partial y_{n-l}} \end{Bmatrix} \tag{3.88}$$

$$\nabla_Z f = \begin{Bmatrix} \frac{\partial f}{\partial z_1} \\ \frac{\partial f}{\partial z_2} \\ \dots\dots \\ \frac{\partial f}{\partial z_{m+l}} \end{Bmatrix} \tag{3.89}$$

$$[C] = \begin{bmatrix} \frac{\partial g_1}{\partial y_1} & \dots\dots & \frac{\partial g_1}{\partial y_{n-1}} \\ \dots\dots & & \dots\dots \\ \frac{\partial g_{m+1}}{\partial y_1} & \dots\dots & \frac{\partial g_{m+1}}{\partial y_{n-1}} \end{bmatrix} \tag{3.90}$$

$$[D] = \begin{bmatrix} \frac{\partial g_1}{\partial z_1} & \cdots & \frac{\partial g_1}{\partial z_{m+1}} \\ \cdots & & \cdots \\ \frac{\partial g_{m+1}}{\partial z_1} & \cdots & \frac{\partial g_{m+1}}{\partial z_{m+1}} \end{bmatrix} \tag{3.91}$$

$$\mathrm{d}Y = \begin{Bmatrix} dy_1 \\ dy_2 \\ \cdots \\ dy_{n-l} \end{Bmatrix} \tag{3.92}$$

$$\mathrm{d}Z = \begin{Bmatrix} dz_1 \\ dz_2 \\ \cdots \\ dz_{m+l} \end{Bmatrix} \tag{3.93}$$

Assuming that the constraints are originally satisfied at the vector X, ($g(X) = 0$), any change in the vector dX must correspond to dg = 0 to maintain feasibility at X+dX. Equation (3.87) can be solved to express dZ as

$$\mathrm{d}Z = -[D]^{-1}\,[C]\,\mathrm{d}Y \tag{3.94}$$

The change in the objective function due to the change in X is given by Equation (3.86), which can be expressed, using Equation (3.94) as

$$\mathrm{d}f(X) = (\nabla_Y^T f - \nabla_Z^T f\,[D]^{-1}[C])\,\mathrm{d}Y \,. \tag{3.95}$$

or

$$\frac{df}{dY}(X) = G_R \tag{3.96}$$

where

$$G_R = \nabla_Y f - ([D]^{-1}[C])^T \nabla_Z f \tag{3.97}$$

is called the generalised reduced gradient.

Noting that Equation (3.94) is based on using a linear approximation to the original nonlinear problem, we find that the constraints may not be exactly equal to zero at λ, that is d$g \neq 0$. Hence when Y is held fixed, in order to have

$$g_i(X) + dg_i(X) = 0,\ i = 1,2,\ldots, m+l \tag{3.98}$$

we must have

$$g(X) + \mathrm{d}g(X) = 0 \tag{3.99}$$

Using Equation (3.87) for dg in Equation (3.99), we obtain

$dZ = [D]^{-1}\,(-g(X) - [C]\,dY)$ (3.100)

The value of dZ given by Equation (3.100) is used to update the value of Z as

$Z_{\text{update}} = Z_{\text{current}} + dZ$ (3.101)

The constraints evaluated at the updated vector X, and the procedure of finding dZ is repeated until dZ is sufficiently small. Note that Equation (3.100) can be considered as Newton's method of solving simultaneous equations for dZ.

The algorithm can be summarised as follow:

1. Specify the design and state variables. Start with an initial trial vector X. Identify the design and state variables (Y and Z) for the problem using the following guidelines.
   (b) The state variables are to be selected to avoid singularity of the matrix, $[D]$.
   (c) Since the state variables are adjusted during the iterative process to maintain feasibility, any component of X that is equal to its lower or upper bound initially is to be designated a design variable.
   (d) Since the slack variable appear as linear terms in the (originally inequality) constraints, they should be designated as state variables. However, if the initial value of any state variable is zero (its lower bound value), it should be designated a design variable.
2. Compute the generalised reduced gradient. The GRG is determined using Equation (3.97). The derivatives involved in Equation (3.97) can be evaluated numerically, if necessary
3. Test for convergence. If all the components of the GRG are close to zero, the method can be considered to have converged and the current vector X can be taken as the optimum solution of the problem. For this, the following test can be used:

   $\|G_R\| \leq \varepsilon$ (3.102)

   where ε is a small number. If this relation is not satisfied, we go to step 4.
4. Determine the search direction. The techniques such as steepest descent, Fletcher-Reeves, etc that is used to find suitable search direction by using gradient of an

unconstrained objective function can be used for this purpose. For example, if step descent method is used, the vector S is determined as:

$$S = -G_R \tag{3.103}$$

5. Find the minimum along the search direction by using following procedures:

   (a) Find an estimate for λ as the distance to the nearest side constraint. When design variables are considered, we have

$$\lambda = \begin{cases} \dfrac{y_i^{(u)} - (y_i)_{old}}{s_i} & \text{If } t_i > 0 \\ \dfrac{y_i^{(l)} - (y_i)_{old}}{s_i} & \text{If } t_i < 0 \end{cases} \tag{3.104}$$

where s_i is the ith component of S. Similarly, when state variables are considered, we have from Equation (3.94),

$$dZ = -[D]^{-1}[C]\, dY \tag{3.105}$$

using $dY = \lambda S$, Equation (3.104) gives the search direction for the variables Z as

$$T = -[D]^{-1}\, [C]\, S \tag{3.106}$$

Thus

$$\lambda = \begin{cases} \dfrac{z_i^{(u)} - (z_i)_{old}}{t_i} & \text{If } t_i > 0 \\ \dfrac{z_i^{(l)} - (z_i)_{old}}{t_i} & \text{If } t_i < 0 \end{cases} \tag{3.107}$$

where t_i is the ith component of T.

   (b) The minimum value of λ given by Equation (3.104), λ_1, makes some design variable attain its lower or upper bound. Similarly, the minimum value of λ given by Equation (3.107), λ_2 will make some state variable attain its lower or upper bound. The smaller of λ_1 or λ_2 can be used as an upper bound on the value of λ for initializing a suitable one-dimensional minimisation procedure. The quadratic interpolation method can be used conveniently for finding the optimal step length λ*.

   (c) Find the new vector X_{new}:

$$X_{\text{new}} = \begin{Bmatrix} Y_{old} + dY \\ Z_{old} + dZ \end{Bmatrix} = \begin{Bmatrix} Y_{old} + \lambda^* S \\ Z_{old} + \lambda^* T \end{Bmatrix} \tag{3.108}$$

If the vector X_{new} corresponding to λ^* is found infeasible, then Y_{new} is held constant and Z_{new} is modified using Equation (3.100) with $dZ = Z_{\text{new}} - Z_{\text{old}}$. Finally, when convergence is achieved with Equation (3.100), we find that

$$X_{\text{new}} = \begin{Bmatrix} Y_{old} + \Delta Y \\ Z_{old} + \Delta Z \end{Bmatrix} \tag{3.109}$$

and go to step 1.

GRG algorithm has been worldwide adopted in many optimisation software. One of them is GRG2. It uses a robust implementation of the BFGS quasi-Newton algorithm as its default choice for determining a search direction. A limited-memory conjugate gradient method is also available. The problem Jacobian is stored and manipulated as a dense matrix. The GRG2 software may be used as a stand-alone system or called as a subroutine. The user is not required to supply code for first partial derivatives of problem functions; forward or central difference approximations may be used instead. This software is also incorporated in MS-Excel Solver for solving non linear programming problems.

Example 3.10

Minimise $f(x_1, x_2, x_3) = (x_1 - x_2)^2 + (x_2 - x_3)^4$

subject to:

$g_1(X) = x_1(1+x_2^2) + x_3^4 - 3 = 0$

$-3 \leq x_1 \leq 3, \ i = 1,2,3$

using the GRG method.

Solution:

Step 1. We choose arbitrarily the independent and dependent variables as

$$Y = \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = \begin{Bmatrix} x_1 \\ x_2 \end{Bmatrix}, Z = \{z_1\} = \{x_3\}$$

Let the starting vector be $X_1 = \begin{Bmatrix} -2.6 \\ 2 \\ 2 \end{Bmatrix}$ with $f(X_1) = 21.6$.

Step 2: Compute the GRG at X_1. Noting that

$$\frac{\partial f}{\partial x_1} = 2(x_1 - x_2)$$

$$\frac{\partial f}{\partial x_2} = -2(x_1 - x_2) + 4(x_2 - x_3)^3$$

$$\frac{\partial f}{\partial x_3} = -4(x_2 - x_3)^3$$

$$\frac{\partial g_1}{\partial x_1} = 1 + x_2^2$$

$$\frac{\partial g_1}{\partial x_2} = 2\, x_1\, x_2$$

$$\frac{\partial g_1}{\partial x_3} = 4\, x_3^3$$

We find, at X_1,

$$\nabla_Y f = \left\{\begin{matrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{matrix}\right\}_{X_1} = \left\{\begin{matrix} 2(-2.6-2) \\ -2(-2.6-2)+4(2-2)^3 \end{matrix}\right\} = \left\{\begin{matrix} -9.2 \\ 9.2 \end{matrix}\right\}$$

$$\nabla_Z f = \left\{\frac{\partial f}{\partial x_3}\right\}_{X_1} = \{-4(x_2 - x_3)^3\}\, x_1 = 0$$

$$[C] = \left[\frac{\partial g_1}{\partial x_1} \quad \frac{\partial g_1}{\partial x_2}\right]_{X_1} = [5 \quad -10.4]$$

$$[D] = \left[\frac{\partial g_1}{\partial x_3}\right]_{X_1} = [32]$$

$[D]^{-1} = [\tfrac{1}{32}]$, $\quad [D]^{-1}[C] = \tfrac{1}{32}[5 \quad -10.4] = [0.15625 \quad -0.325]$

$$G_R = \nabla_Y f - ([D]^{-1}[C])^T \nabla_Z f$$

$$= \left\{\begin{matrix} -9.2 \\ 9.2 \end{matrix}\right\} - \left\{\begin{matrix} 0.15625 \\ -0.325 \end{matrix}\right\}(0) = \left\{\begin{matrix} -9.2 \\ 9.2 \end{matrix}\right\}$$

Step 3: Since the components of G_R are not zero, the point X_1 is not optimum, and hence we go to step 4.

Step 4: We use the steepest descent method and take the search direction as

$$S = -\, G_R = \begin{Bmatrix} -9.2 \\ 9.2 \end{Bmatrix}$$

Step 5: We find the optimal step length along S.

(a) Considering the design variables, we use Equation (3.104), to obtain:

For $y_1 = x_1$:

$$\lambda = \frac{3-(-2.6)}{9.2} = 0.6087$$

For $y_2 = x_2$:

$$\lambda = \frac{-3-(-2)}{-9.2} = 0.5435$$

Thus the smaller value gives λ_1 =0.5435. Equation (3.106) gives:

$$T = -[D]^{-1}\,[C]\,S = -(0.15625 \;\; -0.325) \begin{Bmatrix} 9.2 \\ -9.2 \end{Bmatrix} = -4.4275$$

And hence Equation (3.107), leads to

For $z_1 = x_3$; $\lambda = \dfrac{3-(2)}{-4.4275} = 1.1293$

Thus $\lambda_2 = 1.1293$.

(b) The upper bound on λ is given by the smaller of λ_1 and λ_2, which is equal to 0.5435. By expressing

$$X = \begin{Bmatrix} Y + \lambda S \\ Z + \lambda T \end{Bmatrix}$$

We obtain

$$X = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} -2.6 \\ 2 \\ 2 \end{Bmatrix} + \lambda \begin{Bmatrix} 9.2 \\ -9.2 \\ -4.4275 \end{Bmatrix} = \begin{Bmatrix} -2.6 + 9.2\lambda \\ 2 - 9.2\lambda \\ 2 - 4.4275\lambda \end{Bmatrix}$$

And hence

$$f(\lambda) = f(X) = (-2.6 + 9.2\lambda - 2 + 9.2\,\lambda)^2 + (2-9.2\lambda-2+4.4275)^4$$

$$= 518.7806\,\lambda^4 + 338.56\lambda^2 - 169.28\,\lambda + 21.16$$

$df/d\lambda = 0$ gives

$$2075.1225\,\lambda^3 + 677.12\,\lambda - 169.28 = 0$$

From which we find the root as $\lambda^* \approx 0.22$. Since λ^* is less than the upper bound value 0.5435, we use λ^*.

(c) The new vector X_{new} is given by

$$X_{\text{new}} = \begin{Bmatrix} Y_{old} + dY \\ Z_{old} + dZ \end{Bmatrix} = \begin{Bmatrix} Y_{old} + \lambda^* S \\ Z_{old} + \lambda^* T \end{Bmatrix}$$

$$= \begin{Bmatrix} -2.6 + 0.22 * 9.2 \\ 2 + 0.22 * (-9.2) \\ 2 + 0.22 * (-4.4275) \end{Bmatrix} = \begin{Bmatrix} -0.576 \\ -0.024 \\ 1.02595 \end{Bmatrix}$$

with

$$dY = \begin{Bmatrix} 2.024 \\ -2.024 \end{Bmatrix}, \qquad dZ = \{-0.97405\}$$

Now, we need to check whether this vector is feasible. Since

$g_1(X_{\text{new}}) = (-0.576)\,[1 + (-0.024)^2] + (1.02595)^4 - 3 = -2.4685 \neq 0$.

The vector X_{new} is infeasible. Hence we hold Y_{new} constant and modify Z_{new} using Newton's method Equation (3.100) as

$$dZ = [D]^{-1}\,(-g(X) - [C]\,dY)$$

Since

$$[D] = \left[\frac{\partial g_1}{\partial z_1}\right] = [4x_3^3] = [4\,(1.02595)^3] = [4.319551]$$

$g_1(X) = \{-2.4684\}$

$$[C] = \left[\frac{\partial g_1}{\partial y_1} \quad \frac{\partial g_1}{\partial y_2}\right] = \{[2(-0.576 + 0.024)][-2(-0.576+0.024)+4(-0.024 - 1.02595)^3]\}$$

$$= [-1.104 \quad -3.5258]$$

$$dZ = \frac{1}{4.319551}\left[2.4684 - \{-1.104 - 3.5258\}\begin{Bmatrix} 2.024 \\ -2.024 \end{Bmatrix}\right]$$

$= \{-0.5633\}$

We have $Z_{new} = Z_{old} + dZ = \{2\text{-}0.5633\} = \{1.4367\}$. The current X_{new} becomes

$$X_{new} = \begin{Bmatrix} Y_{old} + dY \\ Z_{old} + dZ \end{Bmatrix} = \begin{Bmatrix} -0.576 \\ -0.024 \\ 1.4367 \end{Bmatrix}$$

The constraint becomes

$g_1 = (-0.576)(1\text{-}(-0.024)^2) + (1.4367)^4 - 3 = 0.6842 \neq 0$

Since this X_{new} is infeasible, we need to apply Newton's method Equation (3.100) at the current X_{new}. In the present case, instead of repeating Newton's iteration, we can find the value of $Z_{new} = \{x_3\}_{new}$ by satisfying the constraint as

$g_1(X) = (-0.576)[1\text{-}(-0.024)^2) + x_3{}^4 - 3 = 0$

or $x_3 = (2.4237)^{0.25} = 1.2477$

This gives

$$X_{new} = \begin{Bmatrix} -0.576 \\ -0.024 \\ 1.2477 \end{Bmatrix} \text{ and}$$

$f(X_{new}) = (-0.576 + 0.024)^2 + (-0.024 - 1.2477)^4 = 2.9201$

Next we go to step 1.

Step 1: We do not have to change the set of independent and dependent variables and hence we go to the next step.

Step 2: We compute the GRG at the current X using Equation (3.97). Since

$$\nabla_Y f = \begin{Bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{Bmatrix} = \begin{Bmatrix} 2(-0.576+0.024) \\ -2(-0.576+0.024)+4(-0.024-1.2477)^3 \end{Bmatrix} = \begin{Bmatrix} -1.104 \\ -7.1225 \end{Bmatrix}$$

$$\nabla_Z f = \begin{Bmatrix} \frac{\partial f}{\partial z_1} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial f}{\partial x_3} \end{Bmatrix} = \{-4(-0.024 - 1.2477)^3\} = \{8.2265\}$$

$$[C] = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} \end{bmatrix} = [(-1 + (-0.024)^2 \quad 2(-0.576)(-0.024)]$$

$$= [1.000576 \quad 0.027648]$$

$$[D] = \left[\frac{\partial g_1}{\partial x_3}\right] = [4x_3^3] = [4\,(1.2477)^3] = [7.7694]$$

$$[D]^{-1}\,[C] = \tfrac{1}{7.7694}[1.000576 \;\; 0.027648] = [0.128784 \;\; 0.003558]$$

$$G_R = \nabla_Y f - ([D]^{-1}[C])^T \nabla_Z f$$

$$= \begin{Bmatrix} -1.104 \\ -7.1225 \end{Bmatrix} - \begin{Bmatrix} 0.128784 \\ 0.003558 \end{Bmatrix}(8.2265) = \begin{Bmatrix} -2.1634 \\ -7.1518 \end{Bmatrix}$$

Since $G_R \neq 0$, we need to proceed to the next step.

Note: It can be seen that the value of the objective function reduced significantly from 21.16 to 2.9201 in one iteration.

3.4.2.2 Indirect Method

In Indirect Methods, there are two basic solving methods, that is Transformation of variables methods and Penalty function methods.

(1) Transformation of variables method

If the constraints are explicit function of the variables and have certain simple forms, it may be possible to use transformation techniques so that the constraints would be automatically satisfied. There are two options to transform the variables, i.e. change of variables and Eliminations of variables.

1.a. Change of variables method

Change of variables method, particularly useful when the variables are bounded by lower and upper limits, can be used to convert a constrained optimisation problem into unconstrained one. The method should be used only when it is possible to eliminate all constraints. Partial transformation may result into a distorted objective function that may be more difficult to minimise than the original function. Some typical transformations are indicated below:

1. If lower and upper bounds on x_i are specified as

$$l_i \leq x_i \leq u_i \qquad (3.110)$$

these can be satisfied by transforming the variable x_i as

$$x_i = l_i + (u_i - l_i)\sin^2 y_i \qquad (3.111)$$

where y_i is the new variable, which can take any value.

2. If a variable x_i is restricted to lie in the interval (0,1), such transformation can be used, i.e.

$$x_i = \sin^2 y_i \,,\ x_i = \cos^2 y_i \tag{3.112}$$

$$x_i = \frac{e^{y_i}}{e^{y_i} + e^{-y_i}} \text{ or } x_i = \frac{y_i^{\,2}}{1 + y_i^{\,2}}$$

3. If the variable x_i is constrained to take only positive values, the transformation can be

$$x_i = \text{abs}(y_i),\ x_i = y_i^2 \text{ or } x_i = e^{y_i} \tag{3.113}$$

4. If the variable is restricted

$$x_i = \sin y_i,\ x_i = \cos y_i, \text{ or } x_i = \frac{2y_i}{1+y_i^2} \tag{3.114}$$

Note the following aspects that are important in transformation techniques

1. The constraints $g_j(X)$ have to be very simple function of x_i.
2. For certain constraint it may not be possible to find the necessary transformation
3. If it is not possible to eliminate all the constraints by making change of variables, it may be better not to use the transformation at all. The partial transformation may sometimes produce a distorted objective function which might be more difficult to minimise than the original function.

Example 3.11

Find the dimensions of a rectangular prism type box that has the largest volume when the sum of its length, width and height is limited to a maximum value of 60 in. and its length is restricted to a maximum value of 36 in.

Solution:

Let x_1, x_2, and x_3 denote the length, width, and height of the box, respectively. The problem can be stated as follows:

$$\text{Maximise } f(x_1, x_2, x_3) = x_1x_2x_3 \tag{E$_1$}$$

subject to:

$x_1 + x_2 + x_3 \leq 60$ (E$_2$)

$x_1 \leq 36$ (E$_3$)

$x_i \geq 0,\ i = 1,2,3$ (E$_4$)

By introducing new variable as:

$$y_1 = x_1,\ y_2 = x_2,\ y_3 = x_1 + x_2 + x_3 \quad (E_5)$$

or

$$x_1 = y_1,\ x_2 = y_2,\ x_3 = y_3 - y_1 - y_2 \quad (E_6)$$

the constraints of Equation (E$_2$) to (E$_4$) can be restated as

$$0 \leq y_1 \leq 36, \qquad 0 \leq y_2 \leq 60, \qquad 0 \leq y_3 \leq 60 \quad (E_7)$$

where the upper bound, for example, on y_2 is obtained by setting $x_1 = x_3 = 0$ in Equation (E$_2$). The constraints of Equation (E$_7$) will be satisfied automatically if we define new variables z_i, $i = 1,2,3$ as

$$y_1 = 36\sin^2 z_1,\ y_2 = 60\sin^2 z_2,\ y_3 = 60\sin^2 z_3 \quad (E_8)$$

Thus the problem can be stated as an unconstrained problem as follows:

Maximise $f(z_1, z_2, z_3)$

$= y_1 y_2 (y_3 - y_1 - y_2)$

$= 2160 \sin^2 z_1 \sin^2 z_2 (60 \sin^2 z_3 - 36 \sin^2 z_1 - 60 \sin^2 z_2)$ (E$_9$)

The necessary conditions of optimality yield the relations

$$\frac{\partial f}{\partial z_1} = 259{,}000 \sin z_1 \cos z_1 \sin^2 z_2 (\sin^2 z_3 - \tfrac{6}{5} \sin^2 z_1 - \sin^2 z_2) = 0 \quad (E_{10})$$

$$\frac{\partial f}{\partial z_2} = 518{,}400 \sin^2 z_1 \sin z_2 \cos z_2 (\tfrac{1}{2} \sin^2 z_3 - \tfrac{3}{10} \sin^2 z_1 - \sin^2 z_2) = 0 \quad (E_{11})$$

$$\frac{\partial f}{\partial z_3} = 259{,}200 \sin^2 z_1 \sin^2 z_2 \sin z_3 \cos z_3 = 0 \quad (E_{12})$$

Equation (E$_{12}$) gives the nontrivial solution as $\cos z_3 = 0$ or $\sin^2 z_3 = 1$. Hence Equation (E$_{10}$) and (E$_{11}$) yield $\sin^2 z_1 = \frac{5}{9}$ and $\sin^2 z_2 = \frac{1}{3}$. Thus the optimum solution is given by $x_1^* = 20$ in., $x_2^* = 20$ in. and $x_3^* = 20$ in., and the maximum volume = 8000 in.^{3}.

1.b. Elimination of variables method

Elimination of variables method is used for an n-variable problem with m inequality constraints, if it is known in advance that r constraints would be active at the optimal point. It may be possible to eliminate any r variables and obtain a new problem involving n-r variables with m-r constraints. This new problem with reduced number of variables and constraints may be easier to solve. However, it may not be possible to know before hand, which of the constraints would be active at the optimum point.

(2) Penalty Function Method

In penalty function methods, also known as Sequential Unconstrained Minimisation Techniques (SUMTs), the constrained minimisation problem is transformed into alternative formulations such that the minimisation problem is solved through a sequence of unconstrained minimisation problems. The alternative formulation is obtained by adding a penalty term that takes care of the constraints.

There two methods of this concept, i.e. exterior penalty function method and interior penalty function method. In exterior penalty function methods, all intermediate solutions lie in the infeasible region and converge to the optimal solution from exterior of the feasible region. Herein, it is not necessary to have a starting feasible solution, however, since intermediate solutions are infeasible, search cannot be stopped before reaching the optimum. In interior penalty function, all intermediate solutions lie in the feasible region and converge to the optimal solution from interior of the feasible region. Herein, the search can be stopped any time and the solution though sub optimal is feasible, therefore it can be taken as the final solution. However, an initial feasible solution is necessary to start the search procedure.

2.1 Interior Penalty Function Method

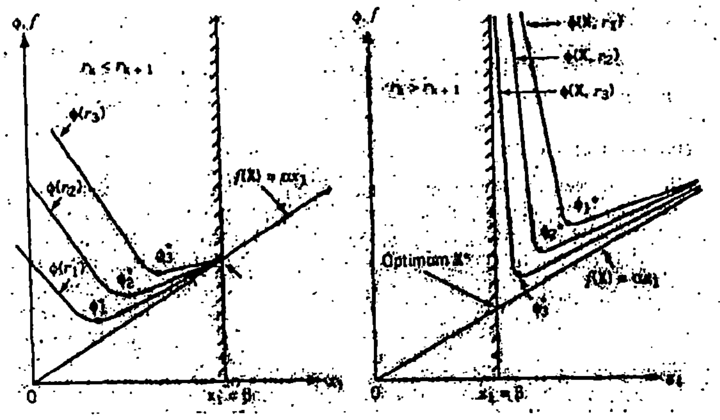


Fig. 3-4 Penalty function methods: (a) exterior method; (b) interior method

In interior penalty function methods, a new function (ϕ function) is constructed by augmenting a penalty term to the objective function. The penalty term is chosen such that its value will be small at points away from the constraint boundaries and will tend to infinity as the constraint boundaries approached. Hence the value of the ϕ function also 'blows up' as the constraints boundaries approach. This behaviour can be seen from Figure 3-4. Thus once the unconstrained minimisation of ϕ (X, r_k) is started from any feasible point X_1, the subsequent points generated will always lie within the feasible domain since the constraint boundaries act as barriers during the minimisation process. This is why the interior penalty function methods are also known as barrier methods. The ϕ function defined originally by C.W. Carroll in 1961 is:

$$\phi(X, r_k) = f(X) - \phi r_k \sum_{j=1}^{m} \frac{1}{g_j(X)} \qquad (3.115)$$

It can be seen that the value of the function ϕ will always be greater than f since g_j (X) is negative for all feasible point X. If any constraint g_j (X) is satisfied critically (with equality sign), the value of ϕ tends to infinity. It is to be noted that the penalty term in Equation (3.115) is not defined if X is infeasible. This introduces serious shortcoming while using the Equation (3.115). Since this equation does not allow any constraint to be violated, it required a feasible starting point for search toward the optimum point.

The algorithm is given as follows:

1. Start with an initial feasible point X_1 satisfying all the constraints with strict inequality sign, that is, $g_j(X_1) < 0$ for $j = 1,2, \ldots, m$, and an initial value of $r_1 > 0$. Set $k = 1$.
2. Minimise $\phi(X, r_k)$ by using any of the unconstrained minimisation method and obtain the solution X_k^*.
3. Test whether X_k^* is the optimum solution of the original problem. If X_k^* is found to be optimum, terminate the process. Otherwise, go to the next step.
4. Find the value of the next penalty parameter, r_{k+1}, as

   $r_{k+1} = c\, r_k$

   where $c < 1$.
5. Set the new value of $k = k+1$, take the new starting point as $X_1 = X_k^*$, and go to step 2.

Example 3.12

Minimise $f(X) = x_1^3 - 6x_1^2 + 11x_1 + x_3$

subject to :

$x_1^2 + x_2^2 - x_3^2 \le 0$

$4 - x_1^2 - x_2^2 - x_3^2 \le 0$

$x_3 - 5 \le 0$

$-x_i \le 0,\ i = 1,2,3$

Solution:

The interior penalty function method, coupled with the Davidon – Fletcher- Powell method of unconstrained minimisation and cubic interpolation method of one-dimensional search, is used to solve this problem. The necessary data are assumed as follows:

$$\text{Starting feasible point, } X_1 = \begin{Bmatrix} 0.1 \\ 0.1 \\ 3.0 \end{Bmatrix}$$

$r_1 = 1.0,\quad f(X_1) = 4.041,\quad \phi(X_1, r_1) = 25.1849$

The optimum solution of this problem is known to be $X^* = \begin{Bmatrix} 0 \\ \sqrt{2} \\ \sqrt{2} \end{Bmatrix}, f^* = \sqrt{2}$. The results of numerical optimisation are summarised in Table 3-5.

Table 3-5 Results for Example 3.12

k	Value of r_k	x_1^*	x_2^*	x_3^*	ϕ_k^*	f_k^*
1	1.0×10^{0}	0.37898	1.67965	2.34617	10.36219	5.70766
2	1.0×10^{-1}	0.10088	1.41945	1.68302	4.12440	2.73267
3	1.0×10^{-2}	0.03066	1.41411	1.49842	2.25437	1.83012
4	1.0×10^{-3}	0.009576	1.41419	1.44081	1.67805	1.54560
5	1.0×10^{-4}	0.003020	1.41421	1.422263	1.49745	1.45579
6	1.0×10^{-5}	0.0009530	1.41421	1.41687	1.44052	1.42735
7	1.0×10^{-6}	0.0003013	1.41421	1.41505	1.42253	1.41837
8	1.0×10^{-7}	0.00009535	1.41421	1.41448	1.41684	1.41553
9	1.0×10^{-8}	0.00003019	1.41421	1.41430	1.41505	1.41463
10	1.0×10^{-9}	0.000009567	1.41421	1.41424	1.41448	1.41435
11	1.0×10^{-10}	0.00003011	1.41421	1.41422	1.41430	1.41426
12	1.0×10^{-11}	$0.9562 x 10^{-6}$	1.41421	1.41422	1.41424	1.41423
13	1.0×10^{-12}	$0.3248 x 10^{-6}$	1.41421	1.41421	1.41422	1.41422

2.2 Exterior Penalty Function Method

In the exterior penalty function method, the ϕ function is generally taken as

$$\phi(X, r_k) = f(X) + r_k \sum_{j=1}^{m} \langle g_j(X) \rangle^q \tag{3.116}$$

where r_k is a positive penalty parameter, the exponent q is a nonnegative constant, and the bracket function $\langle g_j(X) \rangle$ is defined as

$$\langle g_j(X) \rangle = \max \langle g_j(X), 0 \rangle \tag{3.117}$$

$$= \begin{cases} g_j(X) & g_j(X) > 0 \text{ (constraints is violated)} \\ 0 & g_j(X) \leq 0 \text{ (constraint is satisfied)} \end{cases}$$

It can be seen from Equation (3.116) that the effect of the second term on the right side is to increase ϕ (X, r_k) in proportion to the qth power of the amount by which the constraints are violated. Thus, there will be a penalty for violating the constraints, and the amount of penalty will increase at a faster rate than will the amount of violation of a constraint (for $q > 1$). This is the reason why the formulation is called the penalty function method. Usually, the function ϕ (X, r_k) possesses a minimum as a function of X in the infeasible region. The unconstrained minima X_k^* converge to the optimal solution of the original problem as k tends to infinity and r_k also tends to infinity. Thus, the unconstrained minima approach the feasible domain gradually, and as k tends to infinity, the X_k^* eventually lies in the feasible region.

The algorithm is given as follows:

1. Start from any design X_1 and a suitable value of r_1. Set $k = 1$.
2. Find the vector X_k^* that minimises the function

$$\phi\,(X, r_k) = f(X) + r_k \sum_{j=1}^{m} \langle g_j(X) \rangle^q \tag{3.118}$$

3. Test whether the point X_k^* satisfies all the constraints. If X_k^* is feasible, it is the desired optimum and hence terminate the procedure. Otherwise, go to step 4.
4. Choose the next value of the penalty parameter that satisfies the relation

$$r_{k+1} > r_k \tag{3.119}$$

and set the new value of k as original k plus 1 and go to step 2. Usually, the value of r_{k+1} is chosen according to the relation $r_{k+1} = c\, r_k$, where c is a constant greater than 1.

Example 3.13

Minimise $f(x_1, x_2) = \frac{1}{3}(x_1+1)^3 + x_2$

subject to:

$g_1(x_1, x_2) = 1 - x_1 \leq 0$

$g_2(x_1, x_2) = -x_2 \leq 0$

Solution:

To illustrate the exterior penalty function method, we solve the unconstrained minimisation problem by using differential calculus method. As such, it is not necessary to have an initial trial point X_1. The ϕ function is:

$\phi\,(X_1, r) = \frac{1}{3}(x_1+1)^3 + x_2 + r\,[\max(0, 1-x_1)]^2 + r\,[\max(0,-x_2)]^2$

The necessary conditions for the unconstrained minimum of $\phi\,(X_1, r)$ are

Dr. A.A. Kazmi
C.E.D.

(Near Prof SS Jain's room in Transportation section)

Tentative date of Examination
30th June, 2005 at 3.00 Pm
in Seminar room of WRD&M.

[illegible]
20/6/05
(M.L. Kansal)

Roy Pardede
Mobile No. 09897314777

$$\frac{\partial\phi}{\partial x_1} = (x_1+1)^2 - 2r\,[\max(0, 1-x_1)] = 0$$

$$\frac{\partial\phi}{\partial x_2} = 1 - 2r\,[\max(0,-x_2)] = 0$$

These equations can be written as

$\min\,[(x_1+1)^2, (x_1+1)^2 - 2r(1-x_1)] = 0$ (E₁)

$\min\,[1, 1+2rx_2] = 0$ (E₂)

In Equation (E₁) if $(x_1+1)^2 = 0, x_1 = -1$ (this violates the first constraint), and if

$(x_1+1)^2 - 2r(1-x_1) = 0, \quad x_1 = -1 - r + \sqrt{r^2+4r}$

In Equation (E₂) the only possibility is that $1 + 2rx_2 = 0$ and hence $x_2 = -1/2r$.

Thus the solution of the unconstrained minimisation problem is given by

$$x_1^*(r) = -1 - r + r\left(1+\frac{4}{r}\right)^{\frac{1}{2}} \qquad (E_3)$$

$$x_2^*(r) = -\frac{1}{2r} \qquad (E_4)$$

From this, the solution of the original constrained problem can be obtained as

$x_1^* = \lim_{r\to\infty} x_1^*(r) = 1, \quad x_2^* = \lim_{r\to\infty} x_2^*(r) = 0$

$f_{\min} = \lim_{r\to\infty} \phi_{\min}(r) = \frac{8}{3}$

The convergence of the method, as r increases gradually, can be seen from Table 3-6.

Table 3-6 Results for Example 3.13

Value of r	x_1^*	x_2^*	$\phi_{\min}(r)$	$f_{\min}(r)$
0.001	-0.93775	-500.00000	-249.9962	-500.0000
0.01	-0.80975	-50.00000	-24.9650	-49.9977
0.1	-0.45969	-5.00000	-2.2344	-4.9474
1	0.23607	-0.50000	0.9631	0.1295
10	0.83216	-0.05000	2.3068	2.0001
100	0.98039	-0.00500	2.6249	2.5840
1000	0.99800	-0.00050	2.6624	2.6582
10,000	0.99963	-0.00005	2.6655	2.6652
∞	1	0	$\frac{8}{3}$	$\frac{8}{3}$

3.5 STOCHASTIC SEARCH TECHNIQUES

Although the linear and nonlinear methods are good for finding local optima, in real problems it quickly becomes inconvenient to invert matrices (linear programming) or calculate the partial derivatives with respect to the decision variables (nonlinear programming). In such a situation, knowledge of the functional relationship between the objective function value and the decision variables either does not exist or is too complex to be usable. Automated search methods are then used instead of computationally intensive mathematical programming approaches. The feature common to all of these methods is a generate-and-test strategy in which a new point is generated and its function value tested. Depending on the particular method, a new point (or set of points) is generated, and the search for the best solution continues.

3.5.1 Genetic Algorithms

Most real network models are too large or too complex to be handled by any of the previously discussed optimization methods without making significant simplifications. Among the techniques that show promise, *genetic algorithms* (GAs) are most capable of meeting the needs of the design engineers without the necessity of contorting the problem to fit the algorithm (Dandy, Simpson, and Murphy, 1996; Savic and Walters, 1997; Walters, Halhal, Savic, and Ouazar, 1999; Wu et al., 2002).

GAs have a relatively short but promising history, although the basic principles date from the beginning of life on earth. In simple terms, the GA uses a computer model of Darwinian evolution to "evolve" good designs or solutions to highly complex problems for which classical solution techniques such as linear programming or gradient-based methods are often inadequate. The GA incorporates ideas such as a population of solutions to a problem, survival of the fittest (most suitable) solutions within a population, birth, death, breeding, inheritance of genetic material (design parameters) by children from their parents, and occasional mutations of that material (thereby creating new design possibilities).

A genetic Algorithm (GA) is an approach used for optimal design in many fields of engineering including water transmission and distribution networks. It is a search algorithm based on natural selection and the mechanism of population genetics. GA simulates mechanism of population generation and natural rules of survival. It relies on the collective learning process within a population of individuals, each of which represent a point in space of feasible or infeasible solutions.

A GA developed for distribution system optimization uses:

- An objective function defined on a set of decision variables (pipe diameters, for example)
- A calibrated model of the system to simulate its hydraulic behavior and to ensure that continuity and head-loss equations are satisfied at all times (hard constraints)
- A penalty term to penalize insufficient levels of service (soft constraints), such as pressures at nodes, imbalance of reservoir flows, or low/high velocity in pipes.

3.5.1.1 GA Characteristics

Genetic algorithm differs from other search methods in the following ways:

1. GA works with the coding of the parameter set, not with the parameters themselves.

   In other search method, decision variables, such as pipe diameters and nodal HGL values are directly used in the formulation. In GA, however, the decision variables are coded as a finite length string, each string representing a feasible or infeasible solution. Each string consists of sub strings, wherein each sub string represents a parameter, e.g. a pump in on or off condition, a link size, and so on. The coded string is similar to the structure of a chromosome of genetic code. Standard GA uses a binary alphabet (character is 0 or 1) to form a chromosome. Let us assume that in a network optimisation problem we have pipe sizes ranging from 0 (link is absent in a looped network) to 750 mm, a maximum available size, as shown in Table 3-7. Since we have fifteen pipe size possibilities, each sub string, denoting a pipe size, consists of four bits (2^4 = 16>15). Thus, in binary coding as shown in Col. 3, 100 mm is coded by 4 bit string 0010, a size 250 mm by 0110, and so on. A trial solution with combination of pipe size of all links will become a union of binary codes. A network consisting of six links, labelled 1, ..., 6 with pipe sizes of 600, 400, 300, 250, 200, and 150 mm, respectively, is coded by a 24-bit string 110010010111011001010100.

Table 3-7 Example of Pipe Size and coding

Serial Number	Diameter (mm)	Coded sub string	
(1)	(2)	Binary (3)	Gray (4)
1	0	0000	0000
2	80	0001	0001
3	100	0010	0011
4	125	0011	0010
5	150	0100	0110
6	200	0101	0111
7	250	0110	0101
8	300	0111	0100
9	350	1000	1100
10	400	1001	1101
11	450	1010	1111
12	500	1011	1110
13	600	1100	1010
14	700	1101	1011
15	750	1110	1001

One disadvantage with ordinary binary code is that two similar solutions may differ in several bits. For example, in ordinary binary code of Column 3, a pipe of 300 mm diameter is represented by sub string 0111; while the next larger size of 350 mm diameter is represented by substring 1000 in which all four bits of the sub string have changed. To avoid this, Gray coding may be used. In Gray coding adjacent pipe sizes are represented by substrings that differ by only one bit, as shown in Column 4.

2. GA searches from a population of points, not from a single point.

   In GA, a population of strings is generated and tested simultaneously in one iteration and the process is continued successively. This process is similar to a natural biological process wherein successive generations of organisms are born and brought up. Since each string represents a solution, we consider several starting points and climb many peaks simultaneously in a multimodal maximisation problem. Since these solutions are spread through out the solution space, probability of reaching the global optimum solution is increased. Furthermore, a number of optimal (or near optimal) solutions are available in the end; thus, the designer has a wide range of solutions to choose from.

In the usual search methods (e.g. steepest descent method), we consider a single point in space (a particular flow distribution), follow certain rules to select direction of movement and step size and ultimately reach a local optimum solutions (corresponding to a branched configuration). We may obtain several solutions by considering several starting points (several flow distributions or several branching configurations) successively. In GA, however, we consider several starting points right from the beginning, consider them in parallel, and thus obtain several solutions simultaneously.

3. GA requires only the objective function, not trend, derivative or other auxiliary data.

   In direct optimisation methods, the objective function and constraints are considered simultaneously; thus knowledge of optimisation is required. GA, on the other hand, is similar to the traditional approach in which a solution is generated, tested for its feasibility, and the value of the objective function is evaluated. Thus in GA, the objective function and constraints are considered separately.

4. GA uses probabilistic transition rules, not deterministic transition rules.

   Genetic Algorithms uses probabilistic rules rather than deterministic rules in moving from one set of trial solutions to the next set of solutions.

3.5.1.2 GA Operators

In Genetic Algorithms, a set of P initial solutions is generated randomly. The initial solution P, is usually between 30 to 200 for distribution networks. These P initial solutions are represented by P strings, each string consisting of X substring as shown in Fig. 3.5. Here, A, B, .., P represent P strings and subscripts 1,2,,..,x,.., X denote X substrings. Thus, sub string B_2 denotes sub string (link) 2 in string (solution) B and is coded 1011 in ordinary coding if it is of 500 mm diameter (according Table 4.1).

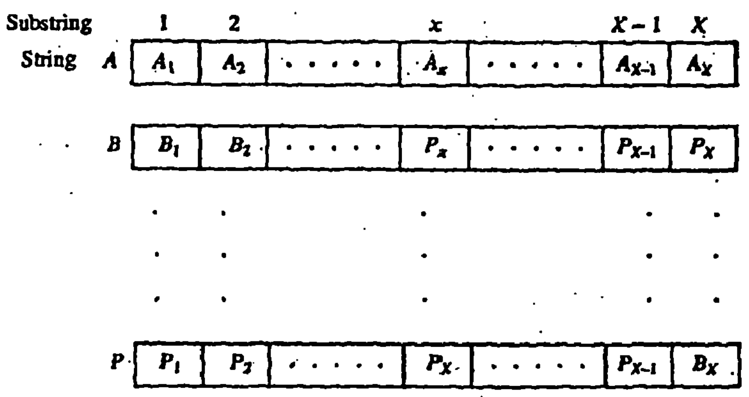


Figure 3-5 Population Strings

A simple genetic algorithm consists of three operators:

1. Reproduction
2. Crossover, and
3. Mutation.

These operators are described as follows:

- Reproduction

Reproduction is an operator in which an old string is copied into the new population according to that string's fitness. Fitness of string (solution) can be taken as the objective function value (maximisation problem) or its inverse (minimisation problem). For distribution networks, in general, fitness of a string can be represented by

$$f_i = \left(\frac{1}{C_{T_i}}\right)^s \qquad (3.120)$$

in which f_i = fitness of string i, $i = A,B, ..,P$; C_{T_i} = total cost of network represented by string i; and s = scaling exponent taken 1 in the early generation (so that GA can sort through the potential strength of the strings); but increase to 3 or 4 in subsequent generations to exaggerate small differences in fitness of strings.

Generation of new members in the next generation is based on probability of selection of string, i.e., p_i given by

$$p_i = \frac{f_i}{\sum_i f_i} \qquad (3.121)$$

Thus, reproduction is based on the survival of the fittest principle – more fit strings make more copies for mating than less fit strings.

- Crossover

Crossover, in its simplest form, is the partial exchange of corresponding segments between two parent strings to produce two offspring strings. The crossing point is decided randomly. Thus, two strings *A* and *B* as parent strings will produce two offspring string *A'* and *B'* after crossover with crossing point 2 as shown in Fig. 3-6.

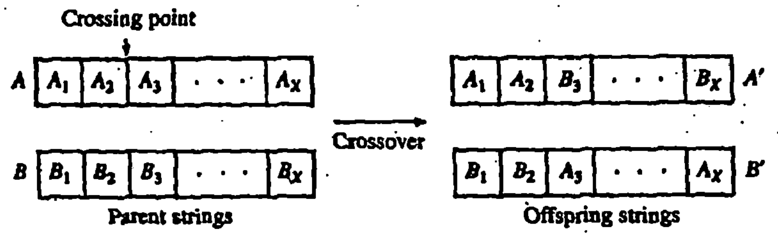


Figure 3-6 Crossover Process

The probability of crossover p_c is usually selected between 0.6 and 1.0. The GA randomly picks two strings from the new population. A uniformly distributed random number is then generated between 0.0 and 1.0. The GA applies the crossover operator if the random number is less than p_c, otherwise the two strings are retained as they are. For example, for a population size of 100 (P=100) and crossover probability of 0.7 (p_c = 0.7), on average

$P \times p_c = 100 \times 0.7 = 70$ strings are crossed over in each generation.

- Mutation

Mutation is an operation in which the mutation operator randomly alters a gene, i.e., a bit (0 to 1 and 1 to 0 in binary code) as shown in Fig. 3-7. Even though reproduction and crossover effectively search and recombine to produce next generation population, they may become overzealous and lose some useful genetic character (a 1 or 0 in a particular location). Mutation operator tries to protect against such irreparable loss.

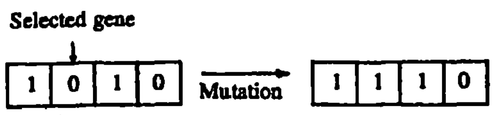


Figure 3-7 Mutation Process

Probability of mutation p_m is usually taken between 0.01 and 0.05.

3.5.1.3 Advantages and shortcoming

Genetic algorithms have a number of advantages over other mathematical programming techniques. In the context of optimisation of pipe network design some advantages include the following:

1. Genetic Algorithms deal directly with a population of solutions at any one time. These are spread throughout the solution space, so the chance of reaching the global optimum is reached significantly.
2. Each solution consists of a set of discrete pipe sizes. One does not have to round diameters up or down to obtain the final solution.
3. Genetic Algorithms identify a set of solutions of pipe network configurations that are close to the minimum cost solution. These configurations may correspond to quite different designs that can be then compared in terms of other important but non quantifiable objectives.
4. Genetic Algorithms use objective function or fitness information only, compared with the more traditional methods that rely on existence and continuity of derivative or other auxiliary information.
5. Genetic Algorithms can easily handle multiple sources and multiple loadings. It can be used for new designs as well as for rehabilitation, replacement and expansion of existing networks.

Despite the advantages, there is also shortcoming in using GA as tools of design of water distribution networks, i.e. Genetics Algorithm requires a large number of objective function evaluations and checking for their feasibility or infeasibility. Thus computer times, even for moderate network are quite large.

There are different steps involved in optimisation of gravity water distribution network through Genetic Algorithm are as follows

Step 1. Generation of initial population.

An initial population of coded strings, each string representing a solution that may be feasible or infeasible, is randomly generated.

Step 2. Computation of cost of each network.

Each string of population is decoded to obtain pipe sizes in the solution and then the network cost is obtained.

Step 3. Hydraulic analysis of each network.

Each network is analysed for the specified demand pattern to obtain link flows and nodal HGL values. These available HGL values at demand nodes, H_j^{avl} are then compared with minimum required HGL values, $H_j^{\min}$ and head deficit at each node, $H_j^{\min}$ - H_j^{avl} is noted.

Step 4. Computation of Penalty Cost.

Rather than ignoring infeasible solutions and considering only the feasible solutions, the infeasible solutions in the population are also considered through exterior penalty function method. A penalty cost for each demand pattern is assigned if the solution does not satisfy minimum HGL requirements. The HGL violation at the node at which the HGL deficit is maximum, [max $(H_j^{\min} - H_j^{avl}), j = 1,2,.., N$] is used as the basis for computation of the penalty cost. The maximum HGL deficit is multiplied by a penalty factor to obtain the penalty cost. The penalty factor is a measure of the cost of violating one unit of node HGL and can be taken equal to the capitalised cost of raising the total quantity of water at the source node by one unit. However, the value of the penalty function should be checked at the end of GA iteration to see that the best infeasible solution (solution with least penalty cost) is not superior to any feasible solution in the population. If so, the value of penalty factor is sufficiently increased.

Step 5. Computation of total cost

The cost of each solution in the current population is obtained as the sum of network and penalty costs, obtained in Steps 2 and 4, respectively.

Step 6. Computation of fitness.

Fitness of each string (solution) is then obtained by using Equation (3.120).

Step 7. Reproduction of new population.

Members of next generation based on the probability of selection of a string given by Equation (3.121) are reproduced.

Step 8. Crossover.

Crossover operation is then carried out to produce offspring strings from parent strings.

Step 9. Mutation

Mutation operation is carried out.

Step 10. Production of successive generations.

Successive generation are produced maintaining the size of the population. A set of least cost strings (e.g., the best 20) is stored and the set is updated as less costly alternatives are generated. Typically, generations between 100 and 1000 are evaluated.

The steps involved in optimisation of a pumped network are as follows:

Step 1-2. Step 1 and 2 are same as those for gravity network.

Step 3. Hydraulic Analysis of each network. A suitable HGL at pumped source node H_s, is either increased or decreased so that maximum HGL deficit [max $(H_j^{\min} - H_j^{avl})$, j = 1,2,.., N] or minimum HGL surplus [min $(H_j^{avl} - H_j^{\min})$, j = 1,2,.., N] is zero. The revised HGL, H_s is used in calculating pumping head h_p (=$H_s - H_c$); and the energy cost (present worth of energy charges) is obtained using first term Equation (2.2) that pointed energy cost.

Step 5. Computation of Total Cost.

The total cost is the sum of network and energy costs.

Step 6 – 10. Step 6 to 10 are same as those for gravity network.

3.5.1.4 GA Applications

The application of GA in water distribution network design is subject of interdisciplinary research and development. Because GA itself does not do the design of water distribution network, it need a number of things from different domains integrated together, including a water distribution design model – formulation of design objectives, design criteria/constraints and design variables, and also the hydraulic simulation model – solving the network hydraulics. By seamlessly integrating three of the models, designer will be able to use a GA code to design water distribution network.

Several researchers have demonstrated the application of GA in water distribution network, for example Dandy, Simpson and Murphy (1996) and Savic and Walters. (1997). One of such software is Water Network Optimiser (Savic and Keedwell). This program use Epanet 2.0 as its simulation engine and the problem must be inputted in Epanet input file format. The Water Network Optimiser is a simple method using genetic algorithms (both single objective – SOGA and multi objective – MOGA) to find optimal sets of pipe diameters for a water distribution network. The genetic algorithm (GA) uses the principles of evolution to test various combinations of pipes in the model to achieve two goals:

1. A minimum cost (bigger pipes more cost to add)
2. Certain pressure limits within the network (every network has head requirements at its nodes)

3.5.2 Simulated Annealing Method

Simulated annealing method is search approach based on the analogy with the physical annealing process. In physical annealing process the temperature of molecules is increased sufficiently high so that they become highly mobile and can attain different states. If the molecules are then gradually cooled from this initial high temperature, they attain a crystalline structure, an optimal one corresponding to minimum energy state. Cunha and Sousa (1999) applied it to the optimal design of looped water distribution networks.

In each step of the algorithm, a change of configuration is produced, and then its cost is evaluated. The new configuration is chosen at random in the neighbourhood of the current configuration s_j. In this algorithm, the neighbourhood includes the configurations having all the pipes, but one, with the same diameter as in the current configuration. The pipe having a different diameter can take either a diameter one size above or one size below its current diameter. The new configuration is accepted or not, according to the Metropolis criterion (p<min[1,exp(Δc)/t}). If it is accepted, this configuration will be used as the starting point for the next step. If not, the original configuration will play this role.

If t is decreased at a suitable rate, the system will tend to converge to the global least–cost configuration as the number of transition attempts increases. This property of convergence to a global optimum cost configuration derives from the fact that transitions from low to high cost configurations are not automatically excluded. They will take place or not depending on the difference between costs and on the level of temperature. Initially, even very negative (counteroptimum) transitions will be accepted; as the temperature falls, the acceptance of such transitions will become increasingly rare. By accepting worsening moves, the annealing algorithm will, in principle, avoid being trapped in local optima.

Simulated annealing method requires the following parameters:

1. Elasticity of Acceptance, a

   This parameter represents the probability of accepting a transition from an initial cost solution to higher cost solution. The value may be between 0.2 to 0.9, the higher the value the more is the probability of covering the entire space and reaching the global optimum solution; however the computation time would be more

2. Minimum Number of Iterations, n_1

   This parameter represents the minimum number of iterations that will be performed before decreasing the temperature even if there is no more improvement in the current solution. The value may be between 10 and 70, the higher the value the more is the probability of reaching global optimum; however computation time is more.

3. Initial Temperature, T_i

   This parameter is the initial temperature at which the annealing process is stated. It may be taken as

$$T_i = -\frac{0.1C_i}{\ln a} \tag{3.122}$$

in which C_i = cost of the initial solution; and a = elasticity of acceptance.

4. Final Temperature, T_f

   This parameter represents the temperature at which the annealing process is stopped. Lower the value, the probability of reaching the global optimum solution increases; however, the computation time also increases.

5. Cooling Factor, r

   This parameter represents the rate at which the temperature is decreased whenever a temperature decrease should occur. Thus,

   $$T_{j+1} = rT_j \qquad (3.123)$$

   in which T_j and T_{j+1} = temperature at steps j and j+1, respectively. The value of r may be between 0.1 and 0.9 (even up to 0.99) and may be a constant value for all steps or may vary from step to step. Faster cooling requires less computer time but may give sub optimal solutions.

6. Number of Temperature Decreases, n_2

This parameter represents the number of temperature decreases that will be performed without an improvement in the current optimum before stopping the algorithm. The value of n_2 may be between 2 and 7.

The algorithm for optimal design of water distribution networks through simulated annealing method consists of following steps (Cunha and Sousa, 1999):

Step 1. Choose s_i, the initial configuration of the network satisfying all constraints. Find its cost, C_i.

Step 2. Choose T_i, the initial temperature according to Equation (3.122).

Step 3. Choose T_f, the final temperature.

Step 4. Choose at random another configuration s_j, in the neighbourhood of the current configuration s_i, by changing the diameter of any one link to either one size higher or one size lower than the current one. Test it for feasibility. If acceptable, find the change in cost ΔC.

Step 5. Choose at random $p \in [0,1]$

Step 6. If $p < \min\,[1, \exp\,(\Delta C/T_j)]$, accept the changed configuration, otherwise retain the original configuration.

Step 7. Choose another temperature, less than the earlier one.

Step 8. Continue step 4 to 7 until the final temperature is reached.

Step 9. The last solution is the optimal solution.

The advantages and shortcoming of GA are applicable to simulated annealing also.

3.6 SUMMARY

The water distribution network design problem is non linear in nature. Solution of this problem can be achieved by using NLP approach with direct or indirect method. Generalized Reduced Gradient (GRG) algorithm that is implemented in GRG2 has been worldwide applied and fully-tested and becomes a robust algorithm backed by more than 15 years of solving real-world problems in the petroleum, chemical, defence, financial, agriculture, and process control industries. GRG2 is like other gradient-based methods, guaranteed to find a local optimum only on problems with continuously differentiable functions, and then only in the absence of numerical difficulties (such as degeneracy or ill conditioning). However, GRG2 has a reputation for robustness, compared to other nonlinear optimization methods, on difficult problems where these conditions are not fully satisfied. The application of this method will be discussed in next chapter for solving water distribution network design problem

Due to the solution that is achieved by using Non Linear Programming approach cannot guarantee to be global optimum value, Random Search Techniques will be adopted. Genetic Algorithms (GA) that is part of Random Search Techniques, will be applied in solving water distribution network design problem. Genetic Algorithm is able to work for complex network where there would be a lot of combination of links to form one global solution. The application of Genetic Algorithm in solving water distribution network design problem will be discussed in next chapter.

CHAPTER IV

CASE STUDY

4.1 APPLICATION OF NON LINEAR PROGRAMMING

As we have already known, for Non Linear Programming approach, there are 3 types of formulation, i.e. D-Q, D-h and Q-h formulation. In all of these formulations, the constraints must be satisfied, i.e. node flow continuity, summation of headloss in every loop equal zero and minimum required pressure at each node. From previous chapter, it has been concluded that Generalized Reduced Gradient (GRG) algorithm is capable enough to solve the real-world problems. Microsoft Excel Solver is one example of software that using GRG algorithm in their calculation. This software application will be discussed further in next paragraphs.

4.1.1 Microsoft Excel Solver

Microsoft Excel Solver is licensed product from Microsoft Corporation. Microsoft Excel Solver uses the Generalized Reduced Gradient (GRG2) algorithm for optimizing nonlinear problems. This algorithm was developed by Leon Lasdon, of the University of Texas at Austin, and Allan Waren, of Cleveland State University. For linear and integer problems, the simplex method, with bounds on the variables and the branch and bound method are used, which is implemented by John Watson and Dan Fylstra, of Frontline Systems, Inc.

GRG2 uses an implementation of the generalized reduced gradient (GRG) algorithm. It seeks a feasible solution first (if one is not provided) and then retains feasibility as the objective is improved. It uses a robust implementation of the BFGS quasi-Newton algorithm as its default choice for determining a search direction. A limited-memory conjugate gradient method is also available, permitting solutions of problems with hundreds or thousands of variables. The problem Jacobian is stored and manipulated as a dense matrix, so the effective size limit is one to two hundred active constraints (excluding simple bounds on the variables, which are handled implicitly).

Microsoft Excel Solver uses iterative numerical methods that involve "plugging in" trial values for the adjustable cells and observing the results calculated by the constraint cells and the optimum cell. Each trial is called an iteration. Because a pure trial-and-error approach would be extremely time-consuming (especially for problems involving many adjustable cells and constraints), Microsoft Excel Solver performs extensive analyses of the observed outputs and their rates of change as the inputs are varied, to guide the selection of new trial values.

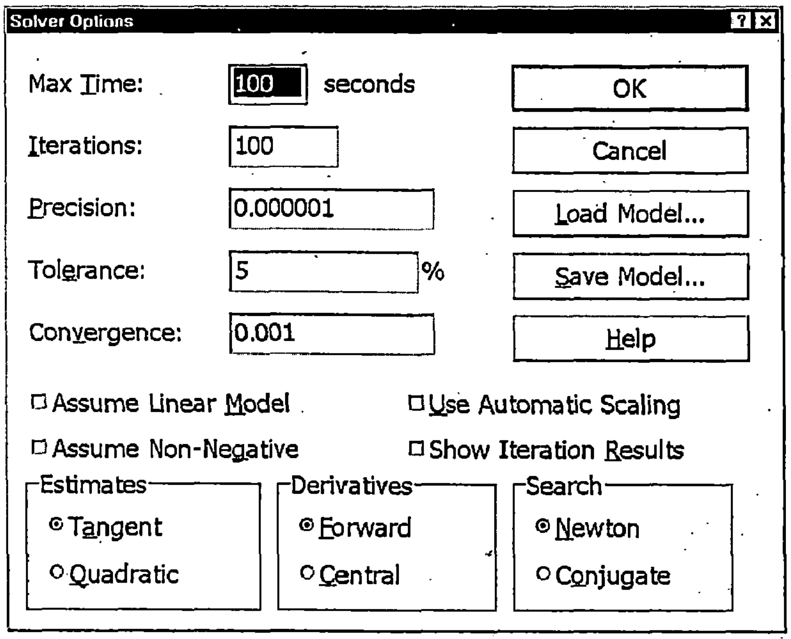
In a typical problem, the constraints and the optimum cell are functions of (that is, they depend on) the adjustable cells. The first derivative of a function measures its rate of change as the input is varied. When there are several values entered, the function has several partial derivatives measuring its rate of change with respect to each of the input values; together, the partial derivatives form a vector called the gradient of the function.

Derivatives (and gradients) play a crucial role in iterative methods in Microsoft Excel Solver. They provide clues as to how the adjustable cells should be varied. For example, if the optimum cell is being maximized and its partial derivative with respect to one adjustable cell is a large positive number, while another partial derivative is near zero, Microsoft Excel Solver will probably increase the first adjustable cell's value on the next iteration. A negative partial derivative suggests that the related adjustable cell's value should be varied in the opposite direction.

4.1.1.1 Forward and Central Differencing

Microsoft Excel Solver approximates the derivatives numerically by moving each adjustable cell value slightly and observing the rate of change of each constraint cell and the optimum cell. This process is called a finite difference estimate of the derivative. Microsoft Excel Solver can use either forward differencing or central differencing, as controlled by the Derivatives option on the Solver Options dialog box which is shown in Figure 4-1.

Forward differencing uses a single point (that is, a set of adjustable cell values) that is slightly different from the current point to compute the derivative, while central differencing uses two points in opposite directions. Central differencing is more accurate if the derivative is changing rapidly at the current point, but requires more recalculations. The default choice is forward differencing, which is fine in most situations.


Solver Options

Max Time: 100 seconds
Iterations: 100
Precision: 0.000001
Tolerance: 5 %
Convergence: 0.001

OK
Cancel
Load Model...
Save Model...
Help

☐ Assume Linear Model
☐ Use Automatic Scaling
☐ Assume Non-Negative
☐ Show Iteration Results

Estimates: ◉ Tangent ○ Quadratic
Derivatives: ◉ Forward ○ Central
Search: ◉ Newton ○ Conjugate

Figure 4-1 Solver Options Menu in MS-Excel Solver

Linear problems can be solved with far less work than nonlinear problems; Microsoft Excel Solver does not need to recompute changing derivatives, and it can extrapolate along straight lines instead of recalculating the worksheet. These time savings are brought into play when user clicks to select the Assume Linear Model check box in the Solver Options dialog box. If the user doesn't select this box, Microsoft Excel Solver can still solve the problem, but it will spend extra time doing so.

When we know that a problem is completely linear, selecting the Assume Linear Model option will speed up the solution process by a factor of 2 to 20 (depending on the size of the worksheet). The downside is that, if the real worksheet formulas are nonlinear and this option is selected, we solve the wrong problem.

Although Microsoft Excel Solver does check the final solution when Assume Linear Model is checked, using a full worksheet recalculation, this is not an absolute guarantee that the problem is truly linear. We can always recheck the solution by running the same problem with the check box cleared.

4.1.1.2 Optimality Conditions

Because the first derivative (or gradient) of the optimum cell measures its rate of change with respect to (each of) the adjustable cells, when all of the partial derivatives of the optimum cell are zero (that is, the gradient is the zero vector), the first-order conditions for optimality have been satisfied (some additional second-order conditions must be checked as well), having found the highest (or lowest) possible value for the optimum cell.

4.1.1.3 Multiple Locally Optimum Points

Some problems have many locally optimum points where the partial derivatives of the optimum cell are zero. A graph of the optimum cell function in such cases would show many hills and valleys of varying heights and depths. When started at a given set of adjustable cell values, the methods used by Microsoft Excel Solver will tend to converge on a single hilltop or valley floor close to the starting point. But Microsoft Excel Solver has no sure way of knowing whether there is a taller hilltop, for example, in some distance away.

The only way to find the global optimum is to apply external knowledge of the problem. Either through common sense reasoning about the problem or through experimentation, the user must determine the general region in which the global optimum lies, and start Microsoft Excel Solver with adjustable cell values that are within that region. Alternatively, we can start Microsoft Excel Solver from several different, widely separated points and see which solution is best.

The maximum number of constraints and variables that can be handled by Microsoft Excel Solver is given as follows:

- For Non Linear Programming problem, the maximum number of variables is 200, and the maximum number of constraints is 100.

- For Linear Programming problem, the maximum number of variables is 200, and the maximum number of constraints is unlimited.

4.1.2 Working with Solver

Solver is part of a suite of commands sometimes called "what-if analysis" tools, i.e. process of changing the values in cells to see how those changes affect the outcome of formulas on the worksheet, for example, varying the interest rate that is used in an amortization table to determine the amount of the payments. Optimization model in Solver has three parts: the target cell, the changing cells, and the constraints.

4.1.2.1 Target cell

The target cell represents the objective or goal. We want to either minimize or maximize the target cell.

4.1.2.2 Changing cells

Changing cells are the spreadsheet cells that we can change or adjust to optimize the target cell.

4.1.2.3 Constraints

Constraints are restrictions we place on the changing cells. In most Solver models, there is an implicit constraint that all changing cells must be nonnegative. With Solver, we can find an optimal value for a formula in one cell, called the target cell, on a worksheet. Solver works with a group of cells that are related, either directly or indirectly, to the formula in the target cell. Solver adjusts the values in the changing cells that we already specify, called the adjustable cells, to produce the result that we specify from the target cell formula. We can apply constraints to restrict the values in the model, and the constraints can refer to other cells that affect the target cell formula.

4.1.2.4 Installing and running Solver

To Install Solver, we can click **Add-Ins** on the **Tools** menu, and then select the **Solver Add-in** check box. Then we click **OK**, and Excel will install the Solver. Once the add-in is installed, we can run Solver by clicking **Solver** on the **Tools** menu.

Figure 4-2 shows the **Solver Parameters** dialog box, in which we input the target cell, changing cells, and constraints that apply to our optimization model.

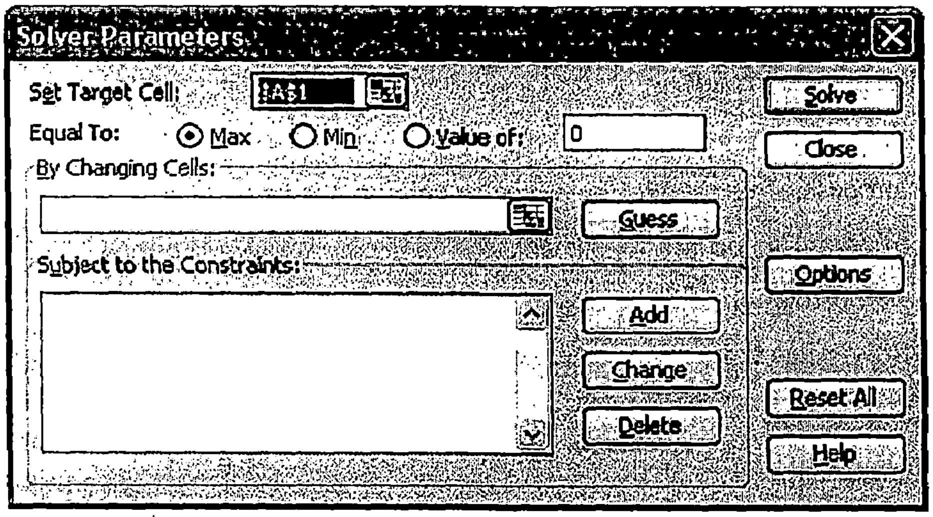


Figure 4-2 Solver Parameters Dialog Box

After we have input the target cell, changing cells, and constraints, Solver is doing calculating to find the feasible solution. Any specification of the changing cells that satisfies the model's constraints is known as a feasible solution. For instance, in looped water distribution network problem, network that satisfies the following three conditions would be a feasible solution:

- Continuity of flow in each node.
- Summation of headloss in every loop equal zero.
- Minimum required pressure at each node is satisfied.

Essentially, Solver searches over all feasible solutions and finds the feasible solution that has the "best" target cell value (the largest value for maximum optimization, the smallest for minimum optimization). Such a solution is called an **optimal solution**. Some Solver models have no optimal solution and some have a unique solution. Other Solver models have multiple (actually an infinite number of) optimal solutions.

4.1.3 Application of Solver

Solver is a useful tool in solving Water Distribution Network problem. The previous problem in Chapter II, i.e. Example 2.1. will be solved using Solver.

4.1.3.1 D-Q Formulation

In D-Q formulation, the initial value of Diameter (D) and Discharge (Q) are assumed. The number of trial in my experiment is 10 trials and the complete results are presented in Appendix A. The final result is given in Table 4-1.

Table 4-1 Result of Example 2.1 (using D-Q Formulation)

Link	Diameter (m)	Discharge (m^3/h)	Head loss (m)
1	0.480	0	5.33
2	0.262	370	13.10
3	0.395	650	5.03
4	0	0	0
5	0.372	530	4.64
6	0.253	200	5
7	0.238	270	11.57
8	0	0	0

Table 4-2 Head pressure at each node (using D-Q Formulation)

Node	*Pressure Head (m)*	
	Actual	*Required*
1	210	210
2	204.67	180
3	191.57	190
4	199.64	185
5	180	180
6	195	195
7	190	190

The cost of this network is Rs. 17,172,733.

4.1.3.2 D-h Formulation

In D-h formulation, the initial value of Diameter (D) and headloss (h) are assumed. The number of trial in my experiment is 8, and the complete results are presented in Appendix A. The final result is given in Table 4-3.

Table 4-3 Result of Example 2.1 (using D-h Formulation)

Link	Diameter (m)	Head loss (m)	Discharge (m^3/h)
1	0.499	4.5	1120
2	0.298	6.4	347.65
3	0.381	6.5	672.36
4	0.104	5.4	20.25
5	0.385	4	532.10
6	0.336	1.3	202.10
7	0.270	5.5	247.65
8	0.099	0.1	2.10

Table 4-4 Head pressure at each node (using D-h Formulation)

Node	*Pressure Head (m)*	
	Actual	*Required*
1	210	210
2	205.5	180
3	199.1	190
4	199	185
5	193.6	180
6	195	195
7	193.7	190

The cost of this network is Rs. 20,181,642.

4.1.3.3 Q-h Formulation

In D-h formulation, the initial value of Diameter (D) and headloss (h) are assumed. The number of trial in my experiment is 6, and the complete calculation is presented in Appendix A. The final result is given in Table 4-5.

Table 4.5 Results of Example 2.1 (using Q-h Formulation)

Link	Discharge (m^3/h)	Head loss (m)	Diameter (m)
1	1120	5.33	0.480
2	370	13.10	0.262
3	650	5.03	0.395
4	0	0	0
5	530	4.64	0.372
6	200	5	0.253
7	270	11.57	0.238
8	0	0	0

Table 4-6 Head pressure at each node (using Q-h Formulation)

Node	*Pressure Head (m)*	
	Actual	*Required*
1	210	210
2	204.67	180
3	191.57	190
4	199.64	185
5	180	180
6	195	195
7	190	190

The cost of this network is Rs. 17,168,950.

4.1.4 Discussion of results

The selections of initial values in all three formulations are very important is iteration process, because if the values that are selected are far from optimum values, the trial process will be quite tough and takes a lot of time. The result of running process will become an input for next running.

The process in getting the optimum point for three formulations is graphically presented in Figure 4-3. This graphic shows us that it is better to use Q-h formulation rather than D-Q or D-h formulation, because the number of iteration that is required to achieve local optimum solution is less, and so computation time will be reduced. Also the solution that is achieved by using Q-h formulation is more economical than the other two formulations. Based of this fact, Q-h formulation will be applied in solving a real world water distribution network problem in next section.

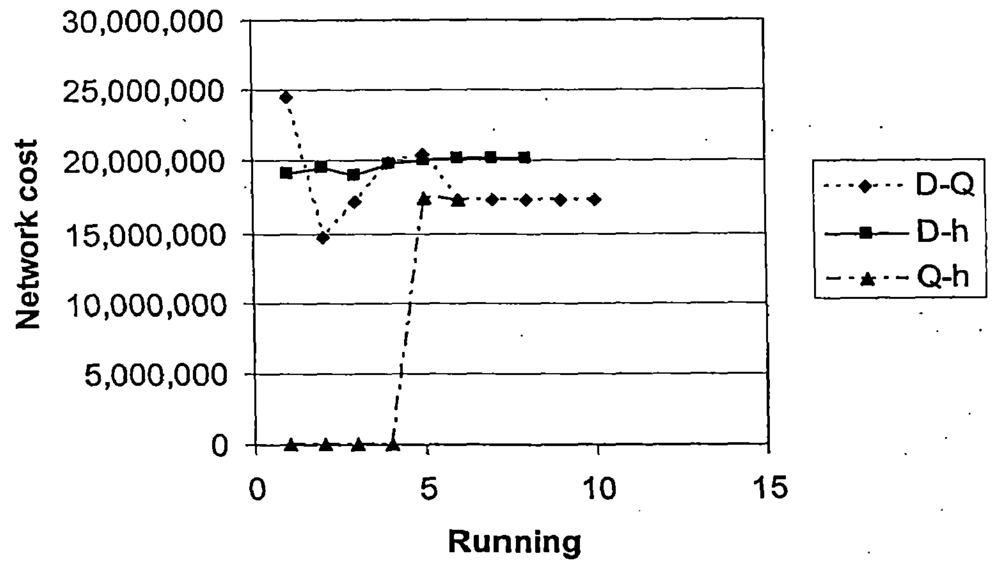


Figure 4-3 Iteration Process using MS-Excel Solver

4.2 APPLICATION OF GENETIC ALGORITHM

The same problem will be solved by using Genetic Algorithm (GA) application. There are many GA applications on water distribution network. In this dissertation, Water Network Optimiser is presented which is developed by Dr. E. Keedwell from Centre for Water Systems, University of Exeter, UK. This particular program that is used in this dissertation is a demo program, which can only run for maximum 50 pipes. The genetic algorithm (GA) uses the principles of evolution to test various combinations of pipes in the model to achieve two goals:

1. A minimum cost of network
2. Pressure of each node must be greater than minimum head requirements.

4.2.1 Illustrative Example

A simple water distribution network will be solved using Water Network Optimiser. It is taken from previous problem in Example 2.1 with different cost of pipe. The available pipe sizes in inches (1 inch = 25.4 mm) and their unit cost in arbitrary units shown in Table 4-7.

Table 4-7 Diameter – Cost relationship of network in Example 2-1

Diameter (inch)	Cost (monetary unit/m')
1	2
2	5
3	8
4	11
6	16
8	23
10	32
12	50
14	60
16	90
18	130
20	170
22	300
24	550

Hazen-Williams head loss formula is used for headloss calculation, with Hazen Williams coefficient is 130 for all links.

4.2.2 Modelling the problem

Water Network Optimiser uses Epanet 2.0 (EPA - Govt. of USA, 2000) as its simulation engine and therefore no other modelling packages can currently be used with it. The Water Network Optimiser uses a simple text file which contains the details of the current project which has the extension `*.prj`. The file must be written correctly for the program to work and consists of 4 sections, i.e.:

1. Input File
2. Costs
3. Modify Links, and
4. Solution Sets

All of section is discussed as follows:

`[InputFile]`

This section shows the location of the Epanet *.inp file which can be exported from the Epanet 2.0 program. This can either include a full path or just the filename.

`[Costs]`

This section shows the costs in monetary units/ unit length of replacing a pipe within the network. The first number is the diameter (in mm), and the second is the cost per unit length of the pipe (in monetary units). They must be Tab separated.

`[RequiredHead]`

This section shows the required head at each node within the network. The first number is the nodeID as found in the Epanet model and the second is the minimum required head at that node. If the resulting model does not achieve this head then a penalty can be applied. Again, these values must be Tab separated.

`[ModifyLinks]`

This is a list of pipes which can be modified within the network. In this case, it is allowed all the section of network to be optimised. The numbers must correspond to nodeID in the Epanet model.

`[SolutionSets]`

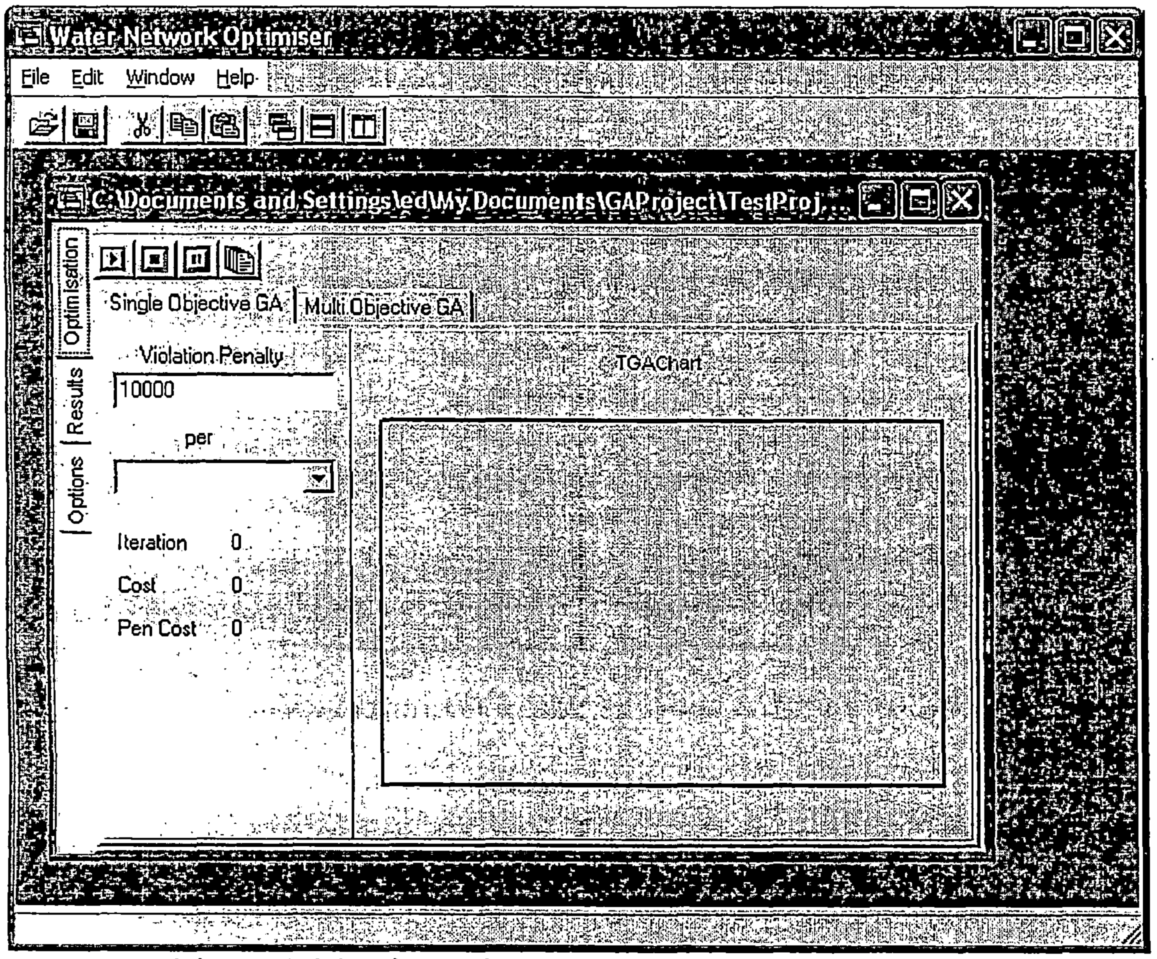
This section is used by the program to point to solution sets from previous runs saved into the project. There must be a blank line between each section.

The project file of above illustrative example is presented in Appendix B.

4.2.3 Running the Model

4.2.3.1 Project Window

Once this project file has been written, it can be opened, where the screen as shown in Figure 4-4 should appear.



Violation Penalty
10000
per

Iteration 0
Cost 0
Pen Cost 0

Figure 4-4 Project Window of Water Network Optimiser

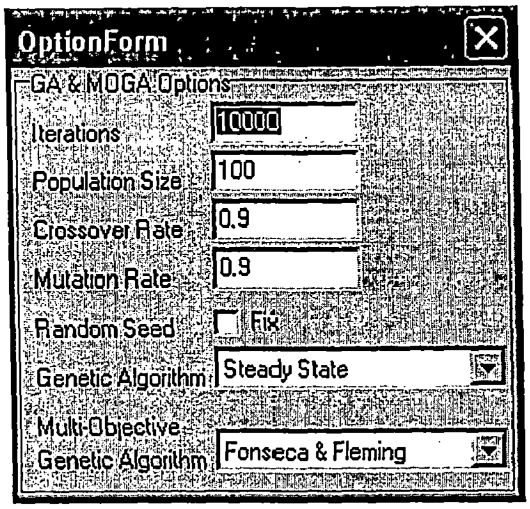
There are three tabs in the window, **Optimisation** where the GA runs are completed, **Results** where the results from runs can be seen and manipulated and **Options** where the Project Options (as specified in the prj file) can be viewed and changed.

4.2.3.2 Optimisation

On this page are the controls for optimising the water network. There are four buttons which control various aspects of the GA as shown in Figure 4-5.

Figure 4-5 Key buttons of Water Network Optimiser software

The play button will start the GA. The GA can be stopped and the solutions saved by pressing the stop button. The pause button will pause the process, to resume we can click play. The final button will show the GA options form where various parameters can be changed in OptionForm menu, as shown in Figure 4-6.



OptionForm

GA & MOGA Options

Iterations: 10000

Population Size: 100

Crossover Rate: 0.9

Mutation Rate: 0.9

Random Seed: [ ] Fix

Genetic Algorithm: Steady State

Multi-Objective Genetic Algorithm: Fonseca & Fleming

Figure 4-6 GA option of Water Network Optimiser

Each of these options affects the way the GA works.

Iterations: Specifies the number of iterations before the GA stops

Population Size: Determines the number of individuals in the population

Crossover Rate: Determines the probability of crossover (0.0-1.0)

Mutation Rate: Determines the probability of mutation (0.0-1.0)

Random Seed: As a default, each time the GA is run it begins from a new random position. To start the GA from the same point each time, we check the "Fix" box and enter the fixed random seed.

Genetic Algorithm: There are two types of GA used in this application, Steady State and Generational.
Multi-Objective Genetic Algorithm: There are two types of MOGA used in this program, Fonseca and Fleming and NSGA-II.

4.2.4 Discussion of results

The program is running after reading the input data in Epanet format. The network is modelled first in Epanet format. Then the project file is developed. Then we select the parameter of Genetic Algorithm, such as: Number of iterations, Population size, crossover rate, mutation rate, and type of Genetic Algorithm. A number of trials are applied, with different values of parameters. The results are given below.

First running:

The parameters are as follows:

Violation Penalty	10000
Population size :	100
Iterations:	10000
Crossover rate:	0.9
Mutation rate:	0.01
GA Type:	Generational

We can see the running process of the program from start up to terminal iterations. Here the price is fluctuated and decreasing, and after some number iterations, the fluctuation will stabilise and the optimise result will be achieved. The animated graphic that shows the process is presented on Figure 4.7. The better result will be achieved if we use Generational types of GA, rather than Steady state, even though time consumed will be more. For two looped network, with 10000 iterations, the Pentium 4 ® computer needs 5 minutes to find the solution. The optimal solution in this trial is given in the Table 4-8.

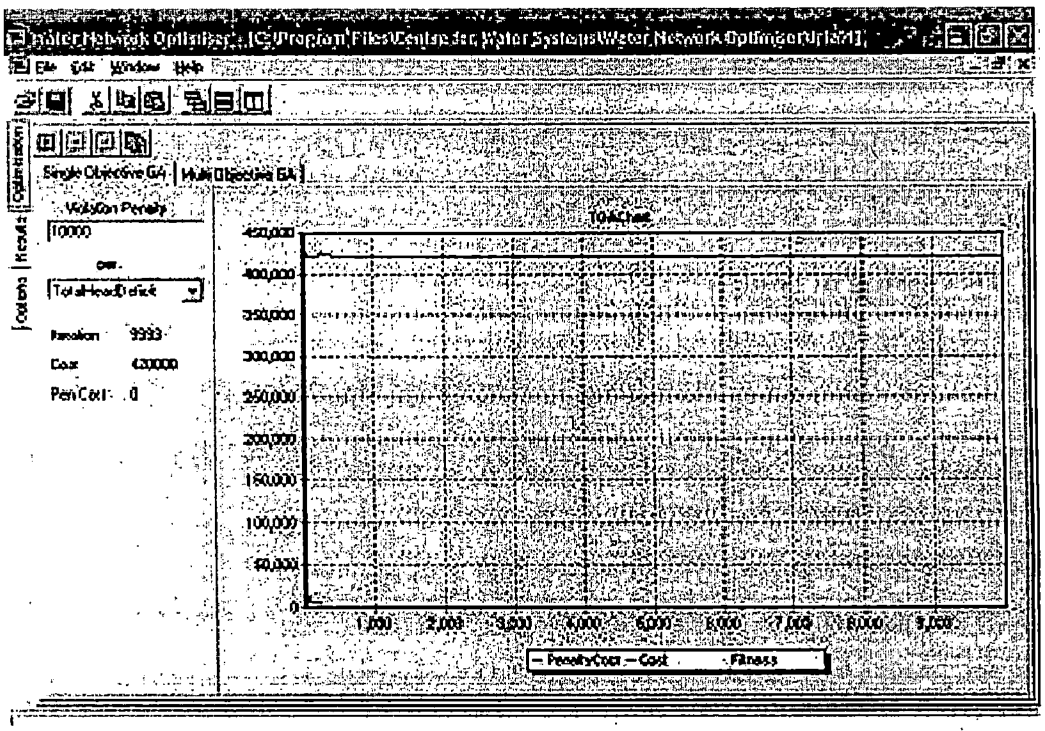

Violation Penalty
10000
Total Head Deficit
Iteration 9999
Cost 420000
PenCost 0

Figure 4-7 Iteration process in Water Optimiser Network

Table 4-8 Diameter of pipe of network (First running)

Link	Diameter (m)	Diameter (in)
1	0.4572	18
2	0.3048	12
3	0.4064	16
4	0.254	10
5	0.3556	14
6	0.1524	6
7	0.254	10
8	0.254	10

The head pressure at each node and the required head is given in Table 4-9.

Table 4.9 Head pressure at each node of network (First running)

Node	*Pressure Head (m)*	
	Actual	*Required*
1	210	210
2	203.25	180
3	198.62	190
4	198.15	185
5	193.09	180
6	195.03	195
7	190.28	190

The cost of water network is 442,000 units.

<u>Second running:</u>

The parameters are as follows:

Violation Penalty	10000
Population size:	100
Iterations:	10000
Crossover rate:	0.9
Mutation rate:	0.05
GA Type:	Generational

The results for this trial are given in Table 4-10 and 4-11.

Table 4-10 Diameter of pipe of network (Second running)

Link	Diameter (m)	Diameter (in)
1	0.4572	18
2	0.3556	14
3	0.3556	14
4	0.0254	1
5	0.3556	14
6	0.1524	6
7	0.3556	14
8	0.2540	10

Table 4-11 Head pressure at each node (Second running)

Node	*Pressure Head (m)*	
	Actual	*Required*
1	210	210
2	203.25	180
3	197.66	190
4	198.13	185
5	193.89	180
6	195.06	195
7	190.95	190

The cost of water network is 420,000 units, and this is the optimum one. From Figure 4-7, we can see that in the early stage of iterations, the cost of network is decrease progressively and in the later stage decreasing process is reduced and finally gives constant value, which is the optimum one. Since the direction of searching the optimum cost is random, the chance of getting the global optimum solution is high.

In GA approach, the program can work with discrete value of pipe diameter as a decision variable, where in Non Linear Programming approach can not handle discrete variables. In Non Linear Programming approach, the diameters that are given in solution are fractional number, so they need to be rounded up to get the market size diameter of pipe.

It is showed that the use of Genetic Algorithm Application in water distribution network problem is more promising than Non Linear Programming Approach. The application of Water Network Optimiser shows that the solution is better than Non Linear Programming approach.

4.3 APPLICATION ON DESIGN OF REAL TYPE NETWORK

In a real world, the network will be more complex than previous example. An example of real water distribution network problem is given in Example 4.1. and this problem will be solved using NLP approach and GA approach.

Example 4.1

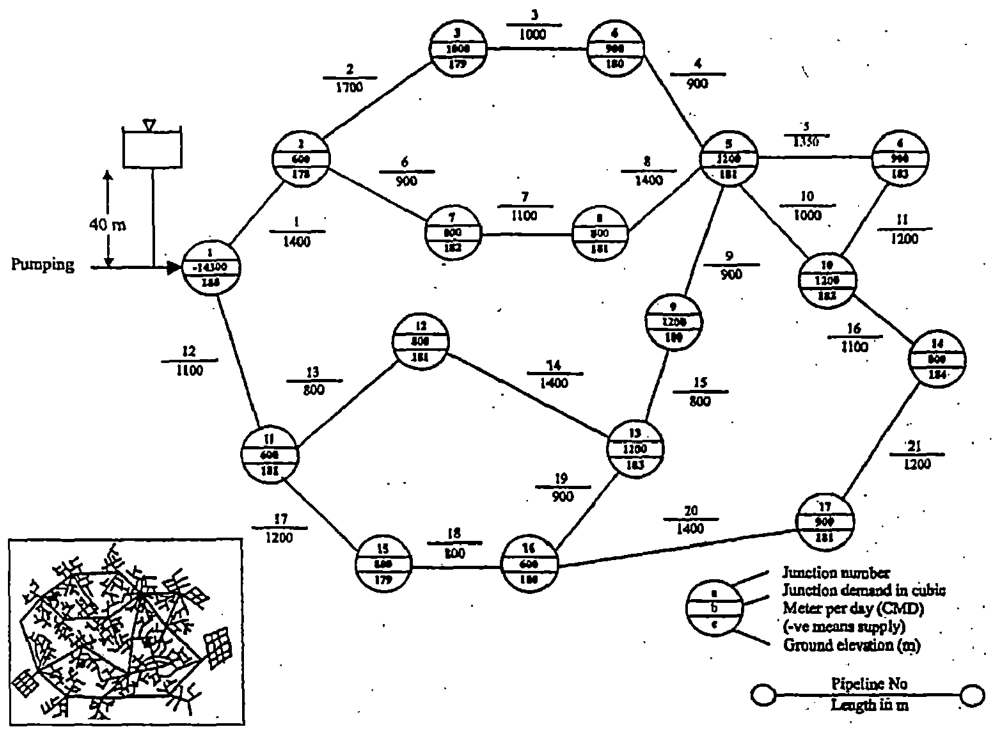


Figure 4-8 An example of real type water distribution network

Water distribution network on one city is proposed and presented in Figure 4-8. Ground level and demand node at each node are presented on Table 4-12.

Table 4-12 Demand node and pressure head requirement of Example 4.1

Node	*Demand (m^3/day)*	*Ground level*
1	-14300	180
2	600	178
3	1000	179
4	900	180
5	1200	181
6	900	183
7	800	182
8	800	181
9	1200	180
10	1200	182
11	600	181
12	800	181
13	1200	183
14	800	184
15	800	179
16	600	180
17	900	181

The required pressure head at each node is 17 m.

The information about links characteristics (length and roughness) are given in Table 4-13. Head loss is computed by using Hazen Williams formula.

Table 4-13 Links Characteristics of network in Example 4.1

Link	*Length (m)*	*Roughness Coeff.*
1	1400	100
2	1700	100
3	1000	100
4	900	100
5	1350	100
6	900	100
7	1100	100
8	1400	100
9	900	100
10	1000	100
11	1200	100
12	1100	100
13	800	100
14	1400	100
15	800	100
16	1100	100
17	1200	100
18	800	100
19	900	100
20	1400	100
21	1200	100

The information about cost of each diameter pipe is given in Table 4-14.

Table 4-14 Cost diameter pipe relationship of Example 4.1

Diameter (mm)	*Cost (monetary unit/m')*
25	2
50	5
75	8
100	11
150	16
200	23
250	32
300	50
350	60
400	90
450	130
500	170
550	300
600	550

4.3.1 Solution by using Non Linear Programming approach

The relationship of diameter of pipe (in m) and cost of pipe (in monetary unit) is presented in Table 4-15.

The equation that depicts the relationship of diameter and cost of pipe should be carried out in order to get the objective function. This is can be done by using MS-Excel. First, the points are plotted on log-log paper and the regression line is drawn so that it will give the best fit of the distribution of the points. The distribution of points and regression line is given in Figure 4-9.

Table 4-15 **Diameter – Cost relationship in mm and monetary unit**

Diameter (m)	Cost (monetary unit/m')
0.025	2
0.050	5
0.075	8
0.100	11
0.150	16
0.200	23
0.250	32
0.300	50
0.350	60
0.400	90
0.450	130
0.500	170
0.550	300
0.600	550

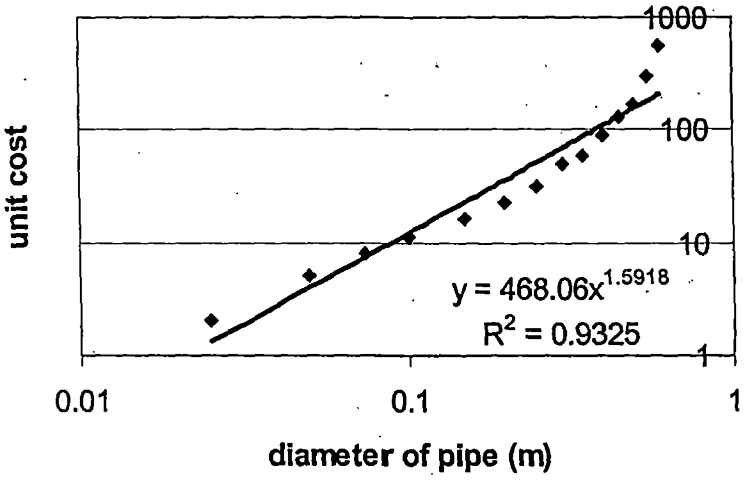


Figure 4-9 Diameter-cost pipe relationship

So the pipe cost equation is :

$C = 468.06\,D^{1.5918}$

where:

C = Pipe cost per m' (in monetary units)

D = Diameter of pipe (m)

The next step is to develop initial distribution of the flow on the network. By using this assumption, the constraints are developed, i.e. flow continuity on every node, sum of headloss on every loop is equal zero, and non negativity value of disharge as well as headloss. The flow direction is drawn in Figure 4-10.

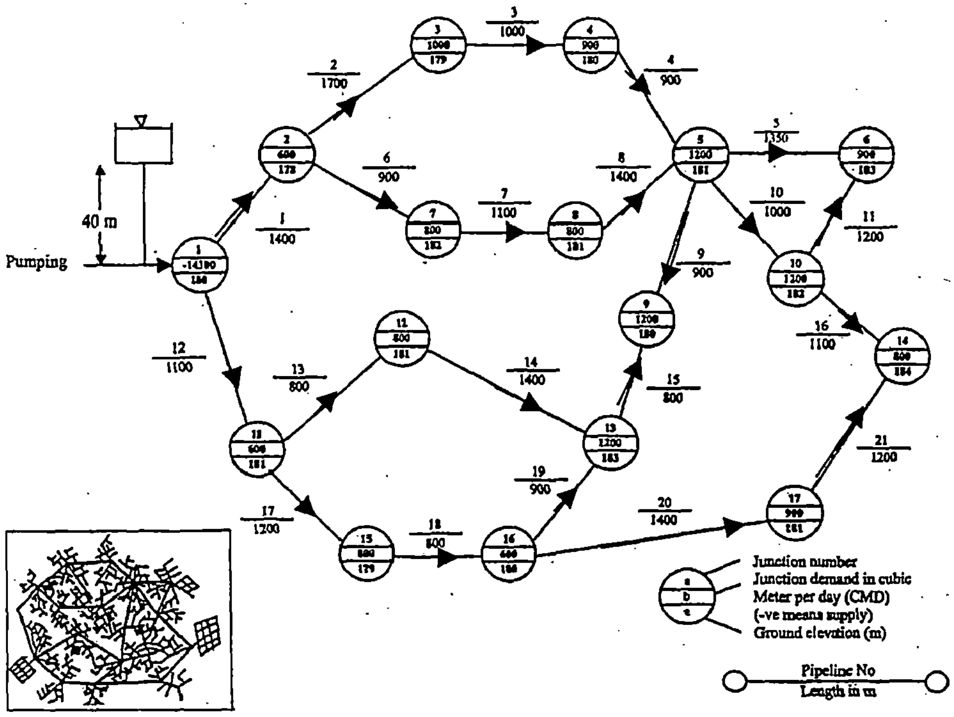


Figure 4-10 Initial guess of flow direction

The problem is formulated in Q-h formulation, and the formulation is given as follow:

Objective function:

$$\text{Minimise } C = 0.0642\sum_{i=1}^{21} L_i^{1.32686} Q_i^{0.60535} h_i^{-0.32686}$$

subject to the constraints as follows:

Node flow continuity constraints

$Q_1+Q_{12} = 14300$

$Q_1-Q_2-Q_6 = 600$

$Q_2-Q_3 = 1000$

$Q_3-Q_4 = 900$

$Q_4+Q_8-Q_5-Q_9-Q_{10} = 1200$

$Q_5+Q_{11} = 900$

$Q_6-Q_7 = 800$

$Q_7-Q_8 = 800$

$Q_9+Q_{15} = 1200$

$Q_{10}-Q_{11}-Q_{16} = 1200$

$Q_{12}-Q_{13}-Q_{17} = 600$

$Q_{13}-Q_{14} = 800$

$Q_{14}+Q_{19}-Q_{15} = 1200$

$Q_{16}+Q_{21} = 800$

$Q_{17}-Q_{18} = 800$

$Q_{18}-Q_{19}-Q_{20} = 600$

$Q_{20}-Q_{21} = 900$

Summation of Headloss equal to 0 's constraints

$h_1+h_6+h_7+h_8+h_9-h_{15}-h_{14}-h_{13}-h_{12} = 0$

$h_2+h_3+h_4-h_8-h_7-h_6 = 0$

$h_5-h_{11}-h_{10} = 0$

$h_{13}+h_{14}-h_{19}-h_{18}-h_{17} = 0$

$h_{19}+h_{15}-h_9+h_{10}+h_{16}-h_{21}-h_{20} = 0$

Path Headloss constraints

$h_1 \leq 25$

$h_1+h_2 \leq 24$

$h_1+h_2+h_3 \leq 23$

$h_1+h_2+h_3+h_4 \leq 22$

$h_1+h_6 \leq 21$

$h_1+h_6+h_7 \leq 22$

$h_1+h_6+h_7+h_8 \leq 22$

$h_1+h_2+h_3+h_4+h_5 \leq 20$

$h_1+h_6+h_7+h_8+h_5 \leq 20$

$h_1+h_2+h_3+h_4+h_{10}+h_{11} \leq 20$

$h_1+h_6+h_7+h_8+h_{10}+h_{11} \leq 20$

$h_1+h_2+h_3+h_4+h_{10} \leq 21$

$h_1+h_6+h_7+h_8+h_{10} \leq 21$

$h_1+h_2+h_3+h_4+h_9 \leq 23$

$h_1+h_6+h_7+h_8+h_9 \leq 23$

$h_1+h_2+h_3+h_4+h_{10}+h_{16} \leq 19$

$h_1+h_6+h_7+h_8+h_{10}+h_{16} \leq 19$

$h_{12} \leq 22$

$h_{12}+h_{13} \leq 22$

$h_{12}+h_{13}+h_{14} \leq 20$

$h_{12}+h_{17} \leq 24$

$h_{12}+h_{17}+h_{18} \leq 23$

$h_{12}+h_{17}+h_{18}+h_{19} \leq 20$

$h_{12}+h_{13}+h_{14}+h_{15} \leq 23$

$h_{12}+h_{17}+h_{18}+h_{19}+h_{15} \leq 23$

$h_{12}+h_{17}+h_{18}+h_{20} \leq 22$

Non negativity constraints:

$Q_1, \ldots, Q_{21} \geq 0$

$h_1, \ldots, h_{21} \geq 0$

By using MS-Excel Solver, we can get the optimal solution as follows:

Table 4-16 **Optimum solution of Example 4.1 (using MS- Excel Solver)**

Link	*Discharge (m^3/day)*	*Headloss (m)*	*Diameter (m)*
1	7400	4.69996526	0.357
2	2201.29051	7.00758381	0.216
3	1201.29051	3.15474065	0.181
4	301.290511	1.52832015	0.122
5	411.052828	3.60939014	0.125
6	4598.70949	3.42465735	0.291
7	3798.70949	3.8530486	0.275
8	2998.70949	4.41293865	0.257
9	0	6.60939014	0
10	1688.94717	2.16997288	0.223
11	488.947172	1.43941725	0.157
12	6900	4.71268226	0.331
13	2281.15598	6.49599232	0.19
14	1481.15598	8.79132542	0.17
15	1200	3	0.175
16	0	0.43941725	0
17	4018.84402	4.80279687	0.273
18	3218.84402	2.80105462	0.258
19	918.84402	7.68346625	0.133
20	1700	4.13419406	0.21
21	800	2.54927219	0.168

Table 4.17 Head pressure at each node of network of Example 4.1

Node	*Pressure Head (m)*	
	Actual	*Required*
1	220	220
2	215.3	195
3	208.2925	196
4	205.1377	197
5	203.6094	198
6	200	200
7	211.8754	199
8	208.0223	198
9	197	197
10	201.4394	199
11	215.2873	198
12	208.7913	198
13	200	200
14	201	201
15	210.4845	196
16	207.6835	197
17	203.5493	198

The cost of water network is 934,412 units. The complete iteration process is presented in Appendix B

This solution gives us information that link 9 and 16 are redundant, so based on optimality point of view, these pipes can be deleted. But if reliability becomes our consideration, then for each node there should be minimum 2 links connected. In this case, link 9 and 16 should be kept available to supply node 9 and 14 respectively. If we give minimum diameter 100 mm for these two links, then the cost of network will becomes 1,159,348 units.

4.3.2 Solution by using Genetic Algorithm approach

The first step is to develop a model of the problem in Epanet Format. The model is written in Epanet format so that the problem can be read by Water network optimiser program and saved in input file (*.inp).

The project file of this problem also developed. The information related to diameter-cost of pipe, the model itself, pipes that need to be modified, required pressure head on every node are given in project file.

The process of finding optimum solution is trial process. The trial process is given as follows:

<u>First trial:</u>

The parameters are as follows:

Violation Penalty	20000
Population size :	100
Iterations:	10000
Crossover rate:	0.9
Mutation rate:	0.1
GA Type:	Generational

The results for this trial are given in Table 4-18.

Table 4-18 Results of Example 4.1 (First trial)

Link	Diameter (mm)	*Node*	*Pressure Actual*	*Head (m) Required*
1	350	1	220.06	220
2	150	2	215.61	195
3	25	3	205.84	196
4	150	4	197.51	197
5	200	5	201.58	198
6	300	6	200.04	200

7	300	7	211.93	199
8	250	8	208.63	198
9	150	9	203.41	197
10	200	10	199.66	199
11	25	11	215.91	198
12	350	12	211.54	198
13	250	13	206.62	200
14	250	14	203.19	201
15	200	15	211.31	196
16	25	16	209.54	197
17	250	17	204.31	198
18	250			
19	25			
20	200			
21	200			

The cost of water network is 625,850 units.

Second trial:

The parameters are as follows:

Violation Penalty	20000
Population size:	100
Iterations:	10000
Mutation rate:	0.05
GA Type:	Generational

The results for this trial are given in Table 4-19.

Table 4-19 Results of Example 4.1 (Second trial)

Link	Diameter (mm)	*Node*	*Pressure Actual*	*Head (m) Required*
1	350	1	220.06	220
2	200	2	213.78	195
3	150	3	206.08	196
4	25	4	201.50	197
5	200	5	203.78	198
6	300	6	202.24	200
7	350	7	209.40	199
8	300	8	207.49	198
9	25	9	199.62	197
10	250	10	202.09	199
11	25	11	214.06	198
12	300	12	210.86	198
13	250	13	201.15	200
14	200	14	201.07	201
15	200	15	206.25	196
16	200	16	205.45	197
17	200	17	199.02	198
18	250			
19	25			
20	150			
21	25			

The cost of water network is 625,450 units which is the optimal one.

If reliability issue becomes our concern, then minimum diameter of pipe should be kept available. If we apply the minimum diameter 100 mm, then the cost of network will increase to 671,350 units.

CHAPTER V

CONCLUSIONS AND SCOPE FOR FUTURE STUDY

5.1 CONCLUSIONS

1. Water distribution network design problem, basically is a highly non linear programming problem in nature, and it can be solved using Non Linear Programming approach or Stochastic Search approach. There are various methods available in Non Linear Programming approach including Generalised Reduce Gradient (GRG) algorithm which is capable to solve the real world type optimisation problems. Genetic Algorithms, a method in Stochastic Search approach, is proved to be a better algorithm for solving water distribution network design problem which is highly non linear type.
2. In Non Linear Programming approach, water distribution network problem can be formulated in 3 types, i.e. D-Q (diameter-discharge), D-h (diameter-headloss), and Q-h (discharge- headloss) formulation. Problem formulation in Q-h is better than D-Q or D-h, because the constraints in Q-h formulation are linearly in type and this condition will bring optimisation process faster than the other two formulations.
3. Microsoft Excel Solver, which is a software that using Generalised Reduced Gradient methods in solving Non Linear Programming problem, is using iterative methods in the optimisation process. In its process, derivatives and gradients play a crucial role on how to variables should be adjusted. Microsoft Excel Solver approximates the derivatives numerically by moving each adjustable cell value slightly and observing the rate of change of each constraint cell and the optimum cell. Because the first derivative (or gradient) of the optimum cell measures its rate of change with respect to (each of) the adjustable cells, when all the partial derivatives of optimum cell are zero, then the optimum solution has been achieved.
4. In solving water distribution network design problem, Microsoft Excel Solver has performed well. However, the solution that is achieved is locally optimal. When started at a given set of adjustable cell values, the methods used by

Microsoft Excel Solver will tend to converge on optimal point close to the starting point. But Microsoft Excel Solver has no sure way of knowing global optimum. For searching global optimum solution, we can start Microsoft Excel Solver from several different, widely separated points and see which solution is best.

5. The solution that is achieved by using NLP approach is continuous value and it need to be round up to available market size of diameter. The process of rounding up the diameter is causing the cost of network is increased and the solution is not optimum.
6. The using of Genetic Algorithm (GA) methods in solving water distribution network design problem giving chances in getting global optimum solution, because Genetic Algorithm simultaneously considers a population of solutions, spread throughout the solution space, so the probability of reaching global optimum solution is increased. By implementing the principle of genetic process, i.e. survival the fittest, the initial population of solutions is randomly generated and the cost of network is computed, also the information about their feasibility. This algorithm tends to search the solutions that give less cost of network and if the solutions are not hydraulically feasible according to hydraulic requirement, the penalty will be applied and the consequences the cost of network will be increased. To know whether solution hydraulically feasible or not, the hydraulic simulation package is incorporated, for example Epanet 2.0 is used by Water Network Optimiser. Water Network Optimiser which is an example of GA application software in water distribution network design performs a good result and gives a better solution than MS- Excel Solver.
7. GA can deal with discrete variable, and in this case it becomes an advantage. Because of discrete variables, such as market size diameter of pipe, can be handled, there are no need of rounding up process as in NLP approach and cost of network is not get increase. This advantage makes GA approach is superior to NLP approach.

5.2 SCOPE FOR FUTURE STUDY

There are large scope for future study in are of looped water distribution network design. The things that can be considered are:

1. Multiple loadings and multiple sources

The design of water distribution network should consider also multiple loading and multiple sources in the network. This dissertation only considers a network with one source and one demand pattern.

2. Reliability point of view

The further study of reliability issue in water distribution network design also can be done, by calculating incremental of reliability could be achieved upon incremental cost of network.

3. Spatial based design of water distribution network

The impact of land use and development issues to installation cost should also be considered. The spatially distributed soil characteristic will impact excavation cost, and this should also be considered.

4. Modification and Expansion of water distribution network

The dissertation is considered for new network. We should also consider dealing with existing network that is needed to be modified or to be expanded.

REFERENCES

1. Alperovits, E and Shamir, U; 'Design of Optimal Water Distribution Systems'; Water Resources Research, Vol. 13, No.6, December 1977
2. Bao, Y. and Mays, L.W.: 'Model for water distribution system reliability'; Journal of Hydraulic Engineering, ASCE, Vol. 116, No. 9, 1990.
3. Bhave, P.R. ; 'Optimal Design of Water Distribution Networks'; Narosa, New Delhi, 2003
4. Bhave, P.R., and Sonak, V.V.; 'A critical study of the linear programming gradient method for optimal design of water supply networks'; Water Resources Research, Vol. 28, No. 6, 1992.
5. Bronson, R.; 'Operation Research'; Schaum Outline Series-Mc Graw Hill, Singapore, 1983
6. Chiplunkar, A., Mehndiratta, S. and Khanna, P.; 'Looped water distribution system optimization for single loading', J. Environmental Eng., ASCE, Vol. 112, No. 2, 1986.
7. Cunha, M.C. and Sousa, J ;'Water Distribution Network Design Optimisation: Simulated Annealing Approach'; Journal of Water Resources Planning and Management, ASCE, Vol. 125, No.4, July/August 1999
8. Eiger, G., Shamir, U., and Ben-Tal, A.: 'Optimal design of water distribution network'; Water Resource Research, Vol. 30, No. 9, 1994.
9. Fylstra, D., Lasdon, L., Watson, J. and Warren, A. ;' Design and Use of the Microsoft Excel Solver'; INTERFACES, September-October 1998
10. Garg, S.K. ;'Water Supply Engineering'; Khanna, New Delhi, 1977
11. Gessler, J.: 'Optimization of pipe networks'; Proc. of the Ninth International. Symposium on Urban Hydrology, Hydraulics and Sediment Control, Univ. of Kentucky., Lexington, USA, July 27-30, 1982.
12. Halhal, D., Walters, G. A., Savic, D. A., and Ouazar, D.; 'Scheduling of Water Distribution System Rehabilitation using Structured Messy Genetic Algorithms', Evolutionary Computation, Vol. 7, No.3, 1999

13. Haupt, R.L. and Haupt, S.E. ;'Practical Genetic Algorithms'; John Wiley and Sons Inc, Canada, 1998
14. Jacoby, S.L.S. ;'Design of Optimal Hydraulic Networks'; Journal of Hydraulic Division, ASCE, Vol. 94, No. 3, May 1968
15. Jeppson R.W.; 'Analysis of Flow in Pipe Networks'; Ann Arbor Science, Michigan, USA, 1977
16. Keedwell, E.C.; 'Water Network Optimiser-Help'; Centre for Water Systems, University of Exeter, U.K.
17. Kiswarman; 'Reliability Based Design of Pumping Station for an Urban Water Supply Scheme'; M.Tech Dissertation, WRDTC, IIT Roorkee, India, 2004
18. Kumari, I.J.; 'Optimal Design of Water Distribution Networks', M.E. Dissertation, Dept. of Civil Engineering, Delhi College of Engineering, India, 1998
19. Lansey, K., Duan, N., Mays, L.W., and Tung, Y.: 'Water distribution system design under uncertainties'; Journal of Water Resources Planning and Management, ASCE, Vol. 115, No. 5, 1989.
20. Lansey, K.E. and Mays, L.W. ;'Optimisation Model for Water Distribution System Design'; Journal of Hydraulic Engineering, ASCE, Vol. 115, No. 10, October 1989
21. Morgan, D.R., and Goulter, I.C.: 'Optimal Urban Water Distribution Design'; Water Resources Research, Vol. 21, No. 5, 1985
22. Rao, S.S. ; 'Engineering Optimisation, Theory and Practice'; New Age International Limited, 2002
23. Rossman, L.A. :'Epanet 2, Users Manual'; US. Environmental Protection Agency, Cincinnati, USA, 2000.
24. Savic, D.A. and Walters, G.A. ;'Genetic Algorithms for Least-Cost Design of Water Distribution Network'; Journal of Water Resources Planning and Management, ASCE, Vol. 123, No. 2, March-April 1997
25. Simpson, A.P, Dandy, G.C. and Murphy, L.J.; 'Genetic Algorithms Compared to Other Techniques for Pipe Optimisation'; Journal of Water

25. Simpson, A.P, Dandy, G.C. and Murphy, L.J.; 'Genetic Algorithms Compared to Other Techniques for Pipe Optimisation'; Journal of Water Resources Planning and Management, ASCE, Vol. 120, No. 4, July/August 1994.
26. Taher, S. and J. Labadie: 'Optimal Design of Water Distribution Networks with GIS'; Journal of Water Resources Planning and Management, ASCE, Vol. 122, No. 4 , July 1996
27. Varma, K.V.K., Narasimhan, S. and Bhallamudi, S.M. ;'Optimal Design of Water Distribution Systems using an NLP methods'; Journal of Environmental Engineering, ASCE, Vol. 123, No. 4, April 1997
28. Walski, T.M. et al. ; 'Advanced Water Distribution Modeling and Management', First Edition, Haestead Press, 2003
29. Winston, W.L. ;'Microsoft Excel Data Analysis and Business Modelling'; Microsoft Press, 2003
30. Wu, Z. Y, Walski, T. M., Mankowski, R., Tryby, M., Herrin, G., and Hartell, W.: 'Optimal Capacity of Water Distribution Systems'; Proceedings of the 1st Annual Enviromental and Water Resources Systems Analysis (EWRSA) Symposium, Roanoke, Virginia, 2002

Appendix A

1. D-Q formulation

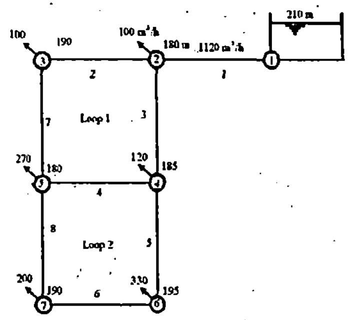
Changing cells		Initial values									
			1st running		2nd running		3rd running		4th running		5th running
d1	0.499555	0.5	0.5	0.4	0.418153212	0.42	0.44801623	0.42	0.4851939	0.42	0.499554975
d2	0.2993802	0.4	0.38322803	0.2	0.234761497	0.23	0.25688143	0.23	0.3034944	0.24	0.299380162
d3	0.4047766	0.4	0.45460242	0.35	0.363531496	0.36	0.38632903	0.36	0.3933749	0.36	0.404776644
d4	0	0.1	0.10854545	0.07	0.068986648	0.07	0.06322111	0.06	0.0350655	0.06	0
d5	0.4347417	0.4	0.44034893	0.25	0.264465428	0.28	0.29477405	0.3	0.4158683	0.3	0.434741709
d6	0.3430467	0.3	0.34146755	0.2	0.213306484	0.3	0.3115242	0.32	0.3359914	0.32	0.343046716
d7	0.2847346	0.3	0.2435461	0.1	0.189800738	0.19	0.21357195	0.21	0.2768684	0.22	0.28473458
d8	0.0707654	0.1	0.3024006	0.1	0.104164367	0.1	0.10729313	0.06	0.0673952	0.06	0.070765361
q2	367.99882	300	300.000051	340	339.9999818	350	349.99997	368	367.99945	368	367.9988166
q3	652.00118	720	719.999949	680	680.0000182	670	670.00003	652	652.00055	652	652.0011834
q4	1.0026373	20	20.0001383	20	20.0000515	10	10.0000821	1	1.0012818	1	1.00263728
q5	530.99855	580	579.999811	540	539.9999667	540	539.999948	531	530.99927	531	530.9985462
q6	200.99855	250	249.999811	210	209.9999667	210	209.999948	201	200.99927	201	200.9985462
q7	267.99882	200	200.000051	240	239.9999818	250	249.99997	268	267.99945	268	267.9988166
q8	0.9985462	50	49.999811	10	9.999966676	10	9.99994784	1	0.9992667	1	0.998546156
q2+q3=1020	1020		1020		1020		1020		1020		1020
q3-q5-q4=120	120		120		120		120		120		120
q2-q7=100	100		100		100		100		100		100
q4+q8+q7=270	270		270		270		270		270		270
q5-q6=330	330		330		330		330		330		330
q6-q8=200	200		200		200		200		200		200
h1	4.3968292		4.37780382		10.45530329		7.47204833		5.0679544		4.396829249
h2	6.7737617		1.39405614		19.11764957		13.0108028		6.3381681		6.773761674
h3	4.4971485		3.07044665		8.204558496		5.93603933		5.1685305		4.497148477
h4	#DIV/0!		4.30698013		39.15726179		16.5919127		4.1262351		#DIV/0!
h5	2.1715749		2.40251281		25.20838663		14.8617163		2.6955496		2.171574934
h6	1.1387577		1.74445983		12.49075488		1.97484642		1.2600468		1.138757672
h7	4.8069176		5.98337015		28.24417163		17.1471866		5.5096113		4.806917632
h8	0.1343666		0.16000749		1.458120282		1.26240617		0.1706384		0.134366593
h2+h7-h3-h4=0	#DIV/0!		-4.8584E-07		9.07474E-07		7.63003736		2.5530137		#DIV/0!
h4-h5-h6-h8=0	#DIV/0!		7.237E-10		1.2057E-13		-1.5070562		2.542E-07		#DIV/0!
h1<=	4.3968292		4.37780382		10.45530329		7.47204833		5.0679544		4.396829249
h1+h2<=	11.170591		5.77185996		29.57295286		20.4828512		11.406122		11.17059092
h1+h3<=	8.8939777		7.44825046		18.65986179		13.4080877		10.236485		8.893977726
h1+h2+h7<=	15.977509		11.7552301		57.81712448		37.6300378		16.915734		15.97750855
h1+h3+h4<=	#DIV/0!		11.7552306		57.81712358		30.0000004		14.36272		#DIV/0!
h1+h3+h5<=	11.065553		9.85076328		43.86824842		28.269804		12.832035		11.06555266
h1+h3+h5+h6<=	12.20431		11.5952231		56.3590033		30.2446504		14.192081		12.20431033

Target cell

f(x)	20,500,330	24,436,593	14,659,743	17,066,752	19,953,123	20,500,330

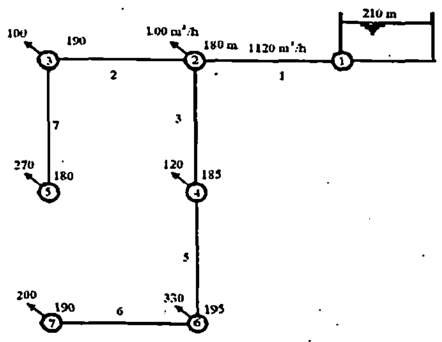
Constraints	
q2+q3=1020	1020
q3-q5-q4=120	120
q2-q7=100	100
q4+q8+q7=270	270
q5-q6=330	330
q6-q8=200	200
h2+h7-h3-h4=0	0
h4-h5-h6-h8=0	0
h1<=	30
h1+h2<=	20
h1+h3<=	25
h1+h2+h7<=	30
h1+h3+h4<=	30
h1+h3+h5<=	15
h1+h3+h5+h6<=	20

q1,..,q4>=0
h1,...,h4 >=0



Changing cells	Initial values	6th running		7th running		8th running		9th running		10th running	
d1	0.480199195	0.42	0.47893276	0.42	0.48019289	0.42	0.4801911	0.42	0.4802169	0.42	0.4801992
d2	0.261991412	0.24	0.258228023	0.25	0.26198128	0.25	0.2619745	0.25	0.2619783	0.26	0.2619914
d3	0.395109475	0.36	0.395682616	0.41	0.39510644	0.41	0.3950946	0.36	0.395096	0.41	0.3951095
d4	0	0	0	0	0	0	0	0	0	0	0
d5	0.371718411	0.3	0.372273622	0.32	0.37172712	0.38	0.3717408	0.3	0.3717165	0.38	0.3717184
d6	0.252692647	0.32	0.252692647	0.3	0.25269265	0.26	0.2526926	0.3	0.2526926	0.26	0.2526926
d7	0.238426255	0.22	0.24566527	0.3	0.23843815	0.26	0.2384455	0.3	0.2384357	0.26	0.2384263
d8	0	0	0	0	0	0	0	0	0	0	0
q2	370	370	370	370	370	370	370	370	370	370	370
q3	650	650	650	650	650	650	650	650	650	650	650
q4	0	0	0	0	0	0	0	0	0	0	0
q5	530	530	530	530	530	530	530	530	530	530	530
q6	200	200	200	200	200	200	200	200	200	200	200
q7	270	270	270	270	270	270	270	270	270	270	270
q8	0	0	0	0	0	0	0	0	0	0	0
q2+q3=1020	1020		1020		1020		1020		1020		1020
q3-q5-q4=120	120		120		120		120		120		120
q2-q7=100	100		100		100		100		100		100
q4+q8+q7=270	270		270		270		270		270		270
q5-q6=330	330		330		330		330		330		330
q6-q8=200	200		200		200		200		200		200
h1	5.329885302		5.398873897		5.33022639		5.330323		5.3289298		5.3298853
h2	13.10207473		14.60112652		13.104543		13.106186		13.105259		13.102075
h3	5.03025524		4.994870486		5.03044361		5.0311758		5.0310926		5.0302552
h4											
h5	4.63985938		4.606256523		4.63933001		4.6385012		4.6399776		4.6398594
h6	5		5		5		5		5		5
h7	11.56803988		10		11.5652306		11.563491		11.565812		11.56804
h8											
h1<=	5.329885302		5.398873897		5.33022639	30	5.330323		5.3289298		5.3298853
h1+h2<=	18.43196004		20.00000042		18.4347694	20	18.436509		18.434189		18.43196
h1+h3<=	10.36014054		10.39374438		10.36067	25	10.361499		10.360022		10.360141
h1+h2+h7<=	29.99999992		30.00000042		30	30	30		30.000001		30
h1+h3+h4<=	10.36014054		10.39374438		10.36067	30	10.361499		10.360022		10.360141
h1+h3+h5<=	14.99999992		15.00000091		15	15	15		15		15
h1+h3+h5+h6<=	19.99999992		20.00000091		20	20	20		20		20
Target cell											
f(x)	17,172,733		17,183,587		17,172,733		17,172,733		17,172,733		17,172,733

Constraints	
q2+q3=1020	1020
q3-q5-q4=120	120
q2-q7=100	100
q4+q8+q7=270	270
q5-q6=330	330
q6-q8=200	200
h1<=	30
h1+h2<=	20
h1+h3<=	25
h1+h2+h7<=	30
h1+h3+h4<=	30
h1+h3+h5<=	15
h1+h3+h5+h6<=	20

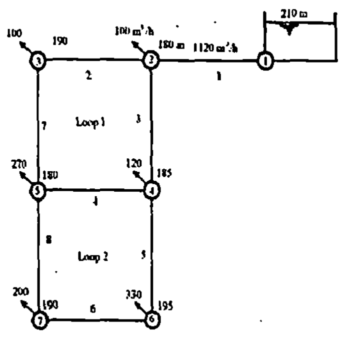


q1,..,q4>=0
h1,...,h4 >=0

2. D-h formulation

Changing cells		Initial values	1st running		2nd running		3rd running
d1	0.499541666	0.42	0.44107101	0.42	0.488827439	0.48	0.488848685
d2	0.297563101	0.3	0.30267446	0.24	0.301907955	0.24	0.283789821
d3	0.381376347	0.36	0.37791807	0.41	0.378436243	0.39	0.39
d4	0.104324057	0.06	0.05832942	0.06	0.068581014	0.06	0.065869544
d5	0.385424822	0.4	0.40887145	0.3	0.39192447	0.35	0.394167457
d6	0.335681071	0.3	0.29861372	0.3	0.298627583	0.25	0.268829553
d7	0.269777606	0.3	0.27623569	0.22	0.275282928	0.22	0.25195509
d8	0.099772511	0.1	0.0995607	0.1	0.100042379	0.1	0.1
h1	4.500082408	5	5.00037002	5	5.00123288	5	5.00017443
h2	6.401708037	6.4	6.40078725	6.4	6.402425084	6.4	6.401878125
h3	6.499867607	6.5	6.50022771	6.5	6.500159591	6.5	6.500160155
h4	5.40144262	5.4	5.40020322	5.4	5.402587873	5.4	5.40182693
h5	4.000049985	3	3.00014178	3	3.002255676	3	3.001058012
h6	1.301121853	2.3	2.30005738	2.3	2.300143414	2.3	2.30059495
h7	5.499602191	5.5	5.49964367	5.5	5.500322381	5.5	5.500108961
h8	0.100270782	0.1	0.10000406	0.1	0.100188783	0.1	0.100173968
q1	1120		855.031711		1120		1120
q2	347.6447909		363.511277		361.1495999		306.9834388
q3	672.3552091		656.488723		658.8504001		713.0165613
q4	20.25401087		4.40221168		6.735183463		6.058005466
q5	532.1011982		532.086512		476.3096172		483.3943156
q6	202.1011982		202.086512		202.1152166		153.3943156
q7	247.6447909		263.511276		261.1495991		206.9834388
q8	2.101198194		2.08651166		2.115216607		2.112696974
q1	1120		855.031711		1120		1120
q2+q3=1020	1020	1020	1020		1020		1020
q3-q5-q4=120	120	120	120		175.8055995		223.5642402
q2-q7=100	100	100	100		100.0000009		99.99999999
q4+q8+q7=270	270	270	270		269.9999991		215.1541412
q5-q6=330	330	330	330		274.1944006		330
q6-q8=200	200	200	200		200		151.2816186
h2+h7-h3-h4=0	0	0	0		0		0
h4-h5-h6-h8=0	-1.80411E-16	0	-4.8572E-16		-9.57567E-16		-3.19189E-16
h1<=	4.500082408	30	5.00037002		5.00123288		5.00017443
h1+h2<=	10.90179044	20	11.4011573		11.40365796		11.40205255
h1+h3<=	10.99995002	25	11.5005977		11.50139247		11.50033459
h1+h2+h7<=	16.40139264	30	16.9008009		16.90398034		16.90216152
h1+h3+h4<=	16.40139264	30	16.9008009		16.90398034		16.90216152
h1+h3+h5<=	15	15	14.5007395		14.50364815		14.5013926
h1+h3+h5+h6<=	16.30112185	20	16.8007969		16.80379156		16.80198755
Target cell							
f(x)	20,181,642		19,073,578		19,527,474		18,917,371

Constraints	
q1	1120
q2+q3=1020	1020
q3-q5-q4=120	120
q2-q7=100	100
q4+q8+q7=270	270
q5-q6=330	330
q6-q8=200	200
h2+h7-h3-h4=0	0
h4-h5-h6-h8=0	0
h1<=	30
h1+h2<=	20
h1+h3<=	25
h1+h2+h7<=	30
h1+h3+h4<=	30
h1+h3+h5<=	15
h1+h3+h5+h6<=	20
q1,..,q4>=0	
h1,...,h4 >=0	



4th running		5th running		6th running		7th running		8th running	
0.48	0.49954681	0.48	0.499545879	0.48	0.49954183	0.48	0.49954167	0.49954167	0.499541666
0.24	0.28994206	0.24	0.296817724	0.26	0.29467416	0.2635	0.29756297	0.29756297	0.297563101
0.39	0.38626622	0.39	0.381861513	0.39	0.38326793	0.381	0.38115142	0.38115142	0.381376347
0.06	0.06444615	0.06	0.064661861	0.1	0.10453755	0.1	0.10432406	0.10432406	0.104324057
0.37	0.38543319	0.39	0.39	0.37	0.38542512	0.37	0.38513757	0.38513757	0.385424822
0.32	0.3356996	0.32	0.33570412	0.32	0.33568163	0.32	0.33502357	0.33502357	0.335681071
0.26	0.26	0.28	0.275663696	0.28	0.26973182	0.28	0.26977708	0.26977708	0.269777606
0.1	0.10041035	0.1	0.100612006	0.1	0.09979559	0.1	0.09979559	0.09979559	0.099772511
4.5	4.49985671	4.5	4.499897587	4.5	4.500075	4.5	4.50008241	4.50008241	4.500082408
6.4	6.40229687	6.4	6.402524392	6.4	6.40173344	6.4	6.40167799	6.40167799	6.401708037
6.5	6.49998355	6.5	6.500045739	6.5	6.49987087	6.5	6.49986761	6.49986761	6.499867607
5.4	5.40189792	5.4	5.401961352	5.4	5.4014615	5.4	5.40141257	5.40141257	5.40144262
4	4.00015974	4	4.000056674	4	4.00005414	4	4.00004998	4.00004998	4.000049985
1.3	1.30122826	1.3	1.30129193	1.3	1.30112772	1.3	1.30108709	1.30108709	1.301121853
5.5	5.49958459	5.5	5.499482699	5.5	5.49959894	5.5	5.49960219	5.49960219	5.499602191
0.1	0.10050992	0.1	0.100612748	0.1	0.10027964	0.1	0.1002755	0.1002755	0.100270782
	1120		1120		1120		1120		1120
	324.772994		345.3873134		338.855667		347.643522		347.6447909
	695.227006		674.6126859		681.144333		671.314814		672.3552091
	5.72042454		5.770857547		20.3630296		20.2539501		20.25401087
	532.139387		548.8418284		532.102575		531.060864		532.1011982
	202.139387		202.1518691		202.102575		201.060864		202.1011982
	224.772994		262.0772733		247.534396		247.643522		247.6447909
	2.13938675		2.151869149		2.1025746		2.10252773		2.101198194
	1120		1120		1120		1120		1120
	1020		1019.999999		1020		1018.95834		1020
	157.367195		120		128.678729		120		120
	100		83.31004012		91.321271		100		100
	232.632805		270		270		270		270
	330		346.6899592		330		330		330
	200		200		200		198.958336		200
	0		0		0		0		0
	6.5226E-16		0		-7.7716E-16		0		-1.80411E-16
	4.49985671		4.499897587		4.500075		4.50008241		4.500082408
	10.9021536		10.90242198		10.9018084		10.9017604		10.90179044
	10.9998403		10.99994333		10.9999459		10.99995		10.99995002
	16.4017382		16.40190468		16.4014074		16.4013626		16.40139264
	16.4017382		16.40190468		16.4014074		16.4013626		16.40139264
	15		15		15		15		15
	16.3012283		16.30129193		16.3011277		16.3010871		16.30112185
	19,775,486		20,018,185		20,174,000		20,168,399		$\boxed{20,181,642}$

3. Q-h formulation

Changing cells		Initial values							
		1st running		2nd running		3rd running		4th running	
q2	369.760183	300	336.637381	370	369.5	370	369.5	370	369.83541
q3	650.239817	720	683.362619	650	650.5	650	650.5	650	650.16459
q4	0.23981637	20	-1.27411101	1	-0.975	1	-0.475	1	-0.052966
q5	530	580	564.63673	530	531.475	530	530.975	530	530.21756
q6	200	250	234.63673	200	201.475	200	200.975	200	200.21756
q7	269.760183	200	236.637381	270	269.5	270	269.5	270	269.83541
q8	0	50	34.6367302	30	28.025	20	18.525	1	0.21756
h1	6.12176293	10	5.19581498	6	6	6	6	6	5.8864285
h2	13.7044433	10	14.804185	13	13	13	13	13	13.130327
h3	6.11654102	5	6.08044071	6	6	6	6	6	5.9721066
h4	17.6000812	15	18.7237443	17	17	17	17	17	17.001999
h5	2.76169605	1	3.72374431	2.5	2.5	2.5	2.5	2.5	3.1414649
h6	4.99181108	5	5	5	5	5	5	5	5
h7	10.0056569	10	10	10	10	10	10	10	10.154895
h8	9.9	10	37.2291272	10	10	10	10	10	10.004421
q2+q3=1020	1020		1020		1020		1020		1020
q3-q5-q4=120	120.000001		120		120		120		120
q2-q7=100	100		100		100		100		100
q4+q8+q7=270	269.999999		270		296.55		287.55		270
q5-q6=330	330		330		330		330		330
q6-q8=200	200		200		173.45		182.45		200
h2+h7-h3-h4=0	0		0		0		0		0.311117
h4-h5-h6-h8=0	0		-27.2291272		-0.5		-0.5		-1.143888
h1<=	6.12176293		5.19581498		6		6		5.8864285
h1+h2<=	19.8262062		20		19		19		19.016755
h1+h3<=	12.238304		11.2762557		12		12		11.858535
h1+h2+h7<=	29.8318631		30		29		29		29.171651
h1+h3+h4<=	29.8383852		30		29		29		28.860534
h1+h3+h5<=	15		15		14.5		14.5		15
h1+h3+h5+h6<=	19.9918111		20		19.5		19.5		20
Target cell									
f(x)	17,405,292		#NUM!		#NUM!		#NUM!		#NUM!

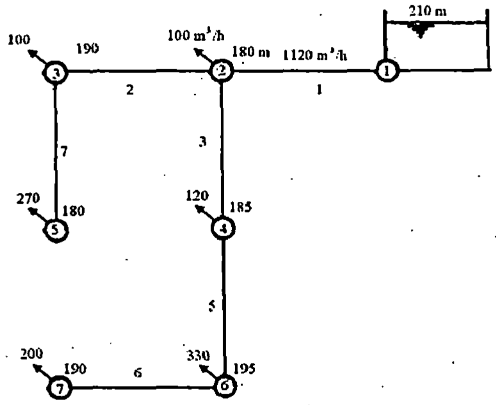
Constraints	
q2+q3=1020	1020
q3-q5-q4=120	120
q2-q7=100	100
q4+q8+q7=270	270
q5-q6=330	330
q6-q8=200	200
h2+h7-h3-h4=0	0
h4-h5-h6-h8=0	0
h1<=	30
h1+h2<=	20
h1+h3<=	25
h1+h2+h7<=	30
h1+h3+h4<=	30
h1+h3+h5<=	15
h1+h3+h5+h6<=	20

q1,..,q4>=0
h1,...,h4 >=0

5th running		6th running	
370	369.7591562	370	370
650	650.2408438	650	650
0.5	0.240842815	0	0
530	530	530	530
200	200	200	200
270	269.7591562	270	270
0	0	0	0
6.1	6.119569268	6.1	5.3301495
13.7	13.77809689	13.7	13.104698
6.1	6.118869123	6.1	5.0304443
17.6	17.66031904	0	0
2.72	2.761561609	2.72	4.6394062
4.98	4.99468899	4.98	5
10	10.00109128	10	11.565153
9.9	9.904068445	0	0

1020	1020
120.000001	120
100	100
269.999999	270
330	330
200	200
0	
0	
6.119569268	5.3301495
19.89766616	18.434847
12.23843839	10.360594
29.89875744	30
29.89875744	10.360594
15	15
19.99468899	20

	$\boxed{17,402,454}$	$\boxed{17,168,950}$
d1	0.466765303	0.4801923
d2	0.259232611	0.2619786
d3	0.379582103	0.3951043
d4	0.015131386	0
d5	0.413508999	0.3717237
d6	0.252745781	0.2526906
d7	0.245574409	0.2384365
d8	0	0



Appendix B

```
[InputFile]
C:\Program Files\trial4.INP

[Costs]
25.4    2
50.8    5
76.2    8
101.6   11
152.4   16
203.2   23
254     32
304.8   50
355.6   60
406.4   90
457.2   130
508     170
558.8   300
609.6   550

[RequiredHead]
1       210
2       180
3       190
4       185
5       180
6       195
7       190

[ModifyLinks]
1
2
3
4
5
6
7
8

[SolutionSets]
GA1
GA3
GA7
```

Appendix C

Changing cells			Initial values									
				1st running	2nd running		3rd running		4th running		5th running	
q1	8195	8195	8000	7999.999	8000	7999.9894	8500	8499.98777	8200	8200	8200	7749.9998
q2	1905	1905	3000	2999.9988	3000	2999.9875	3000	2999.98153	1950	1950	1905	2499.9998
q3	905	905	2000	1999.9988	2000	1999.9875	2000	1999.98153	950	949.9996	905	1499.9998
q4	5	5	1100	1099.9988	1100	1099.9875	1100	1099.98153	50	49.99962	5	599.99977
q5	895	895	1500	1499.9994	1500	1499.9933	2400	2399.99086	950	949.9997	905	1849.9999
q6	5690	5690	4400	4400.0002	4400	4400.0019	4900	4900.00624	5650	5650	5695	4650
q7	4890	4890	3600	3600.0002	3600	3600.0019	4100	4100.00624	4850	4850	4895	3850
q8	4090	4090	2800	2800.0002	2800	2800.0019	3300	3300.00624	4050	4050	4095	3050
q9	5	5	200	199.9994	200	199.99381	400	399.98891	50	49.99979	5	199.99994
q10	1995	1995	1000	1000.0002	1000	1000.0023	400	400.008007	1900	1900	1990	399.99995
q11	5	5	600	599.99938	600	599.99329	1500	1499.99086	50	49.99971	5	949.9999
q12	6105	6105	6300	6300.001	6300	6300.0106	5800	5800.01223	6100	6100	6100	6550.0002
q13	3190	3190	2000	2000.0007	2000	2000.0071	1800	1800.01292	3100	3100	3190	1800.0001
q14	2390	2390	1200	1200.0007	1200	1200.0071	1000	1000.01292	2300	2300	2390	1000.0001
q15	1195	1195	1000	1000.0006	1000	1000.0062	800	800.01109	1150	1150	1195	1000.0001
q16	790	790	400	399.9996	400	399.99561	700	699.998864	750	750	795	149.99986
q17	2315	2315	3700	3700.0003	3700	3700.0035	3400	3399.99931	2400	2400	2310	4150.0001
q18	1515	1515	2900	2900.0003	2900	2900.0035	2600	2599.99931	1600	1600	1510	3350.0001
q19	5	5	1000	999.99986	1000	999.99907	1000	999.99817	50	49.99991	5	1200
q20	910	910	1300	1300.0004	1300	1300.0044	1000	1000.00114	950	950	905	1550.0001
q21	10	10	400	400.0004	400	400.00439	100	100.001136	50	50.00004	5	650.00014
h1	3.47666667	3.74	2	10.79273	5	2.9884904	5	2.33703207	3.74	2.735687	3.74	2.6926758
h2	5.125	5	3	0.6395854	1.2	6.4344766	1.2	6.4542436	4	4.344285	4	4.3375905
h3	4.625	4.5	2	1.8741235	1.9	2.1709599	1.9	3.7249214	3.74	3.790723	3.74	3.7923946
h4	0.525	0.4	3	2.8005778	2.8	2.0813888	2.8	1.74632846	2.22	2.267409	0.4	2.272322
h5	3.02666633	4	2	0.7967892	1	2.5061127	1	-0.09804647	0.42	2.14328	4	2.3418197
h6	3.11166667	3.5	3	2.0546994	2.2	3.5006195	2.2	3.43834588	2.97	3.000759	2.97	3.0010026
h7	3.61166667	4	2	1.1670431	1.4	3.3086077	1.4	4.09907369	3.41	3.440632	3.41	3.4406926
h8	4.11166667	4.5	3	2.0925442	2.3	3.8775982	2.3	4.38807389	3.93	3.961026	3.93	3.960612
h9	1.47666667	0.7	2	-1.3780676	0.6	4.9150622	0.6	11.156313	6.63	8.302758	0.7	8.3352006
h10	2.24333317	3.3	3	2.1857164	2.4	3.8209671	2.4	4.44906522	4.27	5.4029	2	5.505486
h11	0.78333317	0.8	2	1.3889272	1.4	1.3148544	1.4	4.54711169	4.69	3.25962	0.7	3.1636663
h12	5.91666667	6.78	3	4.425393	4.4	5.5905003	4.4	8.8024737	6.78	5.103113	6.78	5.0888733
h13	4.62333333	5.4	2	5.0481135	5.2	6.3286771	5.2	8.00667182	8.74	7.886855	8.74	7.8954307
h14	4.16333333	4.74	3	3.4701032	3.5	3.8677925	3.5	4.48938524	4.74	4.746578	4.74	4.7426793
h15	3.65333333	4.43	2	1.7853397	1.9	2.803408	1.9	4.12030775	4.43	3.704316	4.43	3.7031992
h16	2.46	3.5	3	1.4648096	1.6	1.5037171	1.6	0.28840926	0.67	0.458997	4.3	0.399532
h17	4.77333333	4.46	2	3.1333334	3.1	3.5974198	3.1	4.27233454	4.46	4.460029	4.46	4.461529
h18	4.01333333	3.7	3	3.9502556	3.9	3.635263	3.9	3.19796076	3.7	3.696547	3.7	3.6985955
h19	0	0.2	2	1.4346276	1.6	2.9637869	1.6	5.02576176	5.32	4.476857	0.5	4.4779856
h20	3.48333333	2.97	3	4.6364941	4.6	3.7441214	4.6	2.3484911	2.97	3.812552	2.97	3.814844
h21	0.81333333	0.3	2	3.6120668	3.5	2.4326955	3.5	0.3787399	1.09	1.927761	0.4	1.9361582

Constraints:

Node flow continuity constraints						
q1+q12=14300	14300	14300	14300	14300	14300	14300
q1-q2-q6=600	600	600	600	600	600	600
q2-q3=1000	1000	1000	1000	1000	1000	1000
q3-q4=900	900	900	900	900	900	900
q4+q8-q5-q9-q10=1200	1200	1200	1200	1200	1200	1200
q5+q11=900	900	900	900	900	900	900
q6-q7=800	800	800	800	800	800	800
q7-q8=800	800	800	800	800	800	800
q9+q15=1200	1200	1200	1200	1200	1200	1200
q10-q11-q16=1200	1200	1200	1200	1200	1200	1200
q12-q13-q17=600	600	600	600	600	600	600
q13-q14=800	800	800	800	800	800	800
q14+q19-q15=1200	1200	1200	1200	1200	1200	1200
q16+q21=800	800	800	800	800	800	800
q17-q18=800	800	800	800	800	800	800
q18-q19-q20=600	600	600	600	600	600	600
q20-q21=900	900	900	900	900	900	900
Summation of Headloss equal to 0 's constraints						
h1+h6+h7+h8+h9-h15-h14-h13-h12=0	-2.56833333	0	0	0	1E-06	1E-06
h2+h3+h4-h8-h7-h6=0	-0.56	0	0	0	0	0
h5-h11-h10=0	0	0	0	0	0	0
h13+h14-h19-h18-h17=0	7.9936E-15	0	0	0	0	0
h19+h15-h9+h10+h16-h21-h20=0	2.58333317	0	0	0	0	0
Path Headloss constraints						
h1<=25	3.47666667	10.79273	2.9884904	2.33703207	2.735687	2.6926758
h1+h2<=24	8.60166667	11.432316	9.422967	8.79127567	7.079972	7.0302663
h1+h2+h3<=23	13.2266667	13.306439	11.593927	12.5161971	10.87069	10.822661
h1+h2+h3+h4<=22	13.7516667	16.107017	13.675316	14.2625255	13.1381	13.094983
h1+h6<=21	6.58833333	12.84743	6.4891099	5.77537794	5.736446	5.6936784
h1+h6+h7<=22	10.2	14.014473	9.7977176	9.87445163	9.177079	9.134371
h1+h6+h7+h8<=22	14.3116667	16.107017	13.675316	14.2625255	13.1381	13.094983
h1+h2+h3+h4+h5<=20	16.778333	16.903806	16.181429	14.164479	15.28138	15.436803
h1+h6+h7+h8+h5<=20	17.338333	16.903806	16.181429	14.164479	15.28138	15.436803
h1+h2+h3+h4+h10+h11<=20	16.778333	18.292733	17.496283	18.7115907	18.541	18.600469
h1+h6+h7+h8+h10+h11<=20	17.338333	18.292733	17.496283	18.7115907	18.541	18.600469
h1+h2+h3+h4+h10<=21	15.9949998	18.292733	17.496283	18.7115907	18.541	18.600469
h1+h6+h7+h8+h10<=21	16.5549998	18.292733	17.496283	18.7115907	18.541	18.600469
h1+h2+h3+h4+h9<=23	15.2283333	14.728949	18.590378	25.4188385	21.44086	21.430184
h1+h6+h7+h8+h9<=23	15.7883333	14.728949	18.590378	25.4188385	21.44086	21.430184
h1+h2+h3+h4+h5+h11+h16<=19	17.5616662	18.292733	17.496283	18.7115907	18.541	18.600469
h1+h6+h7+h8+h5+h11+h16<=19	20.5816662	19.757543	19	19	19	19.000001
h1+h2+h3+h4+h10+h16<=19	18.4549998	19.757543	19	19	19	19.000001
h1+h6+h7+h8+h10+h16<=19	19.0149998	19.757543	19	19	19	19.000001
h12<=22	5.91666667	4.425393	5.5905003	8.8024737	5.103113	5.0888733
h12+h13<=22	10.54	9.4735065	11.919177	16.8091455	12.98997	12.984304
h12+h13+h14<=20	14.7033333	12.94361	15.78697	21.2985308	17.73654	17.726983
h12+h17<=24	10.69	7.5587264	9.1879201	13.0748082	9.563141	9.5504023

h12+h17+h18<=23	14.7033333	11.508982	12.823183	16.272769	13.25969	13.248998
h12+h17+h18+h19<=20	14.7033333	12.94361	15.78697	21.2985308	17.73654	17.726983
h12+h13+h14+h15<=23	18.3566667	14.728949	18.590378	25.4188385	21.44086	21.430183
h12+h17+h18+h19+h15<=23	18.3566667	14.728949	18.590378	25.4188385	21.44086	21.430183
h12+h17+h18+h20<=22	18.1866667	16.145476	16.567304	18.6212601	17.07224	17.063842
h12+h17+h18+h20+h21<=19	19	19.757543	19	19	19	19
Target cell						
f(x)	#DIV/0!	#NUM!	$\boxed{1,041,203}$	#NUM!	$\boxed{969,336}$	$\boxed{1,032,580}$
d1			0.365476		0.375681	0.3689019
d2			0.2237642		0.205908	0.2263846
d3			0.2149791		0.144458	0.1718455
d4			0.1690517		0.051272	0.1318519
d5			0.1989956		0.172725	0.2185385
d6			0.2574036		0.292181	0.2713154
d7			0.2514192		0.27934	0.2558563
d8			0.2324057		0.266261	0.2390465
d9			0.074103		0.039276	0.0664873
d10			0.1470631		0.17483	0.096294
d11			0.1565045		0.050485	0.1556414
d12			0.2792839		0.281094	0.2889726
d13			0.1648484		0.186139	0.1513413
d14			0.1684751		0.206884	0.1507433
d15			0.1496995		0.149091	0.1413825
d16			0.1281837		0.20772	0.1158883
d17			0.254226		0.206329	0.2540845
d18			0.2127611		0.169114	0.2239627
d19			0.1516218		0.044587	0.1493033
d20			0.1748449		0.154609	0.186223
d21			0.1182213		0.056235	0.1490178