# Computer Aided Analysis of Flow and Pressure in Pipe Networks

By

Mohamed A-Salam Ali

Dublin City University

MEng 2000

# Computer Aided Analysis of Flow and Pressure in Pipe Networks

By

**Mohamed A-Salam Ali**

**This thesis is submitted to Dublin City University as the fulfilment of the requirement for the award of the degree of**

**Master of Engineering**

**Supervisor: Professor M..S.J.Hashmi**

**School of Mechanical and Manufacturing Engineering**
**Dublin City University**

**September, 2000**

# DECLARATION

**I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of *Master of Engineering* is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.**

**Signed: ------------------------------** **I.D:97970034**

**Mohamed A-Salam Ali**

**Date 8th September, 2000**

# ACKNOWLEDGMENT

I would like to thank to Professor M.S.J.Hashmi for his supervision and guidance during this research work.

I would also like to thank the School of Mechanical and Manufacturing Engineering of Dublin City University specially Mr. Brian Corcoran Liam Domican and Ms. Michelle Considine for their co-operation and assistance.

Finally, the patience and encouragement of my family deserve a greater acknowledgement than words can express.

# Dedication

**To my parents**

# Computer Aided Analysis of Flow and Pressure in Pipe Networks

by

**Mohamed A-Salam Ali.B.Sc (Eng)**

## Abstract

Pipeline systems range from the very simple to very large and quite complex ones. They may be as uncomplicated as a single pipe conveying water from one reservoir to another or they may be as elaborate as an interconnected set of water distribution networks for a major metropolitan area. Individual pipelines may contain several kinds of pumps at one and or at an intermediate point; they may deliver water to or from storage tanks. A system may consist of a number of sub-networks separated by differing energy lines or pressure levels that serve neighbourhoods at different elevation, and some of these may have pressurised tanks so that the pumps need not operate continuously. In order that these transfer systems will adequately fulfil their intended functions, they may require the inclusion of pressure reducing or pressure sustaining valves. These days an understanding of some particular numerical method and the ability to implement them on a computer, for obtaining solutions for a large problem is a vitally needed skill. Computations associated with engineering practice have changed dramatically in the past quarter century from the estimation of a few key values by using a slide rule to the generation of pages of computer output that are the result of detailed simulations of system performance. In the steady state analysis and design of networks, a large system of non-linear equation must be solved. In these work a computer program has been developed which carries out the regular simulation of steady state pressure and flow in pipe system. The computer program is written in C Language, to solve the basic pipe system equations using linear method. Basically the program reads input data defining the parameter for each pipe and each junction in the network. Connecting node number is the only geometric data in put for each pipe. From this data, the basic system equations are generated and solved.

# Contents

## *Chapter Three-Pipe Network Analysis*

***Chapter Four-Simulation***

## *Chapter Five-Conclusion and Suggestion*

# CHAPTER ONE

# INTRODUCTION AND JUSTIFICATION

## 1.1.Introduction

The term network analysis is usually taken to refer to the “analysis of flows and head losses in a pressurised distribution system under given demand conditions. Some methods for steady state hydraulic system can be used to solve for unknown pressure and flows in the networks. The solution of the steady state problem can be achieved when the flows in each pipe are determined under some specified patterns of supply and consumption.

## 1.2.A Computer as a Tool for Analysis and Design

The advent of the computers significantly enhanced our ability to analysis flow. Computer models for analysing pipe flows and pressures in water distribution networks are used through out the world is essential tools for the efficient operation and improvement of very complex systems. In presented here an overview of how computers can be used to support flow analysis is presented here. Most analysis and design problems do not have a single correct answer. The design may begin the solution process by developing a mathematical model of the physical system. Typically this model contains several parameters for which the engineer may select values. The range of possible parameter values is called the design space. The engineer searches the design space to find the best solution based on a set of performance criteria. This search can be a tedious process, even for a simple model with a few parameters. The simplest way in which computers can be used to support this process is to simply automate the calculation procedure. The engineer specifies a set of parameter values, and the computer calculates the corresponding system performance. Computer support becomes more desirable as the complexity of an analysis increases. The design process can be further automated if the computer model can be predict system performance for several sets of parameter values. In general, no matter how simple or complicated the design problem may be, a computer permits a more through search of the design space and therefore enhances the possibility of finding a better solution to the design problem.

Computer hardware provides the capability to perform the calculations required supporting flow analysis. Computer software is required to put this capability to work in particular problem. In general, some types of software are available to support fluid mechanics analysis. First, task-specific software permits the user to make an analysis of a selected class of problem by simply entering the physical characteristics of the problem. For example, pipe network program to analyse the flow through a system of connected pipes. The user simply inputs the physical characteristics of the system, and the program performs all the calculations required to predict system performance.

Second, the most general-purpose software to support flow analysis is programming language. The engineer must formulate the sequence of calculations to be performed and then instruct the computer to perform them by issuing a series of commands in the form of the program.

## 1.3.Aim of Study

This work gives detailed illustration on the development and use of a computer program, which permits simulation of steady pressure and flow in pipe distribution systems transporting liquid. The basis of the program is a direct solution of the basic pipe system hydraulics using a linearization method and solver linear algebra to handle the non-linear terms in the energy equation.

The computer program is written to analysis steady state flows and pressure for pipe distribution system. The program can be applied to other liquids, but does not generally apply to gas flow unless the assumption of constant density is acceptable. The program is written to accommodate any piping configuration and various hydraulic components such as pump, valve, and other components, which produce significant head loss. Computation carried out using English unit of CFS, GPM, MGD or standard international (SI) unit. The computer program, which provides network analysis and simulation, may be applicable to water distribution system as well. The program caters for hydraulic simulation and a description of the programs functionality's is given later.

## 1.4. Method of Approach

Description of the method that has been used in the simulation is depicted in the flow chart Fig. 1.1.

## 1.5.layout of Thesis

This thesis is divided into five chapters following this introductory chapter. Chapter two presents a review of elements of basic fluid mechanics that are pertinent to pipe system hydraulics. Chapter three, Concentrates on analysis techniques and completely describes the primary alternative approaches to the formulation of mathematical model for pipe system; then method for solving each of them is presented. Chapter four presented the new work and the technique, which has been used in the simulation with some example. Conclusion is presented in chapter five.

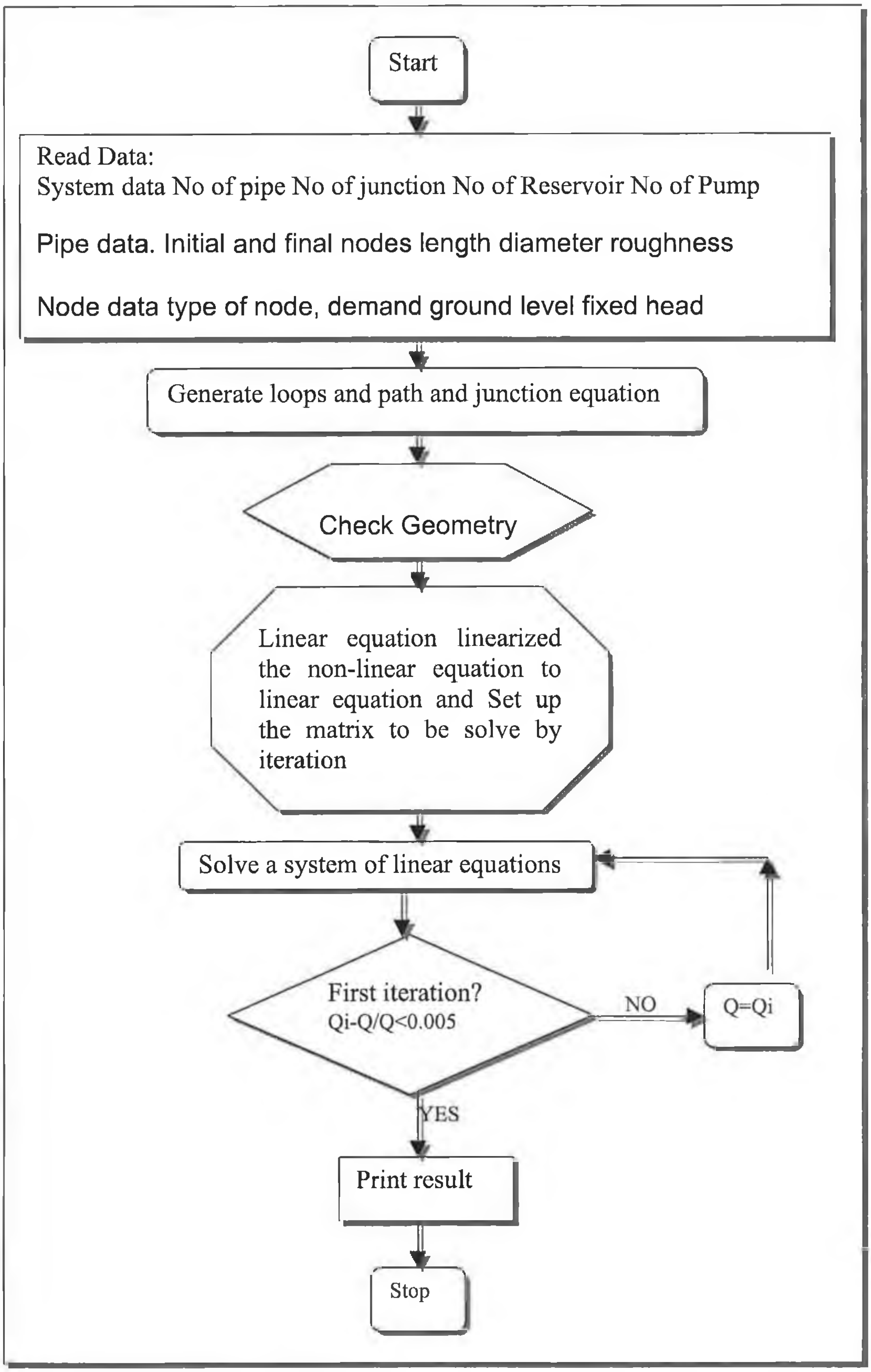

Start
Read Data:
System data No of pipe No of junction No of Reservoir No of Pump
Pipe data. Initial and final nodes length diameter roughness
Node data type of node, demand ground level fixed head
Generate loops and path and junction equation
Check Geometry
Linear equation linearized the non-linear equation to linear equation and Set up the matrix to be solve by iteration
Solve a system of linear equations
First iteration?
Qi-Q/Q<0.005
NO
Q=Qi
YES
Print result
Stop

Figure 1.1 flow chart for the computer program

# ChAPTER TWO

# REVIEW OF FUNDAMENTALS

## 2.1 Introduction

Fluid mechanics is the branch of engineering science that is concerned with force and energies generated by the fluids at rest and motion. The study of fluid mechanics involve applying the fundamental principle of mechanics and thermodynamic to develop physical understanding and analytic tools that engineers can used to design and evaluate and processes involving fluid.

The most engineering curricula require fluid mechanics because its principles and methods find many technological applications such as:

1-Fluid transport, 2-Energy generation, 3- Environments control and Transportation

Fluid transport is the movement of fluid from one place to anther so that the fluid may be used or processed. Examples include home and city water supply system, cross-country oil, natural gas and agricultural chemical pipelines, and chemical plant piping. Engineer involved in fluid transport might design systems involving pumps, compressors, pipes, valves, and host of other components.

In addition to designing a new system, engineers may evaluate the adequacy of existing systems to meet new demand, or they may maintain or upgrade existing systems. Very little useful energy is generated without fluid movement. Typical energy conversion devices such as steam turbines, reciprocating engines, gas turbines, hydrodynamic cycles of these devices usually require fluid machinery such as pumps, or compressors to do work on the fluid. Auxiliary equipment such as oil pumps, carburettors, fuel-injection systems, boiler draft fans, and cooling system also involves fluid motion.

Environmental control involves fluid motion. Most building heating systems use a fluid to transport energy from a combustion process or other heat source to the heated spaces. In air-conditioning systems the circulating air is cooled by a flowing refrigerant. Similar processes occur in automobile engine cooling systems, in machine tool cooling systems, and in the cooling of electronic components by ambient air. With the exception of space travel, all transportation takes place within a fluid medium (the atmosphere or a body of water). The relative motion between the fluid and the transportation device generates a force that opposes the desired motion. The application of fluid mechanics to vehicle design can minimise this force. The fluid often contributes in a positive way, such as by floating a ship or generating lift on air plane wings. In addition, ships and aeroplanes derive propulsive force from propellers or jet engines that interact with the surrounding fluid. These examples are by no means exhaustive. Other engineering application of fluid mechanics includes the design of canals, harbours, and dams. The design of large structures must account for the effects of wind loading. In environmental engineering and biomedical engineering, engineers must deal with natural occurring flow processes in the atmosphere and lakes, rivers, and seas or within the human body.

The phenomena of fluid motion are central to the filed of meteorology and weather forecasting. Few engineers can function effectively without at least a rudimentary knowledge of fluid mechanics. Large numbers of engineers deal primarily with process devices and, systems in which knowledge of fluid mechanics is essential for intelligent design, evaluation, maintenance, or decision making. One needs to obtain a firm grasp of the fundamentals of fluid mechanics.

These fundamentals include a knowledge of the nature of fluid and the properties used to described them, the physical laws that govern fluid behaviour, the ways in which these laws may be cast into mathematical form, and the various methodologies that may be used to solve engineering problems.

## 2.2.Fluid Properties

### 2.2.1.Density (ρ)

The mass per unit volume is referred to as the ***density*** of the fluid and is denoted by the Greek letter ρ. The dimensions of density are mass per length cubed or $M/_{L^3}$. The English system of units (abbreviated *ES*) use Slug for the unit of mass and feet for the unit of length. The dimensions commonly used in connection with the ***ES*** system of units are force *F,* length *L* and time *T.* .In the international system, *SI*, the common dimensions are mass, *M*, length, ***L***, and time *T*, Thus using *LFT*, dimensions for the slug can be obtained by relating mass to force through the gravitational acceleration, g, i.e. $F_{Force} = M_{Mass}g$ (acceleration of gravity). Thus since the unit of acceleration are $ft/_{S^2}$ the units of density are,

$$\frac{Slug}{ft^3} \Leftrightarrow \frac{Ib-s^2}{ft^4} \quad \textit{In the English system}$$

In the international system (system international Units) of unit SI, which is an outgrowth of the metric system, mass is measured in the unit of the gram, gr. (or kilogram, kg, which equals to 1000 grams). Force is measured in Newton, N, and length in meters, *(m)*, a Newton is the force required to accelerate 1.kg mass at $1 {}^{m}\!/_{s^2}$ . The units of density in the SI system are:

$$\frac{kg}{m^3} \Leftrightarrow \frac{N-s}{m^4}$$

The conversion of density from the ES system to the SI system or vice versa can be determine by substituting the equivalent of each dimension doing this

$$\frac{1Slug}{ft^3} = 515.363\frac{kg}{m^3}$$

The density of water equals to $1.49\frac{Slug}{ft^3}\ \boldsymbol{or}1000\frac{kg}{m^3}(1\frac{gr}{cm^3})$. Numerical values of density of several common fluids are presented in Appendix A, Table A.1-A.2. Several other fluid properties are directly related to density. The specific volume (v), is define by

$$V = \frac{1}{\rho}$$

It is of considerable use in thermodynamics but is seldom used in fluid mechanics.

The Specific weight (γ) sometimes referred to as the unit weight is the weight of fluid per unit volume, and is denoted by the Greek letter γ. The specific weight thus dimensions of force per unit volume. Its units in the ES and SI systems respectively are

$$\frac{Ib}{ft^3} and \frac{N}{m^3}$$

$$Or$$

$$\frac{kg}{m^2 - s^2}$$

The specific weight is related to the fluid density by the acceleration of gravity or

$$\gamma = g\,\rho \qquad 2.1$$

Since $g = 32.2\frac{ft}{s^2}(9.81\frac{m}{s^2})$, the specific weight of water is

$$\gamma_{(Water)} = 32.2(1.94) = 62.4\frac{Ib}{ft^3}(ES)$$

$$\gamma_{(Water)} = 9.81(1000) = 9810\frac{N}{m^3}(SI)$$

The specific gravity of the fluid is the ratio of the density of the fluid to the density of a reference fluid. The defining equation is

$$S == \frac{\rho}{\rho_{free}}$$

Because specific volume, specific weight, and specific gravity are all directly related to density, they all are constant if the density is constant.

### 2.2.2.Viscosity (μ)

Another important fluid property is its viscosity (also referred to as dynamic or absolute viscosity). Viscosity is the fluid resistance to flow, which reveals itself as a shearing stress within a flowing fluid, and between a flowing fluid and its container. The viscosity is given the symbol μ Greek letter and it is defined as the ratio of the shearing stress τ to the rate of change in velocity, V, or mathematically $\frac{\Delta V}{\Delta y}$. This definition results in the following important equation for fluid shear.

$$\mu = \frac{\Delta V}{\Delta y} \qquad 2.2$$

in which $\frac{\Delta V}{\Delta y}$ is the derivative of the velocity with respect to the distance $y$ The derivative $\frac{\Delta V}{\Delta y}$ is called the velocity gradient. Eq. (2.2) is valid for viscous or laminar flow but not for turbulent flow when much of the apparent shear stress is due to the exchange of momentum between adjacent layers of flow.

From Eq.2-2 it can be determined that dimensions of viscosity are force multiplied by time divided by length squared or $\frac{F*t}{L^2}$. Its units in the ES and SI system are respectively,

$$\frac{lb-s}{ft^2} or \frac{Slug}{ft-s} and \frac{N.s}{m^2}$$

Or

Kg/ms

Occasionally the viscosity is given in poises $1Poises = 1\frac{Dyne\sec}{cm^2}$. or 1 $Poise = 0.1\frac{N-s}{m^2}$. Because of its frequent occurrence, the absolute viscosity is divided by the fluid density and is called the kinematic viscosity, ν. The Kinematic viscosity thus is

$$\nu = \frac{\mu}{\rho} \qquad 2.3$$

The dimensions of kinematic viscosity are lengths squared per time. Common units for ν in the ES and SI system respectively are:

$$\frac{ft^2}{s} and \frac{m^2}{s}$$

Another often used unit for kinematics viscosity in the metric system is the Stoke (or centistoke =0.01 Stoke). One stoke equal $1\frac{cm^2}{\sec}$.

The viscosity of many common fluids such as water depends upon temperature but not the shear stress or dv/dy. Such fluids are called Newtonian fluids to distinguish them from non-Newtonian fluids whose viscosity does depends upon dv/dy. Table A.1 gives values of the absolute and kinematic viscosity of water over a range of temperatures. Table A.1-A2 in Appendix A contains viscosities of several common fluids.

## 2.2.3.Pressure

Pressure is a fluid property of importance. Most fluid mechanics problem involves prediction of fluid pressure or with the integrated effect of pressure over some surface or surface in contact with the fluid. Unlike density, which is usually one of the known quantities in fluid mechanics, pressure is usually an unknown quantity to be determining by analysis or experiment. Pressure is defined as follows.

the normal compressive force per unit area acting on real or imaginary surface in the fluid

$$P = \lim \frac{df}{da}$$

Pressure is expressed in units of force per unit area and is measured with respect to one of two datum's. Pressure measured relative to local atmospheric pressure is called *gauge pressure.* Pressure measured or define relative to zero pressure is called *absolute pressure*. Gauge and absolute pressures are related as follows.

***Absolute pressure =gauge pressure + atmospheric pressure in vicinity of gage.***

To convert from gage pressure or absolute pressure, you must know the atmospheric pressure at the time and location of measurement. Often this information is not available. In such cases, a "standard" value of 101,330 pa (14.696 $\frac{Ib}{In^2}$, abs; 29.92 in.Hg, abs)is used for atmospheric pressure. The actual atmospheric pressure can be measured with an instrument called a barometer.

In many situations that arise in fluid mechanics, one is more concerned with the difference of pressure than with levels of pressure. Pressure difference is the same whether the pressure is considered as absolute or gage, as long as all pressures are based on a datum. A pressure difference (say, between two locations in a pipeline) is not expressed in either "gauge" or "absolute" units; this designation applies only to pressure level.

## 2.3.The fundamental Principles

Much computation in engineering and the physical sciences are based on a relatively few fundamental principle and concepts. Most important among these are Newton's laws of motion, and conservation of mass, energy, and momentum. With limited exception mass is not created or destroy, energy is only converted form one to another, and momentum is only changed by force acting through time. In fluid mechanics the quantification of the conservation of mass is referred to, as the continuity equation in its various forms. An expression of the conservation of energy frequently used in hydraulics is Bernoulli equation. The momentum principle is extremely useful in determining external force acting on moving fluids. The equation obtained from theses three conservation principles are the most fundamental equations used for solving fluid mechanics problem.

### 2.3.1.Continuity

In more general forms, mathematical equations, which embody the principle that mass is conserved, are differential or integral equation. This is the case because in general the quantities which describe the flow, such as point velocities, are functions of position in space and time. For flows whose variables vary with a single space co-ordinate, and do not vary in time, an algebraic equation can be used for describing the fact that mass is conserved. Flow that does not change in time is called "*steady flow*", and if only one space co-ordinate is used the flow is one-dimensional.

In solid mechanics the conservation of mass principles is applied by simply noting that the mass of any body remains constant. When dealing with the fluid flow it is generally more convenient to deal with the amount of fluid mss passing a given section of the flow rather than to keep track of the position of all individual particle of fluid. Thus the continuity equation will deal with mass flux passing a section of flow instead of mass alone. Mass flux is simply the flow of mass or mass per unit time, and has the unit of $\frac{Slug}{s}$ in the *ES* system and $\frac{kg}{s}$ in the *SI* system.

Denoting mass flux by $G$ it can be related to density $\rho$ the area through which flow occurs and the fluid velocity by:

$$G = \rho AV = \rho Q \qquad 2.4$$

in which A is the cross-sectional area of the section and V is the average velocity of the flow through the section. The area A is normal to the direction of the velocity. The symbol Q in the later of Eq.(1-4) is the volumetric flow rate with dimension of $^{L^3}\!/_{T}$ and given by:

$$Q = VA \qquad 2.5$$

For steady flow in a conduit the mass flow through two section denoted by subscripts 1 and 2 of pipe some distance part must be equal if the flow is steady.

Therefore:

$$G_1 = G_2$$

so that

$$\rho_1 A_1 V_1 = \rho_2 A_2 V_2 \qquad 2.6$$

Eq (2.6) is one of the continuity equation. For fluids that are incompressible, i.e. whose densities are constant regardless of the pressure, the continuity equation reduces to

$$Q_1 = Q_2$$

so that

$$A_1V_1 = A_2V_2 \qquad 2.7$$

Occasionally the weight of flows rate $W$ is wanted. It equals

$$W = gG = g\rho AV = \gamma AV \qquad 2.8$$

In dealing with a junction of two or more pipes the continuity principle states that the mass flow into the junction equals to the mass flow out of the junction. Mathematically this principle is,

$$\sum Gi = \sum \rho iQi = 0 \qquad 2.9$$

in which the subscript (i) take on the values for the pipes, which join at the junction and the summation, indicates the sum of the G with proper regarded for the sign. Again for incompressible flows Eq.(2-9) reduces to,

$$\sum Qi = 0 \qquad 2.10$$

Equation 2.9 or 2.10 will play an important role in the analysis of networks of pipes. Junctions in such networks one commonly referred to as nodes. If the velocity is not constant through the flow section, calculus can be used to determine the mass, weight, or volumetric flow rates past a section.

## 2.4.Conservation of Energy (Bernoulli Equation):

Flowing water contain three forms of energy which are of interest in hydraulic. Due to its motion it possesses kinetic energy, and it contain two forms of potential energy, one by virtue of its elevation and the other by virtue of its pressure.

Energy can be defined as the ability to do work. Work result from force moving through some distance and therefore energy has the dimensions of (force * length) *Fl*. In a fluid flow the amount of energy passing a section will increase with time simply because more fluid will have passed that section. Consequently, it is more convenient to deal with energy per unit of fluid, or $E/_M = Fl/_m$. In the ***ES***, energy per unit mass has the units of $\frac{Ib-ft}{Slug}$, and in the *SI* units are $\frac{N-m}{kg}$. The energy per unit mass due to the elevation of the fluid is gz, in which z is the vertical distance above some datum and g is the acceleration of gravity. The energy per unit mass resulting from the pressure is $\frac{p}{\rho}$ in which p is the fluid pressure. The kinetic energy per unit mass of fluid is $\frac{V^2}{2}$. The sum of these three energies is,

$$E = gZ + \frac{p}{\rho} + \frac{V^2}{2} \qquad 2.11$$

Energy/unit mass due to elev. + due to pressure + kinetic

### 2.4.1.Potential Energy due to Elevation per Unit Mass

Consider a mass of fluid with its centre of mass a distance z above an elevation datum as shown in Fig. [2.1]

With respect to the datum this mass possesses potential energy equivalent to its ability to do work in falling through the distance z. This work equals its Weight (i.e. its mass time the acceleration of gravity) multiplied by the distance z or $g\ M\ z$. To get the energy per unit mass, $g\ M\ z$ is divided by the mass M giving gz or the first term on the right of Eq.(2-11).

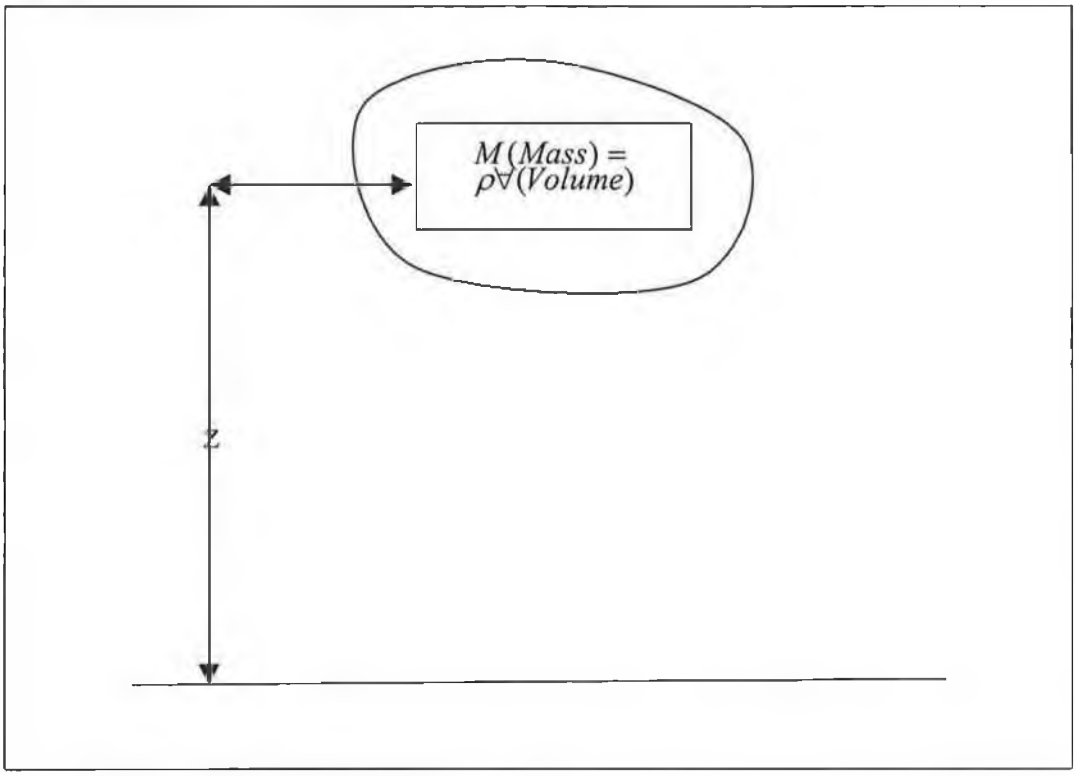


Fig. 2.1 energy per unit mass due to elevation

## 2.4.2.Potential Energy due to Pressure per unit Mass.

A section in which the pressure of fluid is used to do work, for example, the fluid pressure in the vessel is used to transfer fluid to a distance *l*. If the average pressure acting over the piston is P, the fluid pressure force on the piston is *PA* in which *A* is the area of the piston. Thus the energy obtained from the fluid is doing the work in moving the piston through the distance l, equals P*A*). The mass of fluid doing this work is that contained in the cylinder between the initial and final position of the piston, a distance l apart, or M= ρ *A* l. Consequently the energy per unit mass of the fluid due to pressure equals $\frac{pA\ell}{\rho A\ell} = \frac{p}{\rho}$. The second term on the right of Eq(2-11).

## 2.4.3.Kinetic Energy per Unit Masses.

The kinetic energy of any mass M equals half the mass times the velocity, or $\frac{1}{2}MV^2$. To obtain the kinetic energy per unit mass this quantity is divided by *M* resulting in $\frac{1}{2}V^2$ or the final term in Eq (2-11).

## 2.4.4.Total Energy per Unit Mass.

The three simple descriptions of the conservation of energy help in visualising the three forms of energy per unit mass that water flowing in a pipe contains. In using these in a conservation of energy equation it is necessary to account for what happens to the energy level between two level positions along the flow path or pipeline.

Whenever fluid passes a fixed wall or boundary, fluid friction exists. This friction changes some of the useful flow energy into heat, or other forms of energy, which are non-recoverable from the hydraulic viewpoint. Methods for determining the magnitude of this lost energy per unit mass of flowing fluid will be discussed. For now it will be simply denoted by $E_l$.

A pump may exist in a pipeline, which supplies energy to each unit mass of fluid passing through it, or a turbine may extract energy therefrom. $E_m$ will denote these mechanical energies with the subscript standing for all forms of external mechanical energy. A pump produces a positive amount of $E_m$ in the fluid and a turbine produces a negative amount.

With these additional symbols for energy losses and all other forms of mechanical energy, the conservation of energy between two section within flow denoted by 1 and 2 respectively, such as depicted in the sketch below, is given by.

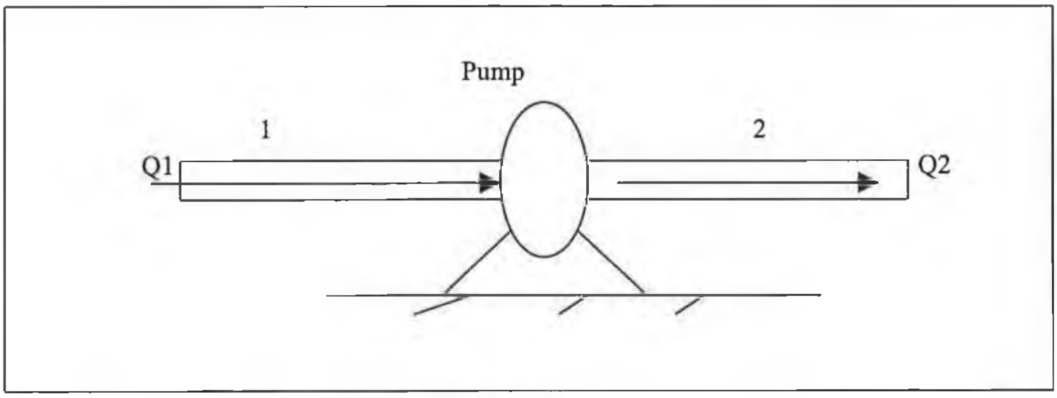


Figure 2.2

$$E_1 + E_m = E_2 + E_l \qquad \text{(2-12a)}$$

Upon substituting from Eq.(2-11)

$$gZ_1 + \frac{P_1}{\rho} + \frac{V_1^2}{2g} Em = gZ_2 + \frac{P_2}{\rho} + \frac{V_2^2}{2} + El \qquad (2\text{-}12b)$$

This equation is very important in computations dealing with incompressible fluid flow.

## 2.4.5.Energy and Hydraulic Grade Lines.

The energy Grade line, also called the Energy Line or simply El, is a plot of the sum of the three terms in the work-energy equations, which is also the Bernoulli sum:

$$\frac{P}{\gamma} + \frac{V^2}{2g} + z \qquad 2\text{-}13$$

Since each term has units of length, one can conveniently superimpose a diagram of the behaviour of each energy term, and the sum, on a drawing of the physical flow problem. For example, a Pitot tube, inserted into a flow to cause locally at its tip a point of zero velocity so the velocity head is converted into additional pressure head there, will cause the liquid to rise to the elevation of the EL for that point in the flow. The Hydraulic Grade Line, or HGL, is the sum of only the pressure and elevation heads. The sum of these two terms is also called the piezometric head, which a can be conveniently measure by a piezometer tube inserted flush into the side of a pipe. It is also important to recognise that any HGL can quickly be located on a diagram if the EL has already been located; downward measure by the amount of local velocity head from the EL to locate the HGL.

Fig.2.3 portrays the relation of the individual heads terms to the EL and HGL and the head that is lost between section 1 and 2

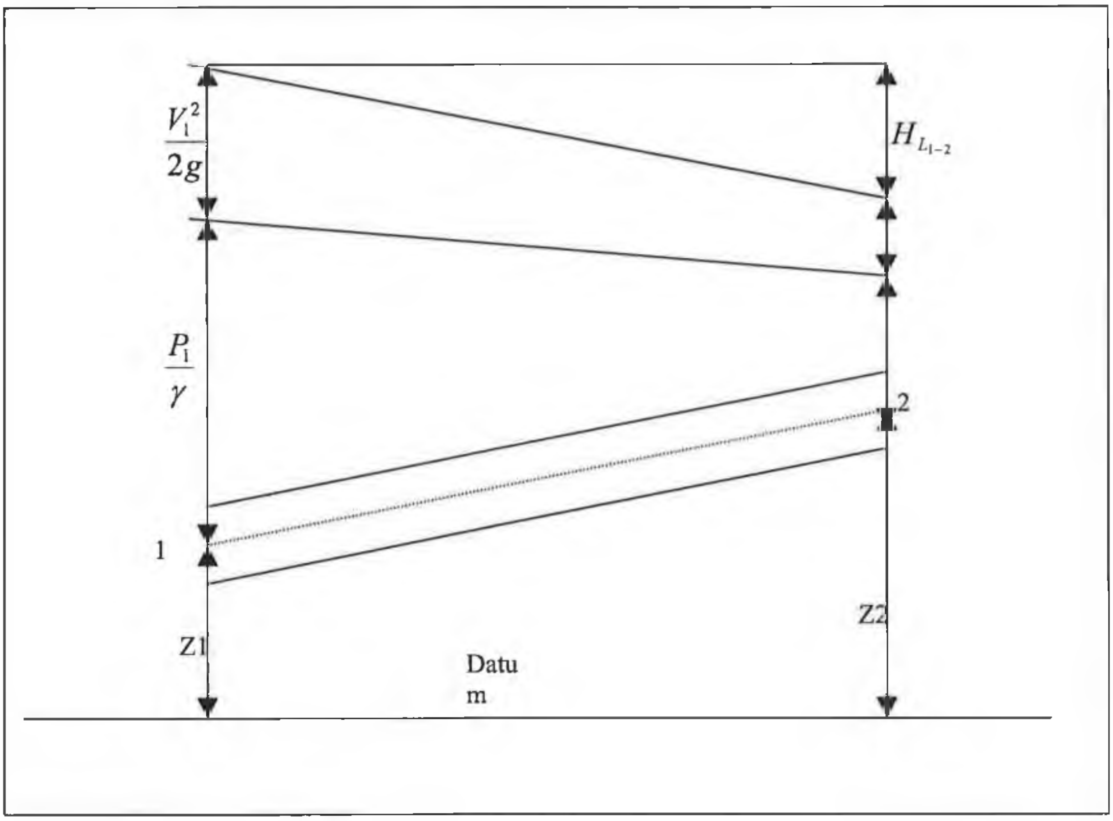


Figure 2.3 the Eland HGL in relation to individual heads and the heads loss

The heads loss is responsible for representing accurately two kinds of real fluid phenomena, head loss due to fluid shear at the pipe wall, called pipe friction, and additional head loss caused by local disruptions of the fluid stream. The head loss due to pipe friction is always present throughout the length of the pipe. Valves, pipe bends, and other such fittings cause the local disruptions, called local losses. Local losses may also be called minor losses if their effect, individually and/or collectively, will not contribute significantly in the determination of the flow; indeed, sometimes minor losses are expected to be inconsequential and are neglected. Or a preliminary survey of design alternatives may ignore the local or minor losses, considering them only in a later design stage.

### 2.4.5.Conversion of Energy per Unit Weight to Power

Available information for pump and turbine refer to power and efficiency and not the energy per unit weight of fluid through the device. Therefore, equations are needed relating $E_m$ to power or horsepower Hp. These equations are.

$$P = \frac{\gamma QHp}{\eta} = \frac{W * Hp}{\eta}$$

And

$$Hp = \frac{\gamma QHp}{550\eta} = \frac{WHp}{550\eta} \qquad \text{ES (2-14a)}$$

$$Hp = \frac{\gamma QHp}{746\eta} = \frac{WHp}{746\eta} \qquad \text{SI (2-15a)}$$

for a pumps, and

$$P = (\gamma QHt)\eta = (WHt)\eta \qquad \text{(2-14b)}$$

$$Hp = (\frac{\gamma QHt)\eta}{550} = \frac{(WHt)\eta}{550} \qquad \text{ES (2-15b)}$$

$$Hp \frac{(\gamma QHt)\eta}{746} = \frac{(WHt)\eta}{746} \qquad \text{SI (2-15b)}$$

for turbines

Considering the dimension of the quantities involved one can derive equation (2-14) and (2-15). Power is the rate of doing work or work per second. The energy exchange between the fluid and the device comprises the work and therefore energy lost or gained per second by the fluid should be equated to power after begin modified by the efficiency of the device in carrying out this conversion. Since $H_m$ and $H_p$ for a pump or $H_t$ for turbine) is energy per unit weight. We can obtain power by multiplying $H_m$ by the weight flow rate W=ρ Q. the factor 550 and 746 in equation 2-14 and 2-15 are the conversion of power to horsepower in the ES and SI systems respectively, since 1Hp=550 $\frac{ft-Ib}{s}$ (1 hp=746 $\frac{N.m}{s}$).

## 2.6.Momentum Principle in Fluid Mechanics:

The third conservation principle, that of momentum, provides an additional powerful tool to solve many fluid flow problem, particularly those dealing with external devices acting on the fluid system, such as at elbows, junction, and reducers or enlargers. A widely used equation resulting from the momentum principal is

$$\vec{F} = \rho Q(\vec{V}_2 - \vec{V}_1) \qquad 2.16$$

in which $\vec{F}$ is the resultant force (a vector with magnitude and direction) which acts on the fluid in a control volume being analysed, $\vec{V}_2, \vec{V}_1$ are the average velocities (also vector) leaving and entering the control volume respectively. The problem, which is handled in here, will not be solved by use of the momentum equation.

# CHAPTER THREE

# PIPE NETWORKS ANALYSIS

## 3.1.Introduction

This chapter describes the analysis of steady flow in pipe systems. The analysis or design of a pipe system is accomplished by patching together the information about the pipes, and components. In analysis problem the system configuration is given and the solution process endeavour to determine the discharge in every pipe and the pressure, etc at every node of the network, and pressure drop or required pumping power to maintain flow in the system. The analysis of a pipe network can be one of the most complex mathematical problems that engineers are called upon to solve, particularly if the network is large, as occurs in the water distribution systems of even quite small cities. This analysis is needed whenever significant changes in patterns or magnitudes of demand or supplies occur in municipal water or natural gas distribution systems. These changes occur whenever new residential subdivisions or industries are attached to the existing systems or new sources of supply are tapped. In the recent years a number of significant improvements have occurred in the method and techniques used to analysis steady flow in large pipe networks. Some commonly used algorithms for pipe networks analyses are described in this section. Details can be found in the references cited for each method.

## 3.2.History of Analysis and Design of Pipe Networks.

Steady state analysis of pressure and flow in piping system is a problem of great important in engineering. The basic hydraulic equations describing the phenomena are non-linear equations, which can not be solved directly. These equations have been expressed in two principal fashions. They have been written in terms of the unknown flow rate in the pipes that referred to as loop equation. Alternatively, they have been expressed in terms of unknown heads at junctions throughout the pipe system (node equation). Several algorithms have been proposed for solving these equations and there is a considerable amount of published material dealing with pipe network analysis, and some principal contribution of historical interest will be cited here. Hardy-Cross [4]- formulated two methods one of them considered only closed loop networks with no pumps, a method for solving the loop equation based on adjusting flow rate to individually balance each of the energy equation is described. This method was very widely and is often referred to as the Hardy-Cross method. Although it is not as widely used, Hardy-Cross-described a second method for solving the node equation by adjusting the head at each node so that the continuity equation is balanced. A number of subsequent papers have appeared which further described these methods or computer programs utilising these methods. [5-6-7-8-9] Because the adjustments are computed independent from each other, convergence problem using the method described by Hardy Cross was frequently noted and procedures developed to improve convergence. Martin and Peters [10] and EPP and Fowler [11] described a procedure to simultaneously compute the flow adjustment for closed loop system. Based on the N-R method this procedure had much improved convergence characteristic and formed the basis for more general application [12-8]

Both methods just described for solving the loop equation require an initial balance set of flow rate and converge depends on a degree on how close this initial set of flow rate is to the correct solution. A method, developed by Wood [15], solves the entire set of hydraulics equation simultaneously after linearization the non-linear terms in the energy equation. This procedure was first described for closed loop system [13] and has subsequently been modified for more general application [14,15]. A similar approach has been developed for the node equation where all the node equations are linearized and solved simultaneously, and Shimmer and Howard [16] described this method for closed loop system. Additional reference to this method has been mad in reference [17.118].

Wood and Charles [13] developed The linear theory method which can also be regarded as an application of the Newton-Raphson technique in the sub-domain of loops, but it requires the solution of a large system of equations (number of loops + number of nodes) although reducing the risk of failures. The optimisation methods by Coollins et al., [19], Contro and Franzetti1b, [20]], minimise a non-linear convex objective function subject to linear equality and inequality constraint using mathematical optimisation techniques. The advantage is obvious since the convexity of the objective function obtained to the linear constraints guarantees the existence and uniqueness of the solution. Unfortunately the numerical solution of the problem requires efficient non-linear programming algorithms, thus reducing the practical when dealing with a large complex networks. A method was developed by Todini [21], which may be regarded as a bridge between the optimisation method and the Netwon-Raphson based techniques in that it starts from minimisation of slightly modified model Collins et al [19]. In order to prove the existence and uniqueness of the solution of the partly linear and partly non-linear system to be simultaneously solved in terms of the nodal head and unknown flows in pipes The Netwon-Raphson technique is then applied in this enlarged space of flow and heads where the proof of the existence and incite of the solution holds, which is the key to the unconditional convergence of the method.

### 3.2.1.Convergence Problem

Convergence problem is largely unreported for the improved methods developed for solving the loop equation. However, additional convergence problems have been reported for methods based on the node equation since Hardy Cross [4] first alluded to such problem. In his original paper by Hardy Cross he noted that "convergence was slow and not very satisfactory" when employing the head adjustment method he developed. This was attributed to using initial head estimates, which were not very good. Of the two methods described by Hardy Cross, the method of adjusting flows becomes the most widely used method. Convergence problem using this method was also recognised, however, and several suggestions were made for improving convergence. Investigations have advocated the use of an over-relaxation factor to multiply the flow adjustment factor [4-13] Hoag and Weinberg[22] suggested using a selective procedure for choosing loops as a mean of accelerating convergence . It appears that these and other procedures suggested for improving convergence of this method will improve only certain situation and will not assure convergence. Dilligham [6] stated that when the method of adjustment heads at individual nodes is applied to a large network it may not converge or may converge very slowly. He described some procedures for improving convergence. Rodinson and Rossum [9] who developed a computer program based on this method say that," convergence is slow when a network contains short length of large diameter mains and convergence is not assured if there are dead end mains". They further state that " convergence may not occur if check valves are present". The method of simultaneously adjusting heads normally gives convergence much more rapidly which lead (Limoux) to state for which convergence was assured [18]. However, it appears that this assessment is optimistic and problems have been noted with this method. Shamir and Howard [16] who also reported that there is a possibility that a solution can not be obtained have noted oscillations. Liu also stated "for poor initial input the method (simultaneous node adjustment) may diverge from the true solution or converge slowly"[18]. Collins and Kennington presented some data which documented convergence problem for a large network using this method [23].

The reliability of the algorithm employed for network analysis is of great importantnce. Failure to obtain a solution is an inconvenience as the failure leads to poor design or management of water distribution systems. The purpose of this study is to document reliability problem, which may occur using various popular algorithms.

Problem of slow convergence and lack of convergence for some mentioned methods have been reported and in particular an extensive reliability study was carried out by Wood [24], the most extensively used algorithms were tested under a wide variety of conditions. Wood and Rayes [25] also published the result of this thorough research. According to their finding, only the method of "simultaneous path" (loop equation solved with a Newton-Raphson algorithm) ,and the linear theory method (the gradient algorithm) and the linear theory method (the gradient loop-algorithm), not the original method published by Wood and Charles [13] are recommended for the reliable solution of the water distribution network analysis problem, since the rest of the algorithms tested shown "significant convergence problem".

The "simultaneous path" and the linear theory method were compared with the new gradient method, based on a simple ill-conditioned problem. Todini (1979) and Pilati and Todini [28] originally proposed the gradient method. Their result showed that the gradient method has the advantage over the simultaneous path adjustment method and the linear theory method described as following.

It can directly solve partly looped and partly branched networks, while for the simultaneous path method the problem needs to be transformed into an equivalent looped network, prior to the application of the iterative algorithms. This has to be done by the user (manually) or incorporated in additional subprogram. The gradient method does not need a loop or path definition as in the case of the simultaneous path adjustment and the linear theory method. Even though a computer can also do this task, it implies an additional computational cost. The gradient method can solve in straightforward way networks that during certain periods of operation can become disjointed (due to the action of check valves or pressure reducing valves, for example). The simultaneous path adjustment and the linear theory method can not cope with this situation, although it might be possible to implement an additional subroutine that could solve this shortcoming. Again this means additional computational cost. The reliability of the method, in term of its ability to converge for both flows and the piezometric heads, under extreme cases based on ill conditioned problems, makes it desirable for optimum intervention to avoid convergence problem. In addition, this has been implemented on personal computer, given low requirement of memory and its high speed. The algorithm needs about

40% of the memory need for the linear theory method and only 5% more than the simultaneous path adjustment method.

Each of these methods requires special iterative computation where the solution is improved until a specified convergence criterion be met. If a sufficiently stringent convergence criterion is met, the solution will be essentially identical for all methods. In some cases, however, it is not possible with certain algorithms to meet a specified convergence criterion regardless of the number of trials completed. In other cases a seemingly stringent convergence criterion may be met but the solution is still in considerable error. Convergence difficulties such as these have been previously noted and reported.

## 3.3.Basic Relation Between Network Element

The two basic principle, upon which all network analysis is developed are (1) the conservation of mass, or continuity principle and (2) the work energy principle, including the Darcy-Weishbch or Hazen-Willams equation to define the relation between the head loss and the discharge in a pipe. The equations that are developed from the continuity principle will be called the junction continuity equations, and those that are based on the work energy principle will be called the energy loop equations. The number of these equations that constitute a non-redundant system of equations is related directly to fundamental relation between the number of pipes, number of nodes and number of independent loops that occur in the branches and looped networks. In defining these relation NP will be denoted to the number of pipes in the networks. NJ will denote the number of junctions in the networks, and

NL will denote the number of loops around which independent equations can be written. In defining junctions, supply source will not be numbered as a junction. A supply source is a point where the elevations of energy line, or hydraulic grade line is established and a junction or node is a point where two or more pipes join. A node can exist at each end of a "dead end " pipe; this instance is an exception to the usual rule, where only one pipe is connected to the node. In branched system there are by definition no loops, and NL=0 for any branched system, or NP=NJ-1, unless a reservoir is shown at the end of one pipe and this is not considered to be a junction. then NP =NJ. This situation actually occurs. Figure 3.1 and 3.1b depict a small-branched network and a small looped network respectively.

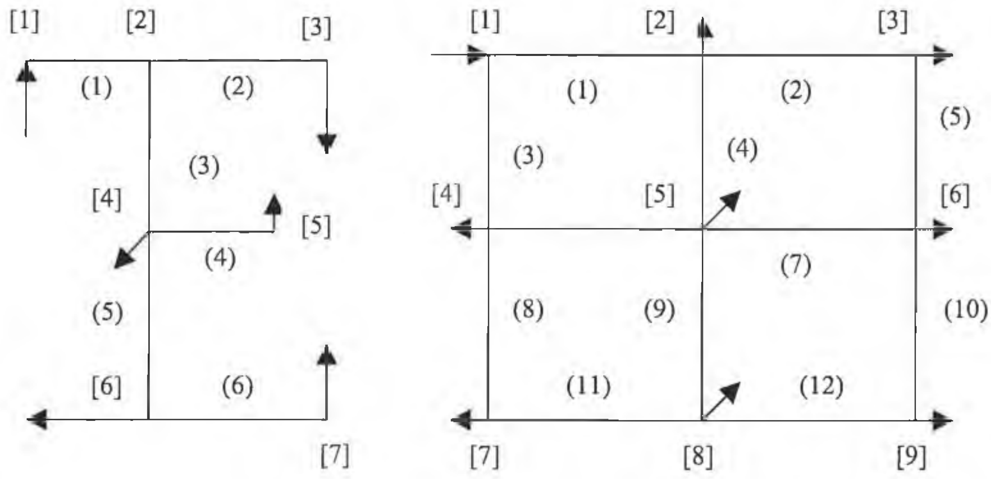


*Figure 3.1 (a) a small-branched system.* *(b) A small looped system*

In the branched system the number of nodes is 7 and the number of pipes is 6 (one less that the number of node) whereas in the looped system there are 12 pipes and 9 nodes, i.e., the number of nodes is less than the number of pipes.

For a looped network the number of loops (around which independent energy equations can be written) is given by

$$NL=NP-NJ\text{------}(3.1)$$

If the network contains two or more supply sources,

$$NL=NP-(NJ-1)=NP-NJ+1\text{----}(3.2)$$

If the network contains less than two supply sources and the flow from the single source is determined by adding all the other demands, then this source is shown as a negative demand and the source is called a node. This case is noted in the small looped network in fig 3.1 so that $NP$=12,$NJ$=9 and $NL$=12-(9-10)=4.

Equation 3.2 also applies for branched system with $NL=NP-NJ+1$=0, since branched system can have at most one supply source. Actually, every pipe system must have at least one supply source, but sometimes the source is not shown since the discharge from this supply source is known , and the source is replaced by a negative demand, which is a flow coming into this junction, equal to the sum of other demands. When this is done, the elevation of the energy line (HGL or pressure) must be specified at a node so the other HGL elevation can be determined. Energy loops that begin at one supply source and end at another called is pseudo loops, i.e., these loops do not close on themselves. The number of pseudo loops, which are numbered as part of NL, equal to the number of supply source minus 1. In forming a pseudo loop all supply sources must be located at the end of the pseudo loop. It is generally possible to form more loops that are needed to produce the set of independent equations. As each loop if formed, see that at least on pipe in the new loop is not a part of any prior loop, in this way the formation of redundant loops can usually be avoided.

## 3.4.Algorithms for Pipe Networks

The development of new methods has been possible mainly due to the availability of digital computer, allowing for the use of more sophisticated mathematical techniques Another motivation for the development of new algorithms has been clearly the unreliability of the existing methods when dealing with some ill-conditioned problems. Optimum design and operation routines also need to be based on robust analysis algorithms.

Piping system are constructed by piecing together components. These components transport the fluid, change its direction, control flow rate, divide it, speed it, or slow down .In addition to conveying fluids, most of the components are responsible for mechanical energy loss. Only the pump does not contribute a net loss! But instead increase the fluid's mechanical energy. Analysis of pipe consists of relating flow variables, such as energy loss and flow rate, to pipe system parameters, such as size, shape, length, and number and location of fittings.

### 3.4.1.Pipe Networks Geometry

Basic geometrical considerations for pipe networks are summarised as follows. A pipe network is comprised of a number of pipe sections, of constant diameter section which can contain pumps and fittings such as bend and valve. The end point of the pipe section are nodes which are identified as either junction nodes or fixed grad nodes .A junction node is a point where two or more pipe sections join and is also a points where a flow can enter or leave the system. A fixed grade is a point where a constant head is maintained such as a connection to a storage tank or reservoir or to constant pressure region. In addition primary loops can be identified in a pipe network and these include all closed pipe circuit within the networks which have no additional closed pipe circuit within them.

When junction nodes and fixed grad nodes, and primary loops are identified the following relationship hold.

$$p = j + l + f \text{ ---------}(3.3)$$

In which p= number of pipes; j= number of junction nodes; l= number of primary loops, and f= nmuber of fixed grade nodes.

### 3.4.2.Basic Equation

Pipe networks equation for steady -state analyses have been commonly expressed in two ways. Equations, which express mass continuity and energy conservation in term of the discharge in each pipe section, have been referred to as loop equations and this terminology will be followed here.

A second formulation, which expresses mass continuity in term of head at junction nodes, produces a set of equations referred to ass node equations.

### 3.4.3.Loop Equation

Equation (3.3) offers basic formulation for a set of hydraulic equations to describe a pipe system. In terms of the unknown discharge in each pipe, a number of mass continuity and energy conservation equations can be written equalling the number of pipes in the system. For each junction node continuity relationship equating the flow in to the junction ($Qin$) to the flow out ($Qout$) is written as:

$$\sum Qin - \sum Qout = Qe \text{---3.4}$$

Here $Qe$ represents the external inflow or demand at the junction nodes. For each primary loop the energy conservation equation can be written for a pipe section in the loop as follows:

$$\sum Hl = \sum Ep(l - equation)\text{---(3.5)}$$

in which HL= energy loss in each pipe section including minor losses, and EP = energy put in to the liquid by pumps. If there are fixed grad nodes (f -1) independent energy conservation equation can be written for paths of pipe section between any two-fixed grad nodes as follows:

$$\Delta E = \sum Hl - \sum Ep(f - 1 - equation)\text{---(3.6)}$$

in which $\Delta E$ = the head difference between the fixed grade nodes. The terms in the energy equation can be expressed as function of flow rate. Thus, the continuity equation (3.4) and the energy equation (3.5) and (3.6) form the set of p simultaneous equations in terms of unknown flow rate, which are termed as the loop equation. Since these are non-linear equations, no direct solution is possible. Their algorithm for solving the loop equation are presented.

### 3.4.4.Node Equations

The analysis is carried out in terms of an unknown total head H, at each junction nodes in the pipe system. The basic relationships used in the continuity relationships Eq 3.4 the flow rate, in the pipe section are connecting nodes labelled a and b, is expressed in term of the head at junction nodes a, *Ha*, the head at the other end of the pipe section, *Hb*, and the loss coefficient for the pipe, *Kab*. This

$$Qab = \left|\frac{H_a - Hb}{K_{ab}}\right|^{\frac{1}{n}} \text{------------}\mathbf{(3.7)}$$

This expression assumes that the pipe section contain no pump and a head loss relationship is used of the form

$$hl = KQ^n \text{-------}\mathbf{(3.8)}$$

in which the term *K*= the loss coefficient for the pipe section and is a function of a pipe parameter and flow condition and is dependent on the head loss expression used and may include minor loss term. The exponent (n) also is dependent on the head loss expression used. Combining Eq 3.4 and 3.7 gives:

$$\sum_{b=1}^{N} \pm \left[ \frac{Ha - H_b}{K_{ab}} \right]^{\frac{1}{n}} = Qe \text{ ---------------} \textbf{(3.9)}$$

This expresses continuity at junction nodes a in which *N* pipes connect, in terms of the head at *Ha*, and the head at adjacent nodes, *Hb*, the sign of the term in the summation is dependent on whether, the flow is into or out of the junction nodes a. A total of **j** equations are written in this manner.

The basic set of equations can be expanded to incorporate pumps. For each pump, junction nodes are identified at the suction and discharge sides of the pumps. Two additional equations can be written as terms of two additional unknown heads at the suction, and discharge side of the pump and the adjacent heads. One equation expresses flow continuity in the suction and discharge lines using equation (3.7) and a second equation relates the head change across the pump to the flow in either the discharge or suction line.

For pipe networks of ***(j)*** junction nodes and *NP* pumps a set of (j+2 NP**)** equations are obtained. These represent the full set of pipe network node equations which are expressed in terms of the unknown head at junction nodes and pumps suction and discharge head at all pumps in the pipe system. Like the loop equation, these are non-linear algebraic equations and no direct solution is possible.

## 3.5.Algorithm for Solution of Loop Equation:

Three methods for solution of the loop equations have been developed and are in significant use today. Each use gradient methods (Newton extrapolation) to eliminate non-linear term from the energy equation.

### 3.5.1.Single Path Adjustment (path) Method:

This method of solution was described by Hardy cross (4) and is the oldest and most widely used technique. The original method was, however, limited to closed loop system and includes only line losses. The procedure has since been generalised. The method of solution is summarised as follows.

1. Determine an initial set of flow rate, which satisfy continuity at each junction node
2. Compute a flow adjustment factor for the path of pipes for each energy equation, which tend to satisfy the energy equation written for that path. The application of this correction factor will not disturb the continuity balance.
3. Use improved solution for each trial step 2 until the average correction factor is within a specified limit.

The flow adjustment factor for a path is computed from the energy equation for that path and is intended to correct the flow rate so that the energy equation is satisfied. However, a correction for a particular path will disturb the energy relationship for the other entire path, which have common pipes. A trial with this method requires a flow adjustment to all paths in the pipe networks (*l* loops and f-1 path between fixed grade nodes).

### 3.5.2.Simulation Path Adjustment (S-Path) Method:

In order to improve the convergence characteristic, a method of solution was devised which simultaneously adjusts the flow rate in each path of pipes representing energy equation (3.8). This method can be summarised as follows.

1. Determine an initial set of flow rate, which satisfy continuity at each junction mode
2. Simultaneously compute a flow adjustment factor for path, which tend to satisfy the energy equation without disturbing the continuity balance.
3. Use improved solution and repeat step 2 until the average flow adjustment factor is within specified limit.

The simultaneous path flow adjustment factor requires the simultaneous solution of (l+f-1) equation. Each equations accounts for the unbalance in the energy equation due to incorrect values of flow rate and includes the contribution for a particular path as well as contribution from all other paths, which have pipes common to both paths. Sets of *(l+f-1)* simultaneous linear equations are formed in terms of flow adjustment factor. This linear equation can be solved using standard procedures and the solution provides an improved set of balance flow rate, which can be used, for another trial. Trails are repeated until a specified accuracy is attained.

## 3.6.Algorithm for Solving of Node Equation:

### 3.6.1.Single Node Adjustment (nodes) Method

This method is summarised as follows:

1. Assume a reasonable head for each junction nodes in the system. This assumed head does not have to satisfy any condition. However the better the initial assumption the fewer the required trials.
2. Compute the head adjustment factor for each junction node, which tends to satisfy flow continuity at the junction.
3. Repeat step 2 until the average correction factor for heads is within a specified accuracy or some other specified convergence criterion is satisfied.

The head adjustment factor is the change in head at a particular node, which will result is satisfying continuity equation (3.4) considering the head at the adjacent nodes as fixed. Again a gradient approximation is used to compute the required head change. A single trial for this method requires the adjustment of the head for each junction node within the pipe system. The trials continue until the specified convergence criterion is met.

### 3.6.2.Simultaneous Node Adjustment (s-node) Method:

This method is based on a simultaneous solution of the basic equation pipe networks and requires linearization of these equations in terms of approximate values of the head (16). This produces set of (*j+2np*) simultaneous linear equations (in which NP is the number of pumps). These equations are solved as follows, starting with any assumed values for the junction node heads these linear equation are solved simultaneously for an improved set of junction node heads. These heads are then used to linearized the set of node equations and the procedure is repeated until subsequent calculation satisfy a stated convergence or accuracy criterion.

## 3.7.Linear Method

This method is based on a simultaneous solution of the basic hydraulics equation for the pipe system and has been reported for closed loop system **(13)** and general system **(15)**. Since the energy equation for the paths are non-linear one and no direct solution is possible, these equations are first linearized in terms of an approximate flow rate, *Q*, in each pipe to make solvable in iterative fashion. This is traditionally obtained via a Taylor expansion of the head loss-flow relationship h (Q), retaining only the linear terms.

$$h(Q) \approx h(Q_i) + \frac{\delta h}{\partial Q}[(Q - Q_i)]_{Q=Q_i} \text{---}(3.10)$$

Or simply:

$$h(Q) \approx H_i + J_i \Delta Q \text{---}(3.11)$$

where:

$h(Q) =$ head loss (gain) in the link produced by the flow q.

$H_i = h(Q)$ Evaluated at $Q = Q_i$.

$J_i =$ the gradient of *h (Q),* evaluated at $Q = Q_i$

$Q_i$ = a pervious flow solution around which the linearization is carried out.

$\Delta Q$ = flow correction for all links in the loop or path.

Introducing equation (3.11) into (3.6), for every loop or energy path of the network, taken one at a time:

$$\delta E_k = \sum_K H_i + (\sum_K J_i)\Delta Q_k \text{---(3.12)}$$

for all loop or path "k", k=1,2...NL,

where the summation are carried out over all the links is belonging to the k-the path.

Equation (3.12) can be solved for the scalar flow correction:

$$\Delta Q_k = \frac{(\delta E_k - \sum_k H_i)}{\sum_k J_i} \text{----(3.13)}$$

for all paths "k", k=1,2...NL

The flow correction $\Delta Q_k$ is computed for every loop or path and is applied to the entire link in the loop, in order to improve their closeness to the simultaneous fulfilment of equation (3.1) and (3.2).

This indeed the *Hardy-Cross* method (loop version) formulated in 1936 and extensively used for hand computation (and some simple computer application these today). Also this method is referred to as a single (path adjustment) algorithm.

On considering all the loop correction simultaneously (instead of one at a time as in the previous case). Equation (3.12) become a vectorial function and it must incorporate an additional term to cater for the flow correction made in neighbouring loops;

$$\delta E_k = \sum_K H_i + (\sum_K J_i)\Delta Q_k + \sum_K (J_i \Delta Q_n) \text{ ----(3.14)}$$

Equation (3.14) is valid for every loop or path "k" and the summation are carried out over all the links in that loop.

$\Delta Q_k$ refer to the loop correction of loop (k)

and $\Delta Q_n$ refer to the correction of the loop $n \neq k$ neighbouring the loop *k* with the sign of $\Delta Q_n$ being compatible with that of $\Delta Q_k$

Reordering equation (3.14) gives:

$$(\sum_K J_i)\Delta Q + \sum_K (J_i \Delta Q_n) = \delta E_k - \sum_K H_i \text{ --(3.15)}$$

The first term of the left side of the equation (3.15) caters for the flow correction over all the link in the path "*k*" while the second term of the equation include the effect of the flow correction in the neighbouring path. When equation (3.15) is written for all the path, the whole left hand side of the resulting system of equation can be expressed as the product of an (*NL*Nl* ) symmetric characteristic matrix by (*NL*1*) vector of flow correction $\Delta Q$. The right hand side become an (*NL *1*) vector of difference between available head and the total head losses in each path.

This is the so-called "simultaneous path adjustment" method and the system (13) is solved in a recursive fashion as many times as necessary to get the flow corrections smaller than a pre-established accuracy.

If instead of expressing equation (3.12) in term of d $Q$ is left as a function of the explicit difference $Q - Q_i$:

$$\delta E_k = \sum_K H_i + \sum_K J_i(Q - Q_i) \text{----}(3.16)$$

$$\delta E_k = \sum_K H_i + \sum_K (J_i Q) - \sum_K J_i Q_i \text{----}(3.17)$$

which when reordered gives:

$$\sum_K (J_i Q) = \sum_K (J_i Q_i - H_i) + \delta E_k \text{---}(3.18)$$

for all paths “k”, k=1,2...NL

When equation (3.18) is written for all the paths, the left side can be expressed as the product of non-symmetric (NL*NT) characteristic matrix by an (NT*1) vector of flows per link, while the right hand side is an (NL*1) vector.

The (l+f-1) equation (3.18) linearized energy equation has to be coupled to the (j) continuity equation (3.4) to form a set of p simultaneous linear equations in terms of the flow rate in each pipe. For the linear method an initial flow rate is needed but was not required to satisfy continuity according to Wood and Charles [13]. For this method an initial flow rate based on a constant flow velocity in arbitrary direction was assigned. The solution provided by the coupled systems of equations (3.4) and (3.18) has superior convergence properties than the original one, as recognised by Wood [15].

## 3.8.Gradient Algorithms

The main distinctive characteristic of this method from the rest of the existing methods is the fact that it is based on the gradient operator being applied over both the heads and the flows and not only to one of them. In order to find a solution of the system of partly linear and partly non-linear equation describing the network flow problem, the application of N-R technique in the space of both unknown pipe flow and unknown nodal heads, where the existence and unity of the solution can proved, leads to an extremely convergent scheme. R. Salgado,and Todini[27] showed the necessary conditions for the steady state flow are simply the simultaneous fulfilment of nodal balance and the head loss-flow relationship; see also Tondini and Pilati [28]for details on the derivation of the alternative gradient formulation. Figure 3.2 shows the main flowchart of the computer program developed for the global gradient algorithm.

## 3.9.Steady Incompressible Flow in Pipe Networks

Analysis and design of pipe networks create a relatively complex problem, particularly if the network consists of a lager number of pipes as frequently occurs in the water distribution system of large metropolitan area, or natural gas pipe network. Professional judgement is involved in deciding which pipes should be included in a single analysis.

Obviously it is not practical to include all pipes which deliver to all customers of a large city, even though they are connected to the total delivery system. Often only those main trunk lines which carry the fluid between separate sections of the area are included, and if necessary analysis of the networks within this section may be included. In water distribution system, the steady-state analysis is a small but vital component of assessing the adequacy of a network.

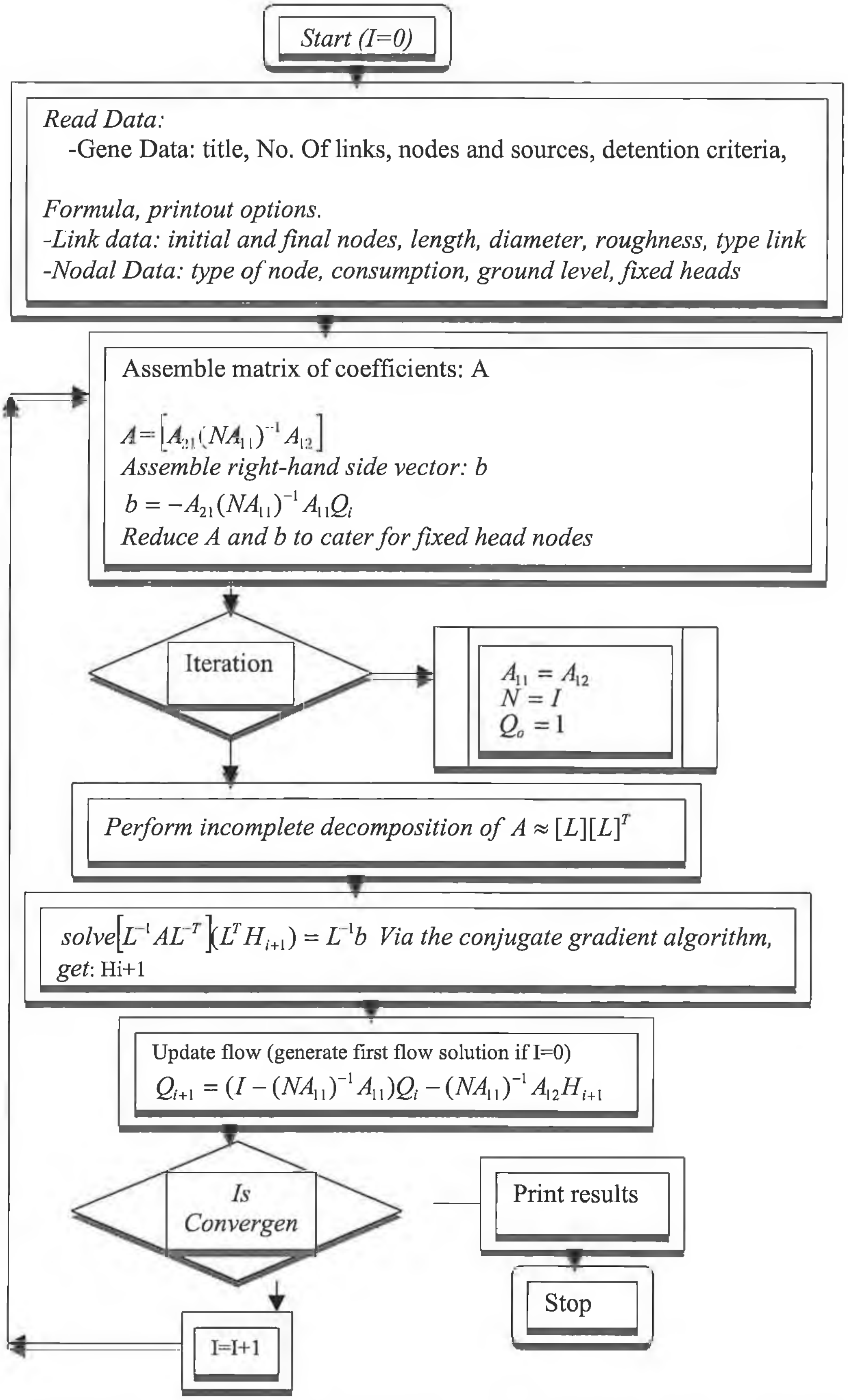

Start (I=0)
Read Data:
-Gene Data: title, No. Of links, nodes and sources, detention criteria,
Formula, printout options.
-Link data: initial and final nodes, length, diameter, roughness, type link
-Nodal Data: type of node, consumption, ground level, fixed heads
Assemble matrix of coefficients: A
$A=\left[A_{21}(NA_{11})^{-1}A_{12}\right]$
Assemble right-hand side vector: b
$b=-A_{21}(NA_{11})^{-1}A_{11}Q_i$
Reduce A and b to cater for fixed head nodes
Iteration
$A_{11}=A_{12}$
$N=I$
$Q_o=1$
Perform incomplete decomposition of $A\approx[L][L]^T$
solve$\left[L^{-1}AL^{-T}\right](L^TH_{i+1})=L^{-1}b$ Via the conjugate gradient algorithm, get: Hi+1
Update flow (generate first flow solution if I=0)
$Q_{i+1}=(I-(NA_{11})^{-1}A_{11})Q_i-(NA_{11})^{-1}A_{12}H_{i+1}$
Is Convergen
Print results
Stop
I=I+1

*Figure 3.2 flowchart of computer program GRAD (the global gradient algorithm)*

Such analysis is needed each time to changing patterns of consumption or delivery are significant or add-on feature, such as supplying new subdivision, addition of booster pumps, or storage tanks changing the system. In addition to steady flow analysis, studies dealing with unsteady flows or transient problem, operation and control, acquisition of supply, optimisation of networks performance against cost, and social implication should be given consideration but are beyond the scope of this study.

The simulation of the steady flow in pipe networks is based on the mathematical representation of the network components and their connectivity through an appropriate topological system of nodes and elements.

The behaviour of the network elements is simulated by applying the energy equation while, in order to simulate the interaction between the network elements, the continuity equation has to be satisfied at any network node. The aforementioned equation along with the boundary conditions constitutes a non-linear system of equations, which describe mathematically the behaviour of the whole piping system.

The steady state problem is considered solved when the flow rate in each pipe is determined under some specified patterns of supply and consumption. The supply maybe from reservoir storage tanks and/or pumps or specified as inflow or outflow at some point in the networks.

### 3.9.1.Types of Pipe Flow Problems

The hydraulic engineer is confronted with many problems in the planning, design and operation of pipe supply system. The problem can be divided into analysis and design types both for steady flow and unsteady flow.

The analysis of steady flows in simple pipes may be for rate (given the head loss) or for head loss (given flow rate). The same calculations apply to compound pipes (in parallel or series) although solution of more than one equation are then involved. When it comes to branched or looped networks more sophisticated methods become necessary.

The design problem is usually treated as a steady state problem that for known head and draw-off, the engineer has to select the pipe layout and diameter and reservoir location and size. The latter aspect, namely reservoir sizing is really unsteady state flow problem which may be often solved using steady-state equation. Multiplying draw-off rate by time may assess net outflow over the peak draw-off period.

For more rapid variation in flow (rigid column) surge theory or even elastic water hammer theory is necessary to determine heads and transient flows. Computer analysis is practically essential. Once to determine steady-state flows or head.

The design problem associated with unsteady flows is the determination of pipe thickness, and the operating rules for valves, pump…etc

### 3.9.2.Method of Solution

Where complex pipe networks are utilised for water distribution it is not easy to calculate the flow in each pipe or the head at each point. Even if the flow-head loss equation assumed is explicit for each given pipe length, diameter and roughness, the non-linear relationship between the head loss and flow makes calculations difficult. In un-looped tree-like networks the flow will be defined by the draw-off but if the pipe networks incorporated closed loops flow are unknown as well as head at the various nodes.

The complexity of the pipe networks, as well as the facilities available for computation, will dictate which method of analysis is to be utilised. Many of the following methods can be performed manually whereas computer are required for more complex methods, particularly where unsteady flow is involved

1. Equivalent pipes for compound pipes in series.
2. Equivalent pipes for complex pipes in parallel
3. Trial and error methods for multiple reservoir problems
4. Analytical solution of flow-head loss equation for compound pipes
5. Analytical solution of flow-head correction for pseudo-steady flow
6. Iterative node head correction for predominantly branched networks(by hand computer)
7. Iterative loop flow correction for looped networks (by hand or computer)
8. Simultaneous solution of the head loss equation for all pipes by matrix or iterative method
9. Linearization of head loss equation and iterative solution for head at nodes
10. Linearization of head loss equation and iterative solution for flow in pipes
11. Analytical solution of rigid column unsteady flow equation
12. Numerical solution of finite difference form of rigid column acceleration equation ,head loss equation and continuity equation
13. Graphical analyses for unsteady rigid column flow.
14. Graphical analysis for unsteady elastic water hammer
15. Finite difference and characteristic solution of differential water hammer equation-using computer. Valve, pump, vaporisation, release system and branches may be considered

### 3.9.3.General Equation

1. Continuity of flow at junction (net flow in draw-off must be zero)
2. Head difference between nodes equal to friction head loss in the pipes linking them. Minor losses and velocity head are generally neglected or included in the friction term, or an equivalent length pipe is added to the pipeline to allow for minor losses.
3. . Dynamic equation of motion – only where acceleration or deceleration of water is significant.

## 3.10.Reducing Complexity of Pipe Networks:

In general, pipe networks may include series pipes, parallel pipes, and branching pipes (i.e. pies that form the topology of a tree). In addition, elbow valves, meter, and other device which cause local disturbances and minor losses may exist in pipes. All of the above should be combined with or converted to an "equivalent pipe" in defining the networks to be analysed. The concept of equivalent is useful in simplifying networks. Methods of defining equivalent pipes for each of the above mentioned occurrence follows.

### 3.10.1.Series Pipes

It is possible to replace series pipes by an equivalent pipe. This can be done to simplify a system by reducing the number of pipe. It can also be used as a design aid where a good design can be obtained using pipes of any diameter and finally the equivalent series pipe can be sized (using available diameter) which has the same hydraulic properties wood [25]

The method for reducing two or more pipes of different size in series will be explained by reference to the diagram below. The same flow must pass through each pipe in series Fig 3.3. An equivalent pipe is a pipe, which will carry this rate and produce the same head loss as two or more pipes, or Jepson [8]

$$Hfe = \sum hfi \text{----}(3.19)$$

Expressing the individual head losses by the exponential formula gives,

$$KeQ^{ne} = K_1 Q^{n_1} + K_2 Q^{n_2} + \dots\dots = \sum K_i Q^{n_i} \text{-----}(3.20)$$

For network analysis ***Ke*** and ***ne*** are needed to define the equivalent pipes hydraulic properties. If the Hazzen-Willams equation is used, all the exponents *n*=1.852, and consequently

$$K_e = K_1 + K_2 + \dots\dots = \sum K_i \text{------}(3.21)$$

Or the coefficient ***Ke*** for the equivalent pipe equals the sum of ***K*** of the individual pipes in series.

Equation (3.21) can be written in different way for an equivalent length as.

$$\frac{L}{D_e^{4.87}}=\frac{L_1}{D_1^{4.87}}+\frac{L_2}{D_2^{4.87}}=L=L_1+L_2 \text{--------(3.22)}$$

where *L1* and *L2* are the lengths of pipes with diameter *D1* and *D2* respectively and *De* is the diameter of a pipe length l which is equivalent to the series pipe. In terms of the equivalent diameter the length of pipe of diameter D2 can be computed as

$$L_2=L\left[\frac{\left(\frac{D_1}{D_e}\right)^{4.87}-1}{\left(\frac{D_1}{D_2}\right)^{4.87}-1}\right] \text{---------(3.23)}$$

If the Darcy-Weishbach equation is used, the exponents ***n*** in Eq (3.20) will not necessarily be equal, but generally these exponents are near enough equal to that the ***ne*** for the equivalent pipe can be taken as the average of these exponents and Eq (3-21) used to compute ***Ke***.

### 3.10.2.Series Pipe Flow with Pump(s)

The solution of pipe flow problems involving pump normally requires the data from pump characteristic curves to be used. However, if a computer is used to solve these problems, such reading can no longer be done in this way. But the resolution of this problem is not difficult as part of the computer solution of this kind of problem, sufficient data are supplied to the program so that the head HP can be expressed as a polynomial in discharge that fits the pump-curve data.

Let the pump characteristic curve for the head hp be expressed by second-order polynomial *Hp=aQ+bQ+c*, in which the coefficients *a,b*,and *c* are determined by the use of three *Hp,Q* data pairs that bracket the expected range of operation of the pump. To obtain the coefficients, three equations are written by substituting reach data pair into the polynomial to obtain

$$aQ_1^2 + bQ_1 + c = H_{p_1} \text{---}(3.24)$$

$$aQ_2^2 + bQ_2 + c = H_{p_2}$$

$$aQ_3^2 + bQ_3 + c = H_{p_3}$$

In matrix equation (3.24) becomes

$$\begin{bmatrix} Q_1^2 \, Q_1 \, 1 \\ Q_2^2 \, Q_2 \, 1 \\ Q_3^2 \, Q_3 \, 1 \end{bmatrix} \begin{Bmatrix} a \\ b \\ c \end{Bmatrix} = \begin{Bmatrix} H_{p_1} \\ H_{p_2} \\ H_{p_3} \end{Bmatrix}$$

which can be solved in various ways to determine the coefficients.

An alternative approach is to use the Lagrangian interpolation. Lagrange's formula is

$$H_p = \frac{(Q-Q_2)(Q-Q_3)}{(Q_1-Q_2)(Q_1-Q_3)} H_{p_1} + \frac{(Q-Q_1)(Q-Q_3)}{(Q_2-Q_1)(Q_2-Q_3)} H_{p_2} + \frac{(Q-Q_1)(Q-Q_2)}{(Q_3-Q_1)(Q_3-Q_2)} H_{p_3} \text{ (3.25 )}$$

The head Hp is again expressed as quadratic equations in Q, but the terms are rearranged from the earlier approach. The coefficients a,b and c can be found by expanding the numerators. Letting

$$c_1 = \frac{H_{p_1}}{(Q_1 - Q_2)(Q_1 - Q_3)}$$

$$c_1 = \frac{H_{p_2}}{(Q_2 - Q_1)(Q_2 - Q_3)} \text{-----(3.26)}$$

$$c_1 = \frac{H_{p_3}}{(Q_3 - Q_1)(Q_3 - Q_2)}$$

it is shown that

$$a = c_1 + c_2 + c_3$$

$$b = -2[(Q_2 + Q_3)c_1 + (Q_3 - Q_1)c_2 + (Q_1 - Q_2)c_3] \text{--(3.27)}$$

$$c = Q_2Q_3c_1 + Q_3Q_1c_2 + Q_1Q_2c_3$$

Irrespective of which approach is used, the result can be interesting in a computer program so that the need to read from a pump characteristic curve during the solution can be avoided.

## 3.10.3.Parallel Pipes

It is often necessary to analyse a system with parallel lines. An existing system may have such lines and system improvement often take the form of lines laid parallel to the existing ones. In any case it may be desirable to use the concept of an equivalent pipe to represent parallel ones Fig 3.4. This concept is especially useful for system simplification or to check change of this type proposed for an existing system. In reality each parallel pipe adds one loop to the system and would require a change in the basic system data. However with nothing more than a pipe line data change the effect of a parallel pipe can be investigated if the concept of an equivalent pipe used.

The basic principle behind the concept of an equivalent pipe is to replace parallel pipes by a single pipe, which will transport the same total flow rate for the same head difference between the connecting junction.

$$Hf = hf_1 = hf_2 = hf_3 = \text{------------}(3.28)$$

The total flow rate will equal the sum of the individual flow rate or

$$Q = Q_1 + Q_2 = \sum Q_i \text{ ----}(3.29)$$

Solving the exponential formula ( $Hf = kQ^n$ ) for $Q$ and substituting into Eq.(3.29) gives

$$\left(\frac{Hf}{K_e}\right)^{\frac{1}{n_e}} = \left(\frac{Hf}{K_1}\right)^{\frac{1}{n_1}} + \left(\frac{Hf}{K_2}\right)^{\frac{1}{n_2}} + .... = \sum\left(\frac{Hf}{K_i}\right)^{\frac{1}{n_i}} \text{ ----}(3.30)$$

If the exponents are equal, as will be the case in using the Hazan-Williams equation, the head loss (*hf*)_may be eliminated from Eq (3.30) giving

$$\left(\frac{1}{K_e}\right)^{\frac{1}{n}} = \left(\frac{1}{K_1}\right)^{\frac{1}{n}} + \left(\frac{1}{K_2}\right)^{\frac{1}{n}} + .... = \sum\left(\frac{1}{K_i}\right)^{\frac{1}{n}}$$

the above can be written for an equivalent diameter for example if the subscript 1 represents a pipe and the subscript 2 represents an existing or proposed pipe parallel

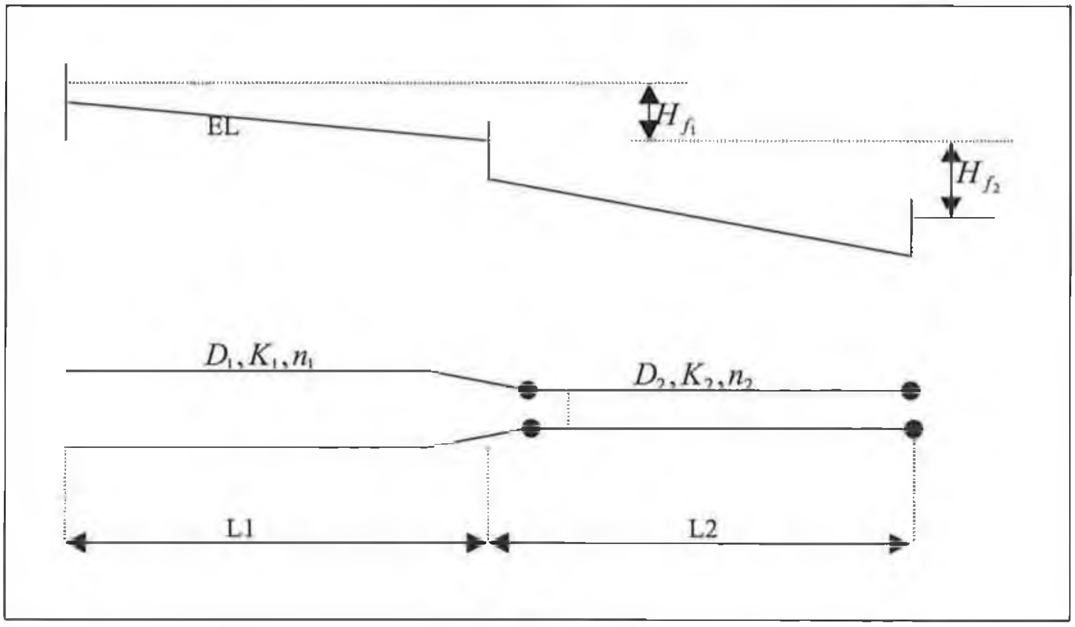


Figure 3.3 Series Pipes

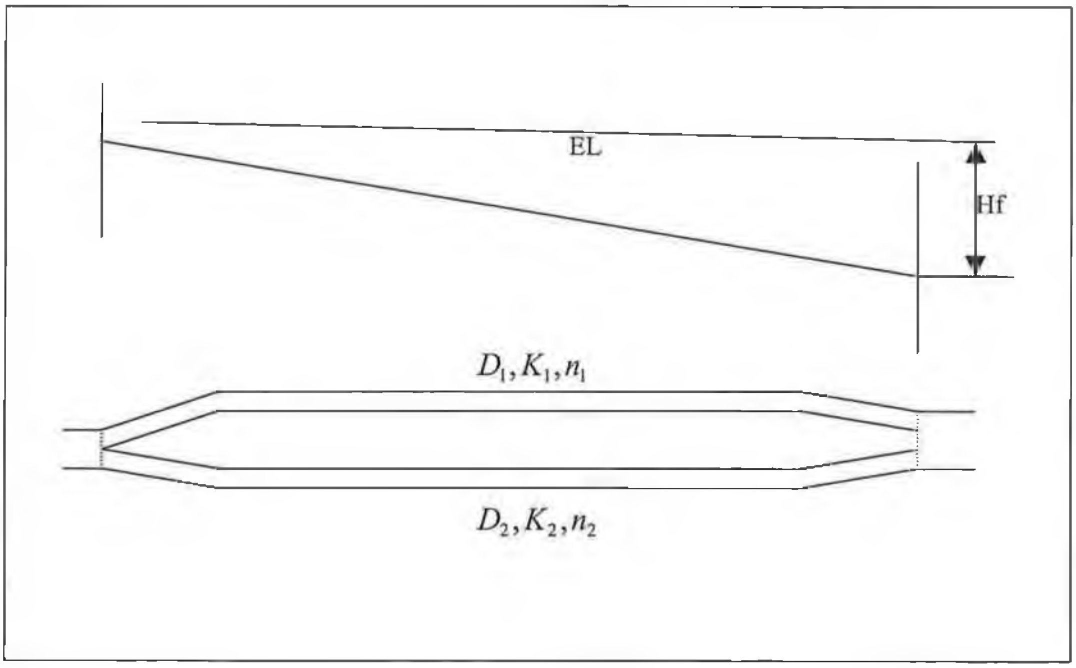


Figure 3.4 Parallel Pipes

to pipe the relationship for the diameter of the equivalent pipe obtained from the Hazan Williams equation is

$$D_e = \left[ \frac{C_1}{C_e} \left( \frac{L_e}{L_1} \right)^{.54} D_1^{2.63} + \frac{C_2}{C_e} \left( \frac{L_e}{L_2} \right)^{.54} D_2^{2.63} \right]^{.38} \text{----(3.32)}$$

where D, C and L are the diameter, the Hazan-Willamis roughness coefficient and length respectively and the subscript e represents the equivalent pipe. To use this expression, the equivalent length Le and the Hazan-Williams coefficient Ce are chosen as desired and the equivalent diameter De computed from the above expression. For example consider that an existing 1000'; 6" line is paralleled by a new 1100'; 10'' line the following data holds

C1=90 C2=130

D1=6''D2=10''

L1=100'L2=110'

If Le is taken as 1000' and Ce as 130 the equivalent diameter is computed to be De=10.45''. A pipe with this L, C and D is equivalent to the two pipes in parallel and will give the same total flow rate and the same junction pressure.

When the Darcy-Weishbach equation is used for the analysis, it is common practice to assume N as equal for all pipes and use Eq (3.32) to compute the ***Ke*** for the equivalent pipe.

### 3.10.4.Branching System

In a branching system a number of pipes are connected to the main to form the topology of tree. Assuming that flow is from the main into the smaller laterals it is possible to calculate the flow rate in any pipe as the sum of the downstream consumption or demand. If the laterals supply water to the main, as in a manifold, the same might be done. In either case by proceeding from the outermost branches toward the main or "root of the three" the flow rate can be calculated, and from the flow rate in each pipe the head loss can be determined using the Darcy-Weichbaher or Hazan-Williams equation. In analysing a pipe network containing a branching system, only the main is included with the total flow rate specified by summing from the smaller pipes, upon completing the analysis the pressure head in the main will be known. By subtraction individual head loss from the known head, the head or the pressures at any point throughout the branching system can be determined

## 3.11.Equations Describing Steady Flow

### 3.11.1System of Q-Equations

The analysis of flow in pipe networks is based on the continuity and energy equation laws as described in chapter one. To satisfy continuity, the mass, weight, or volumetric flow rate into a junction must equal the mass, weight, or volumetric flow rate out of a junction. If the volumetric flow rate is used this principle, as discussed in chapter 2, can be expressed mathematically as,

$$\sum Qout - \sum Qin = C \text{--------------}(3.33)$$

in which C is the external flow at the junction (commonly consumption or demand). C is positive if flow is into the junction and negative if it is out from the junction.

If a pipe network contain (J) junctions (also called nodes) and all external flows are known then *(J-1)* independent continuity equations of the form of Eq.(3.33) can be written. The last, or the (J) continuity equation, is not independent; that is, it can be obtained from some combination of the first *(J-1)* equation. Each of these continuity equations is linear, i.e., *Q* appears only to the first power.

In addition to the continuity equation, which must be satisfied, the energy principle provides equation, which must be satisfied. These additional equation are obtained by noting that if one adds the head losses around a closed loop, taking into account whether the head loss is positive or negative, that upon arriving at the beginning point the net head losses equal zero. Mathematically, the energy principle gives *L* equations

$$\sum_{L}^{I} Hfl = 0 \text{--------}(3.34)$$

$$\sum_{L}^{L} Hlf = \Delta WS$$

When the head losses are expressed in terms of the exponential formula, then these equations take the form

$$\sum K_i Q_i^n = 0 \text{------}(3.35)$$

$$\sum K_i Q_i^n = \Delta WS$$

In which the summation includes the pipes that form the loop. If the direction of the flow should be in the opposite direction of that was assumed when the entire loop equations were written, such as Qi become negative, then there are two alternatives: one is to reverse the sign in front of this terms, i.e., correct the direction of the flow. The second, which is generally preferred when writing a program to solve these equations, is to rewrite the equations as follows:

$$\sum K_i Q_i |Q_i|^{n-1} = 0 \text{----}(3.36)$$

$$\sum K_i Q_i |Q_i|^{n-1} = \Delta WS$$

A pipe network consisting of ***(J)*** junctions and ***(L)*** non-overlapping loops and ***(N)*** pipes will satisfy the equation

$$N = (J-1) + L \text{-----------}(3.37)$$

(If all of the external flows are not known, then all junction equations are independent and available for use as will be discussed in the next chapter).

Since the flow rate in each pipe can be considered unknown, there will be *(N)* unknowns. The number of independent equations, which can be obtained for a network, as described above are (J-1+L). Consequently the number of independent equations is equal in number to the unknown flow rates in the *N* pipes. The (j-1) continuity equation are linear and the L energy (or head loss) equations are non-linear. Since large networks may consist of hundreds of pipes, systematic methods, which utilise computers, are needed for solving this system of simultaneous equations. The solution of the equations can be verified by substituting into each of the equation. It is relatively easy to determine flows in individual pipes, which also satisfy the (J-1) continuity equations. However, the correct flow rates, which simultaneously satisfy the L energy equations, are virtually impossible to obtain by trial and error if the system is large.

After solving the system of equations for the flow rate in each pipe, the head losses in each pipe can be determined. The HGL-elevation at the nodes or at any point along the pipe can be found by starting at a known HGL-elevation and repeatedly applying the exponential formula for head loss to each pipe. By subtracting the head loss from the head at the upstream junction, plus accounting for difference in elevation, if this were the case. If the network is branched system, then the Q-equations consist of only junction continuity equations. These can be solved, given the discharge in every pipe, with a linear algebra solver. Thereafter the individual heads are computed from the head loss equation for each pipe. In some problems the external flows may not be known. Rather the supply of water may be from reservoirs and/or pumps. The amount of flow from these individual sources will not only depends upon the demands but also will depend upon the head losses throughout the system

## 3.11.2.System of H-equations

If the head (either the total head or the piezometric head, since the velocity head is generally ignored in determining heads or pressure in pipe networks) at each junction is initially considered unknown instead of the flow rate in each pipe, the number of simultaneous equations which must be solved can be reduced in number. The reduction in number of the equations, however, is at the expense of not having some linear equations in the system.

To obtain the system of equations, which contain the head at the junction of the networks as unknown, the (J-1) independent quantity equations are written as before. Thereafter the relationship between the flow rate and head loss is substitute into the continuity equation. In writing these equations it is convenient to use a double subscript for the flow rate. These subscripts correspond to the junction at the ends of the pipe. The first subscript is the junction number from which the flow comes and the second is the junction number to which the flow is going. Thus *(Q12)* represents the flow in the pipe-connecting junction (1) and (2) assuming the flow is from junction (1) to junction (2) see Figure [3.5]. If the flow is actually in this direction *(Q12)* is positive and *(Q21)* equals minus *(Q12)*. Solving for *(Q)* from the exponential formula (using the double subscript notation) gives

$$Q_{ij} = \left(\frac{Hl_{ij}}{K_{ij}}\right)^{\frac{1}{n_{ij}}} = \left(\frac{H_i - H_j}{K_{ij}}\right)^{\frac{1}{n_{ij}}} \text{--------(3.38)}$$

If equation (3.38) is substituted into junction continuity equation (3.33), the following equation results:

$$\left\{\sum\left(\frac{H_i - H_j}{K_{ij}}\right)^{\frac{1}{n_{ij}}}\right\}_{out} - \left\{\sum\left(\frac{H_i - H_j}{K_{ij}}\right)^{\frac{1}{n_{ij}}}\right\}_{in} = C \text{ -----(3.39)}$$

Upon writing an equation of the form Eq (3.39) at *(j-1)* junction, a system of *(j-1)* non-linear equations are produced.

As an illustration of this system of equation with the heads at the junction as the unknowns, consider the simple one loop network fig 3.5 which consists of three junction and three pipes. In this network two independent continuity are available and consequently the head at one of the junction must be known. In this case at [1] the two-continuity equations are

$$Q_{12} + Q_{13} = Q_{J_1} + Q_{J_3}$$

$$Q_{21} + Q_{23} = -Q_{J_2}$$

Although in the second equation the flow in pipe 1-2 is towards the junction, the flow rate *(Q21)* is not preceded by a minus sign since the notation 2-1 takes care of this. Alternatively the equation could have been written at junction 2 and 3 instead of 1- and 2. Substituting Eq. (3.38) into these continuity equation gives the following two equation to solve for the heads, *H2* and *H3* (*H1* is known and the subscript of the H'S denote the junction number);

$$\left\{\frac{H_1 - H_2}{K_{12}}\right\}^{\frac{1}{n_{12}}} + \left\{\frac{H_1 - H_3}{K_{13}}\right\}^{\frac{1}{n_{13}}} = C_2 + C_3$$

$$-\left\{\frac{H_1 - H_2}{K_{12}}\right\}^{\frac{1}{n_{12}}} + \left\{\frac{H_2 - H_3}{K_{23}}\right\}^{\frac{1}{n_{23}}} = -C_2 \text{--(3.40)}$$

Since a negative number can not be raised to a power, a minus sign must precede any term in which the subscript notation is opposite to the direction of flow, i.e. the second form of equation as given in parentheses is used. System of these equations will be referred to as H-equation.

Upon solving this non-linear system of equations, the pressure at any junction (*J)* can be computed by subtracting the junction elevation from the head (*hj)* and then multiplying this difference by or the specific weight of the fluid. To determine the flow rates in the pipes of the network, the now known heads are substituted into Eq. (3.38).

### 3.11.3.System of $\Delta Q$ -Equations

Since the number of junction minus 1 (J-1) will be less in number than the number of pipes in a networks by the number of loops L in the networks, the last set of h equations will generally be less in number than the system of q equations. This reduction in number of equations is not necessarily an advantage since all of the equations are non-linear and may contain many terms. These equations consider the loop corrective discharges or $\Delta Q$ 's as the primary unknowns. These corrective discharges or $\Delta Q$'s will be determined from the energy equations that written for NL loops in the networks, and thus NL of these corrective discharge equations must be developed. To obtain these equations, the discharge in each pipe of the network is replaced by an initial discharge, denoted by $Q_{oi}$, plus the sum of all of the initially unknown corrective discharge that circulate through the pipe I or

$$Q_i = Q_{oi} + \sum \Delta Q_k \text{ ----(3.41)}$$

in which the summation includes all of the corrective discharges passing through pipe i. The initial discharges $Q_{oi}$ must satisfy all of the junction continuity equations. It is not difficult to establish the initial discharge in each pipe so that the junction continuity equations are satisfied. However, these initial discharges usually will not satisfy the energy equations that are written around the loops of the network.

Equation 3.37 is based on the fact that any adjustment can be added (accounting for sign) to the initially assumed flow in each pipe in a loop of the network without violating continuity at the junction. It is very important to understand the validity of this decomposition; it may help to note that any $\Delta Q$ entering the junction as it flows around a loop must leave that junction, and vice versa see Fig.3.6. Because of this fact, we decide to establish NL energy loop equations around the NL loops of the network, in which each initial discharge plus the sum of the corrective loop discharges $\sum \Delta Q$ is used as the discharge.

The junction continuity equations are satisfied by the initial discharge $Q_{oi}$ and are not a part of the system of equations. Th corrective discharges can bee chosen as positive if these circulate around the loop either in the clockwise or counter-clockwise

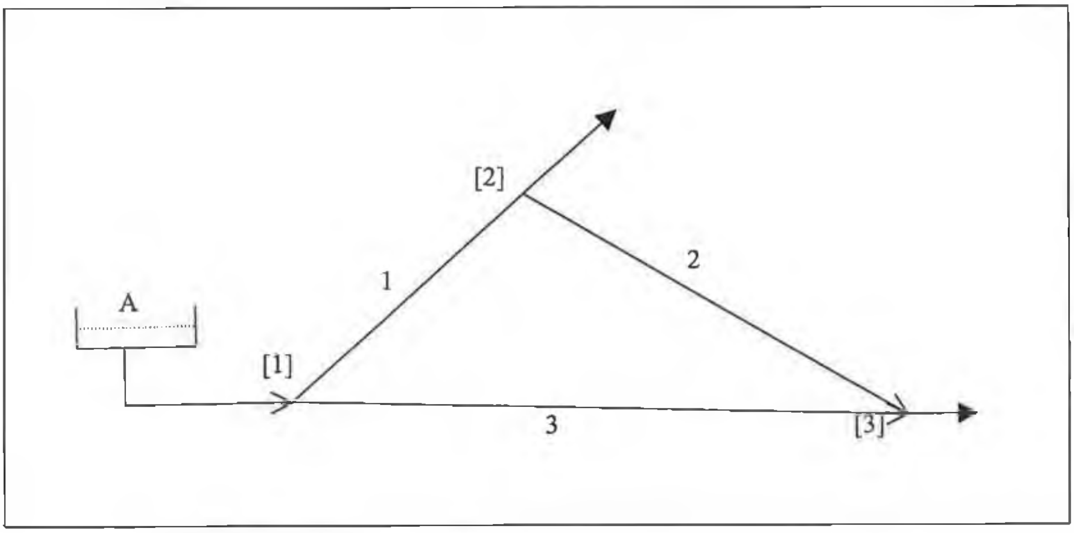


Figure 3.5 a network with three pipes and three junctions.

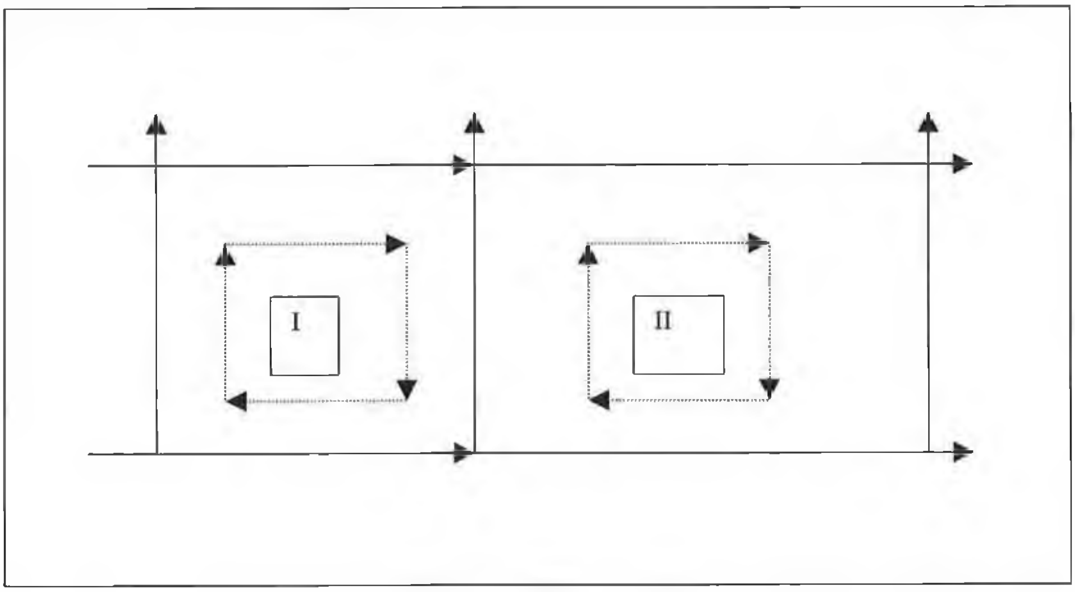


Figure 3.6 a two-loop portion of a network

direction. It necessary to be consistent within any one loop, but the sign convention may change from loop to loop, if desired. A corrective discharge adds to the flow $Q_{oi}$ in the pipe I if it is in the same direction as the pipe flow, and it subtracts from the initial discharge if it is in the opposite direction.

To summarised how the $\Delta Q$ are obtained, replace the Q's in the energy loop equations, Eq 3-34 and 3-35, by

$$Q_i = Q_{oi} \pm \sum \Delta Q_k \text{ ----(3.42)}$$

Here the summation includes all corrective discharges which pass through pipe I, and the plus sign is used if the net corrective discharge and pipe flow are in the same directions; otherwise the minus sign is used before the summation. Thus Eqs 3-34 3-35 become

$$\sum K_i \{Q_{oi} \pm \sum \Delta Q_k\}^{n_i} = 0 \qquad \text{(Head loss around loop I)}$$

$$\sum K_i \{Q_{oi} \pm \sum \Delta Q_k\}^{n_i} = \Delta WS \text{ -------(3.43)}$$

To automate the choice of sign, these equations can be rewritten as

$$\sum K_i \{Q_{oi} \pm \sum \Delta Q_k\} \left| Q_{oi} \pm \sum \Delta Q_k \right|^{n_i - 1} = 0$$

$$\sum K_i \{Q_{oi} \pm \sum \Delta Q_k\} \left| Q_{oi} \pm \sum \Delta Q_k \right|^{n_i - 1} = \Delta WS \text{ ---(3.44)}$$

# 3.12.Solving Networks Equations

## 3.12.1 Newton Method for Large Systems of Equation

These are System of algebraic equations to describe the relation between the discharges, pressure, and other variables and parameters in a pipe network. Many of the equation in these systems of equations are non-linear. A good method for solving non-linear equations is therefore needed. Numerous methods exist, but the Newton method is the methods of choice her. Its application to the solution of the Q-equations, the H-equations and the $\Delta Q$-equations will be discussed in this section. To treat the unknown discharges (when using the Q-equation, the unknown heads (when using the H-equation), and the unknown corrective loop discharges (when using the $\Delta Q$-equations) in a uniform way, the primary unknown variable in this section will be called the vector $\{x\}$.

The Newton iterative formula for solving a system can be written as

$$\{x\}^{(m+11)} = \{x\}^{(m)} - [D]^{-1}\{F\}^{(m)} \text{---(3.45)}$$

Here $x$ is an entire column vector $\{x\}$ of unknowns, $\{F\}$ is an entire column vector of equations, and $\{D\}^{-1}$ is the inverse of matrix $\{D\}$, which is the Jacobian. The Jacobian occurs in several applications in mathematics, and it represents the following matrix of derivatives:

$$[D] = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & . & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & . & \frac{\partial F_2}{\partial x_n} \\ . & . & . & . \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & . & \frac{\partial F_n}{\partial x_n} \end{bmatrix} \text{---(3.46)}$$

Likewise $\{x\}$ and $\{F\}$ are actually

$$\{x\} = \begin{Bmatrix} x_1 \\ x_2 \\ . \\ x_n \end{Bmatrix} \qquad \{F\} = \begin{Bmatrix} F_1 \\ F_2 \\ . \\ F_n \end{Bmatrix} \text{---(3.47)}$$

Eq. (3.45-3.48) indicates that the Newton methods solve a system of non-linear equations by iteratively solving a system of non-linear equations because $\{D\}^{-1}$ $\{F\}$ represents the solution of the linear system of equations

$$[D]\{z\} = \{F\} \text{---(3.48)}$$

That is, the vector that is subtracted from the current estimate of the unknown vector $\{x\}$ in Eq3.45 is the solution $\{z\}$ to the linear system of equations that is Eq3.48. In practice are therefore sees that he Newton method solves a system of equations by iterative formula

$$\{x\}^{(m+1)} = \{x\}^{m} - \{z\} \text{---(3.49)}$$

Where $\{z\}$ is the solution vector that is obtained by solving $[D]\{z\} = \{F\}$? If the system should actually contain only linear equations then first iterative will produce the exact solution.

The development of Eq. (3-45.48.49) follows. one begins by using a multidimensional Taylor series expansion to evaluate the individual equations Fi in the neighbourhood of an initial estimated solution that are called $\{x\}$ which are presumed to be near the actual solution:

$$\begin{aligned}
F_1^{(m+1)} &= F_1^{(m)} + \frac{\partial F_1}{\partial x_1}\Delta x_1 + \frac{\partial F_1}{\partial x_2}\Delta x_2 + \ldots\ldots + \frac{\partial F_1}{\partial x_n}\Delta x_n + O(\Delta x^2) = 0 \\
F_2^{(m+1)} &= F_i^{(m)} + \frac{\partial F_2}{\partial x_1}\Delta x_1 + \frac{\partial F_2}{\partial x_2}\Delta x_2 + \ldots\ldots + \frac{\partial F_2}{\partial x_n}\Delta x_n + O(\Delta x^2) = 0 \qquad \text{-----(3.50)} \\
&\cdot \\
F_n^{(m+1)} &= F_n^{(m)} + \frac{\partial F_n}{\partial x_1}\Delta x_1 + \frac{\partial F_n}{\partial x_2}\Delta x_2 + \ldots\ldots + \frac{\partial F_n}{\partial x_n}\Delta x_n + O(\Delta x^2) = 0
\end{aligned}$$

When one uses matrix notation and makes the substitution $\Delta x_i = x_i^{(m=1)} - x_i^{(m)}$, this system of equations becomes

$$\begin{Bmatrix} F_1 \\ F_2 \\ \cdot \\ F_n \end{Bmatrix}^{(m)} + \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdot & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdot & \frac{\partial F_n}{\partial x_n} \\ \cdot & \cdot & \cdot & \cdot \\ \frac{\partial F_n}{\partial x_1} & \frac{\partial F_n}{\partial x_2} & \cdot & \frac{\partial F_n}{\partial x_n} \end{bmatrix} \begin{Bmatrix} x_1^{(m+1)} - x_1^{(m)} \\ x_2^{(m+1)} - x_2^{(m)} \\ \cdot \\ x_n^{(m+1)} - x_n^{(m)} \end{Bmatrix} = 0 \text{---(3.51)}$$

Which can be written compactly as $\{F\}^{(m)} + [D]^{(m)}(\{x\}^{(m+1)} - \{x\}^{(m)} = \{0\}$ and solved for $\{x\}^{(m+1)}$ to produce Eq. (3.45).

# CHAPTER FOUR

# SIMULATION

## 4.1.Introduction

The purpose of a water distribution networks is to supply system user with the amount of water and to supply it within adequate pressure. A system may be subject to a number of different loading conditions, such as fire demand at different nodes, peak daily demands, a series of patterns varying through a day or a critical load when one or more pipes are broken. A loading condition is defined as pattern nodal demand Fig 4.1-4.2 shows example networks for two loading conditions. In order to insure a design is adequate, a number of the critical conditions must be considered during the design process. The ability to operate under a variety of load patterns has been equated with a reliable networks Templeman,[29].

To meet the pressure requirements and varying demands, the designer can incorporate a number of components into the network. Pipes, whose function is to convey water between nodes, are the most obvious. The cost of the pipe is related to the type of the material and its diameter. The design of a water distribution system can be separated into two parts, the layout and design. The layout problem for a water distribution system is to select the sites, given a set of candidate location for different components. The design problem, on the other hand, is to determine the optimum sizes of the

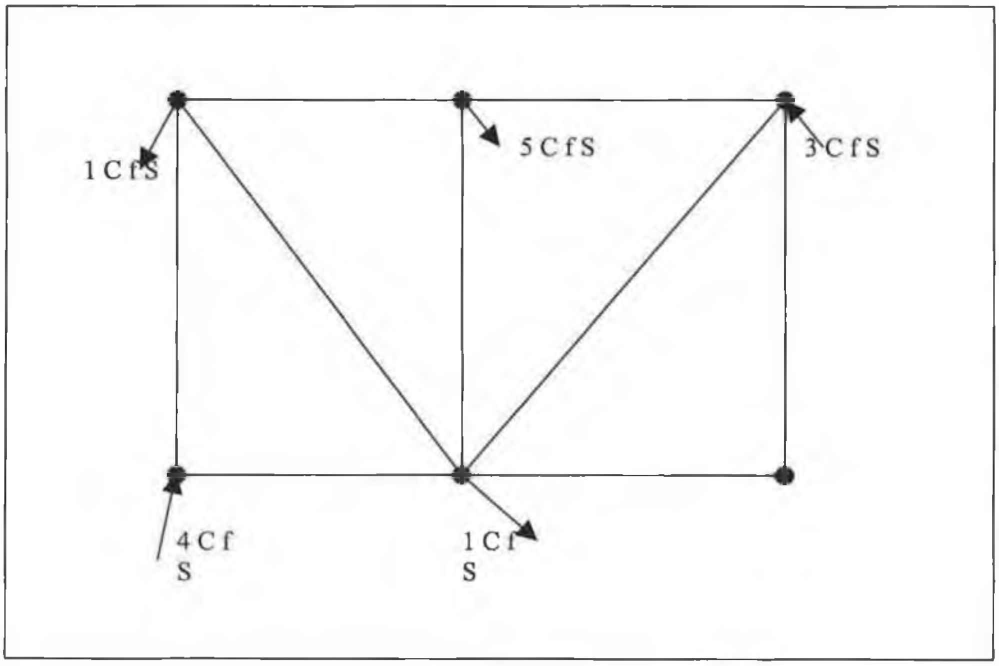


Figure 4.1 pipe system

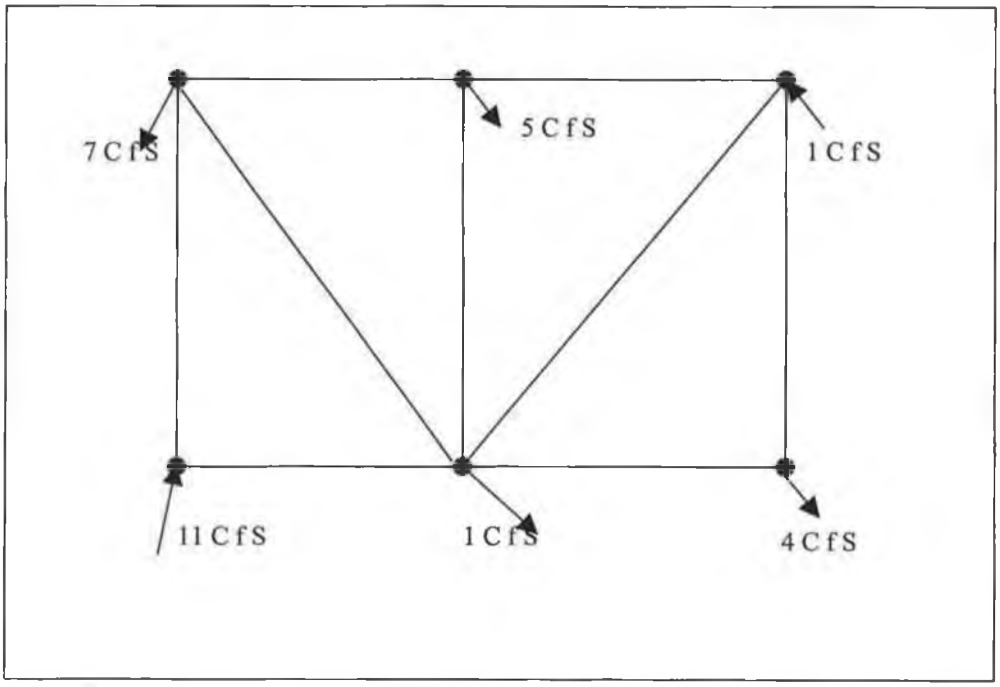


Figure 4.1.b-pipe system

components, for example, for pipe diameter, given the location Kevin E. Lansey and Larry.W. Mays[30]

In addition to pipes, pumps can be included in the system at locations throughout the networks. Pumps are comprised of an initial cost based on the size, and an energy cost, which is a function of the amount of water pumped. The designer may choose to use a single pump or a group of pumps working in series or in parallel, depending upon the demand and reliability requirements. Tanks are also useful in a networks for storage in emergency condition, to help smooth the peak demands throughout a day and to make use of energy price difference. The price of a tank is a function of the volume and elevation, in the case of elevated tanks. Both regulating valve and control valve may also be incorporated.

## 4.2.Pipe System Characteristics

It is necessary to describe the feature of the piping system using data, which assigns numerical value to the pertinent system characteristics. Parts of this data refer to as physical characteristics of the pipe system components and the rest to pressure and flow requirements imposed on the system. Also a general description of pipe system configuration and pipe system parameters which require data input. This section includes a general description of pipe system configuration and pipe system parameter which require data input.

## 4.3.Pipe System Geometry

The principal element in the pipe system are pipe sections which are constant diameter sections which can contain fittings such as bends, valves and pump as shown in Fig.4.2. The end points of pipe section are called nodes and are classified either as junction nodes or fixed grade nodes (FGN). These are shown in Fig.4.4 and Fig.4.5.

### 4.3.1.Junction Node

A node where two or more pipes meet or where flow put in or removed from the system .If a pipe diameter change occurs at a component such as a valve or a pump, this point is a junction node.

### 4.3.2.Fixed Grade Nodes (FGN)

A node in the system where both the pressure and elevation (or hydraulic grade lines) are known is usually connected to a storage tank or reservoir or a source or discharge point of specific pressure. Each system must have at least one fixed grade node.

### 4.3.3.Primary Loop

A close pipe circuit is one with no closed pipe circuits contained within it. When the junction primary loops, and fixed grade nodes are properly identified as described above the following relation holds for all pipe system. Wood and rays [31]

$$P = J + L + F - 1 \text{------}(4.1)$$

where: -

- **P**= number of pip sections
- **J**= number of junction nodes
- **L**= number of primary loops
- **F**= number of fixed grade nodes

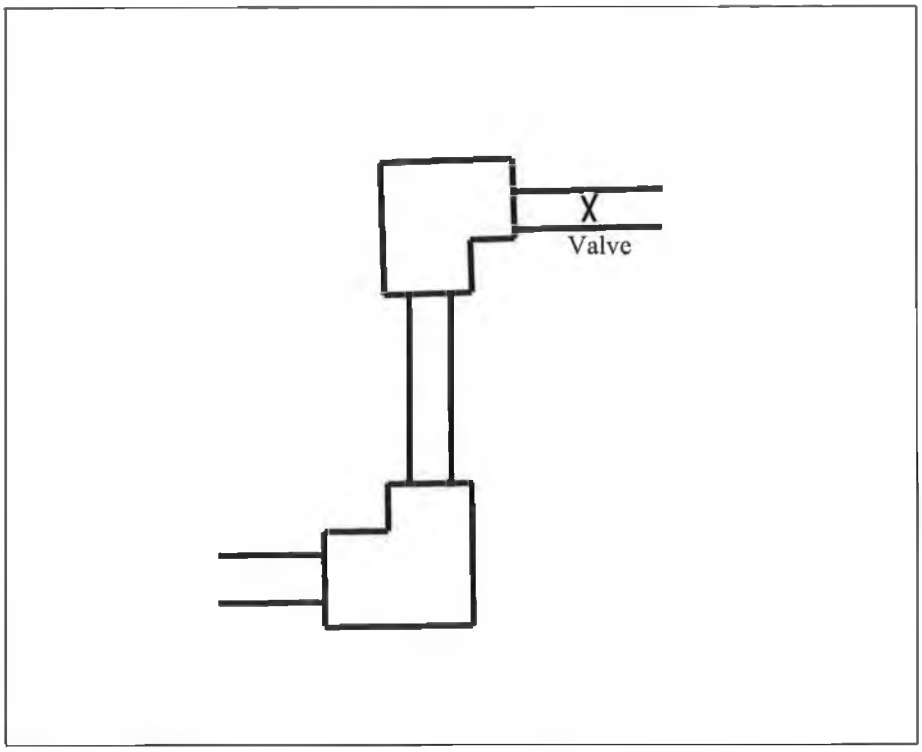


Figure 4.2 Pipe Section

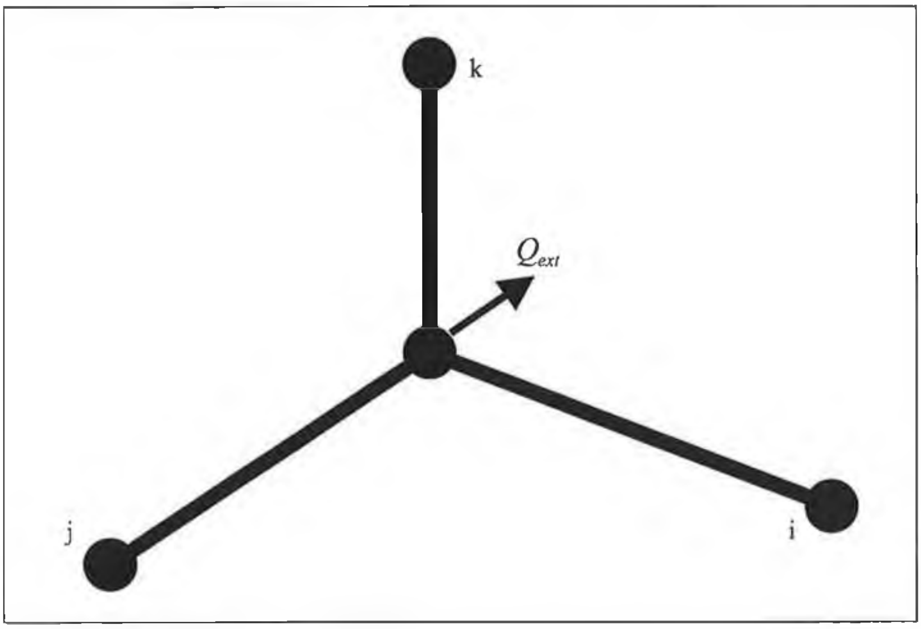


Figure 4.3 Junction Node

In addition, open loops (pseudo-loops) can be identified in a pipe system. An open loop is defined as a non-intersection path of pipes that does not contain a closed path between any two fixed-grade nodes. It follows from this definition that in any connected network with Nf fixed-grade nodes, the number of open loops NE equals (F-1) Paul F. Boulos,Don J. Wood-[32]. A proper identification of pipe section, junction nodes, a fixed grade node and primary loop is useful to assure a proper system description. Prior to coding data for a pipe network the pipe sections should be numbered and counted, and the junction nodes numbered and counted, and the fixed grand nodes labelled usually by A, B, C, …etc. Before proceeding Eq. (4.1) should be verified. Fig 4.6 illustrates this procedure. It shows the number of pipe sections and junction nodes and labelling the fixed grade node and a verification of Eq (4.1) is made. Although non-consecutive numbering is allowed, consecutive numbering (pipe section 1 to *p* and junction 1 to j) is suggested to help minimise any possible mistakes.

## 4.4.Pipe System Components

Data regarding the physical characteristics of the components in the pipe system must be obtained prior to making a computer analysis. A general description of the components, which are incorporated in the program and the necessary data, follows:

### 4.4.1.Pipe Section

The total length, inside diameter and the roughness of each pipe section must be in put as data .The designation of a pipe roughness depends on the head loss equation used. Estimating the head loss due to the friction in closed pipes is an important task in the solution of many practical problems in different branches of the engineering profession Giles et al., [33]. There are several equations, which are often used to

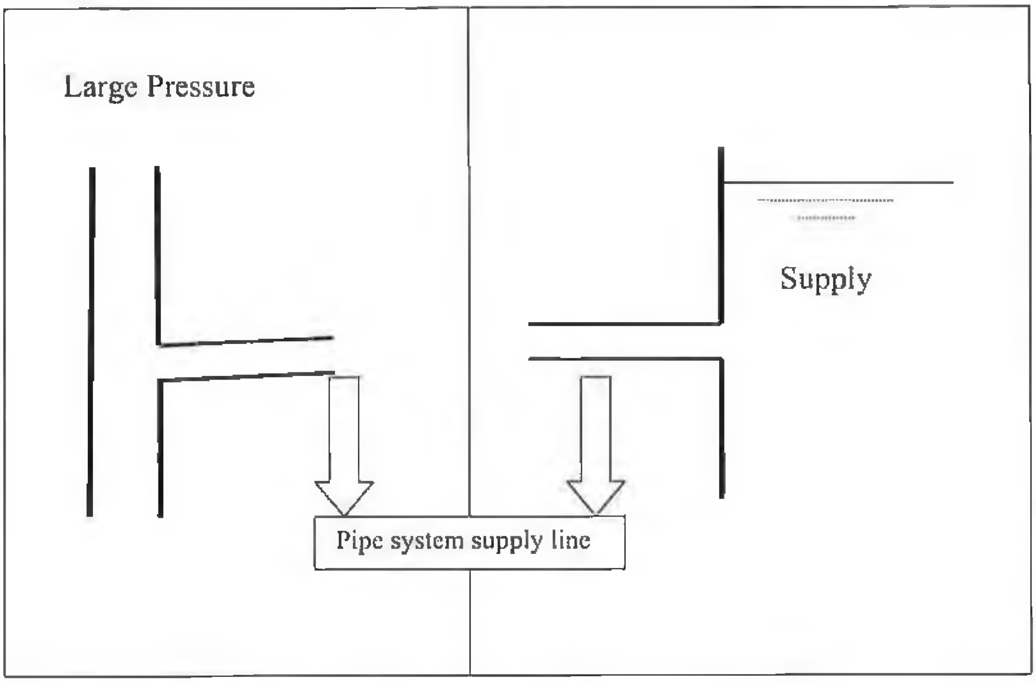


Figure 4.5 Fixed Grade Nodes

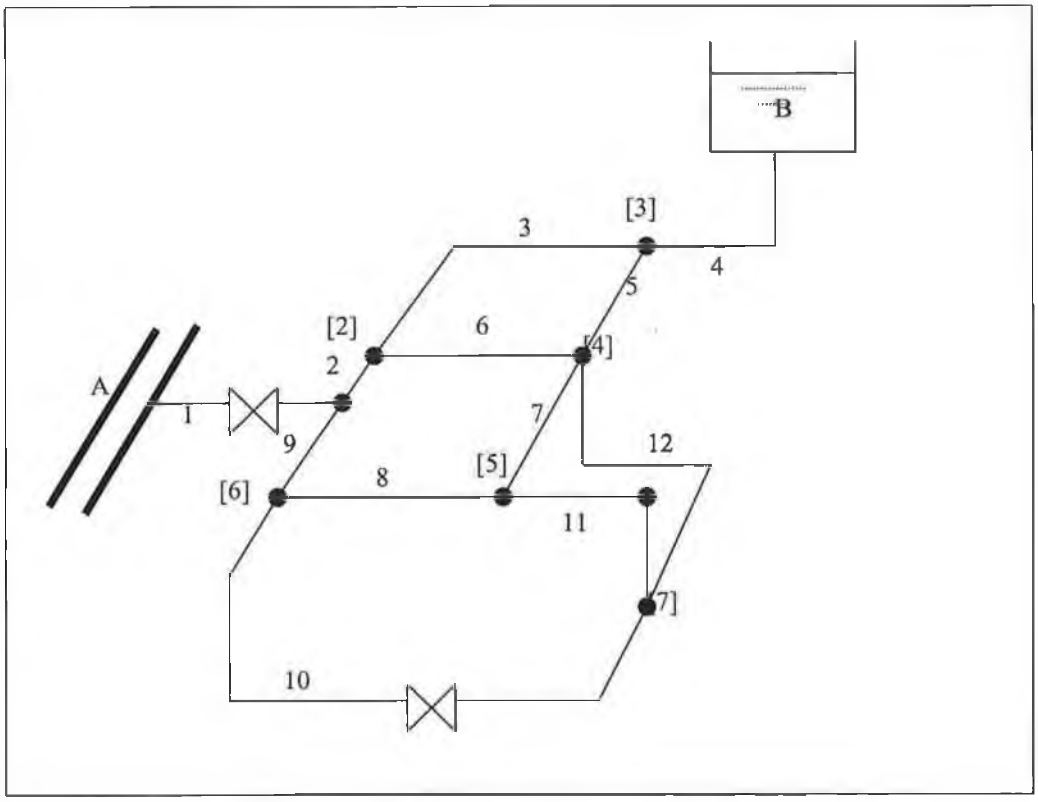


Figure 4.6 sample pipe system demonstrating

evaluate the friction head loss (i.e.conversion of energy per unit weight into a non-recoverable form of energy). Because most users are primarily interested in water distribution system, the most widely used such equation is the Hazen-Williames Equation, which was developed primarily for this purpose and is normally used to compute line losses.

$$Q = 1.318Chw * A * R^{0.63} * S^{0.54} \text{--------(4.2)}$$

$$Q = 0.849 * Chw * A * R^{0.63} * S^{0.54} \text{--------------SI}$$

in which

- ***Chw*** is the *H-W* roughness coefficient.
- S is the slop of the energy line and equals *HF/L.*
- R is the hydraulic radius defined as the cross-sectional area divided by the wetted perimeter, ***P*** and for pipes equals ***D/4.***

If the head loss is desired with ***Q*** known the *H-W* equation for pipe can be written as.

$$Hf = \frac{4.73l}{Chw^{1.852} d^{4.82} Q^{1.852}} \text{-------(4.3)}$$

with (d) and (l) in feet or

$$Hf = \frac{8.52 * 10^5 * L}{Chw^{1.852} d^{4.87} Q^{1.852}} \text{----------}$$

with (d) in inches and (l) feet

$$Hf = \frac{10.49 * L}{Chw^{1852} d^{4.87} Q^{1.852}} \text{---------SI}$$

If this expression is to be employed, the roughness coefficient depends on the type and condition of each pipe and Table 4.1 gives the Hazen-Williames Equation along with some representative value for this coefficient. However, the variation of age depends somewhat on the location of water distribution system and sometimes-field tests are required to obtain reliable values of the Hazen-Williames Roughness Coefficient for old pipes.

Another method, which is the most fundamental sound method for computing the head loss, is by mean of the Darcy-Weisbach equation Dake, [34]; Jepson [35] and Grant, [36]. Darcy-Weisbach equation can be written as:

$$Hf = \frac{\Delta p}{\gamma} = f\frac{L}{d}\frac{V^2}{2g} \text{-----------}(4.4)$$

in which:

- *f*: - Is a dimensionless friction factor whose determination is described in the following pages.
- d: - is the diameter
- L: - is the length of the pipe.
- V: - is the velocity of flow.
- g: - is the acceleration of gravity.

The friction factor, f, is a function of the Reynolds number, *Re*, and relative roughness of the pipe (i.e. e/D, where e is the pipe roughness). The Reynolds number is a dimensionless term representing the ratio of inertial to viscous forces Giles et al., [33] that is defined as:

Table 4.1 values of C in Hazen-Wlliams Equation

| Type of pipe | Condition | | C |
|---|---|---|---|
| Cast iron | New | all sizes | 130 |
| | | 12’’+ over | 120 |
| | 5 years old | 8’’ | 119 |
| | | 4’’ | 118 |
| | | 24’’ | 113 |
| | 10 years old | 12’’ | 111 |
| | | 4’’ | 107 |
| | | 24 + over | 100 |
| | 20 years old | 12’’ | 96 |
| | | 4’’ | 89 |
| | | 30” + over | 90 |
| | 30 years old | 16’’ | 87 |
| | | 4’’ | 75 |
| | | 30” + over | 83 |
| | 40 years old | 16’’ | 80 |
| | | 4’’ | 64 |
| | | 40’’ | 77 |
| | 50 years old | 24’’ | 74 |
| | | 4’’ | 55 |
| Welded steel | Value of C the same as for cast-iron pipes, | | 5 years older |
| Riverted steel older | Value of C the same as for cast-iron pipes, | | 10 years |
| Wood stave | Average, regardless of age | | 120 |
| Concrete | Large sizes, good workmanship, steel forms | | 140 |
| Concrete lined | Large sizes, good workmanship, wooden forms | | 120 |
| | Centifugally spun | | 135 |
| Vitrified | In good condition | | 110 |
| Plastic or Drawn Tubing | | | 150 |

$$\mathrm{Re} = \frac{VD}{\nu} \text{------------}((4.5))$$

where is $\nu$ the kinematics viscosity of the fluid (*L2/T*)

The relationship between f, on the one hand, and Re and *e/D*, on the other hand, varies in complexity depending on the flow regime. While f is only a simple and direct function of Re for laminar flow, the equation for estimating f in the turbulent flow region is more complex. According to Giles et al. [33], these relationships are, respectively:

$$f = \frac{64}{\mathrm{Re}} \, for \, \mathrm{Re} < 2100 \text{-----}(4.6)$$

$$\frac{1}{\sqrt{f}} = -2\log 10\left[\frac{e}{3.7D} + \frac{2.51}{\mathrm{Re}\sqrt{f}}\right] for \, \mathrm{Re} > 4000 \text{----}(4.7)$$

The Colebrook equation, represented by Eq-(4.7), is transcendental or implicit with respect to f. For this reason, it is usually solved either by trial-and-error or using an iterative solution scheme. Alternative solutions of the Colebrook equation are also available after employing some assumptions, which often reduce Eq. (4.7) to an explicit one. Among the many approximations implemented are those for a wholly rough pipe when the flow becomes independent of Re as the latter becomes very large, thereby neglecting the second term in Eq. (4.7). Other equations, such as that of Blasius Streeter and Wylie, [37], have been suggested for obtaining an explicit solution for f. The Blasius equation, however, is only applicable to smooth pipes and Re in the range of 3000-100000 Giles et al., [33]. More equations for estimating f were developed to simplify the iterative process. Murdock [38] and Miller [39] each presented an equation for the friction factor in an explicit form. However, their equations were meant to produce good initial estimates for the iterative solution scheme of the Colebrook equation, thereby dramatically reducing the number of iterations required to achieve accurate solutions. Solving their proposed equations Independently involves some error that may become substantial when the parameter e/D and Re are outside the established ranges Streeter and Wylie, [37].

The hydraulic analysis of branched and looped pipe networks often involves the implementation of a tedious and time-consuming iterative procedure that requires extensive use of computers Mohtar et al., [40] and Gerrish et al., [41]. The same is applicable for the design and hydraulic analysis of large-scale microirrigation and sprinkler irrigation systems where the accuracy of the numerical analysis is crucial. The hydraulic analysis of sub-main units using the finite element formulation, for example, results in excessively large systems of algebraic equations to be solved iteratively Bralts et al., [42]; Gerrish et al., [41] and Gerrish et al., [43]. The number of equations varies with the number of pipe elements and flow outlets within the system and may reach the order of thousands. While the Darcy-Weisbach equation is the most fundamentally sound and accurate equation for computing frictional head losses in closed conduit flow, several other empirical equations, such as the one by Hazen-Williams Brater et al., [44], are often utilised for this purpose and it is described in references Bralts and Segerlind, [45]; Haghighi et al., [46] and Mohtar et al., [40]. Otherwise, approximate explicit solutions of the Colebrook equation are carried out to eliminate the need for calculating the friction factor, f, iteratively Kang and Nishiyama, [47]. The approximate direct solution of the Colebrook equation is implemented since the sub-main unit is discretized into thousands of pipe elements resulting in large systems of non-linear algebraic equations to be solved iteratively, making it impractical to solve for f iteratively within each pipe section and during any given iteration.

Users have the option to employ the Darcy-Weisbach Equation for computing the head loss. This expression can be applied to systems transporting water and also liquid other than water. If this option is employed, the roughness for each pipe section corresponding to the Darcy-Weisbach expression must be input as data as well as the kinematics viscosity of the liquid for that system.

Table 4.2 gives the Darcy-Weisbach Equation along with the explicit relationship for the friction factor employed in the program and some typical value for roughness for new pipe. This parameter depends on type and condition of the pipe.

### 4.4.2.Pump

Many different types of pumps exist but centrifugal pumps are most frequently used in water distribution systems. Centrifugal pumps impart energy to the water through a rotating element called an impeller and may be classified in two types, centrifugal and exile flow, depending upon the direction the water is forced. The number and angles of the blades on the impeller and the speed of the pump motor affect the operating characteristics of centrifugal pumps. Fig 4.7 shows a pump curve for a centrifugal pump relating the flow to the amount of head added by the pump. Also shown is the efficiency curve, which defines how well the pump is transmitting to the water the energy, supplied to the pump motor. Two points are of interest on the pump curve; the shutoff head, and the normal discharge or rate capacity. The shutoff head is the head output by the pump at zero discharge while the normal discharge head is the discharge

Table 4.2 values of the Darcy-Weishbach Equation

| Material | $\varepsilon(ft)$ | $\varepsilon(m)$ |
|---|---|---|
| Riveted steel | 0.003-0.03 | 0.0009-0.009 |
| Concrete | 0.001-0.01 | 0.0003-0.003 |
| Cast iron | 0.00085 | 0.00026 |
| Galvanised iron | 0.005 | 0.00015 |
| Asphalted cast iron | 0.0004 | 0.00012 |
| Commercial steel | 0.00015 | 0.000045 |
| Wrought iron | 0.00015 | 0.000045 |
| Drawn tubing | 0.000005 | 0.0000015 |
| Plastic pipe | 0.000005 | 0.0000015 |

$$h_{LP} = \frac{fLV^2}{2gd} \text{ (Darcy Weishbach Equation)}$$

$$f = \frac{.25}{[\log(\frac{\varepsilon}{3.7d} + \frac{5.74}{R^{.9}})]^2}$$

Where: -

F= friction factor

R= Reynolds Number

$\varepsilon$ =Roughness (ft.)

Table [4.1] values of C in Hazen-Wlliams Equation head where the pump is operating at its most efficient level. Variable speed motors can drive pumps at a series of rotative speed, which would result in a set of pump curves for the single pump. Usually, however to supply a given flow and head, a set of pumps is provided to operate in series or parallel and the number of pumps working depends on the flow requirements. This makes it possible to operate the pump near their efficiency. There is a wide range of pumps and corresponding pump characteristic curve but the relationships do not cover the continuous domain. Nor is their relationship, which could approximate the pump characteristic curves with a parameter describing the pump. Thus, the selection of a specific curve for a task requires finding what is available and most closely fits the project's needs.

A pump can be included in any line of the pipe system. The characteristics of the pump can be described in two ways. The useful powers the pump put in to the system (in horsepower or kilowatts) can be specified. The useful power refers to the actual power, which is transformed into an increase in pressure head, and kinetic energy of the liquid as it passes through the pump. This method of describing a pump is particularly useful for a preliminary analysis or design when the specific operating characteristics of the pump are not known. The useful power can be easily computed from a typical head-discharge data point using the following

$$Usefu - power = \frac{(Head)(Disch\arg e.CfS)(Liquid.Density}{550} \text{---}\mathbf{(4.8)}$$

Alternately the relationship between the head added H-pump and discharge, $Q$, can be described by point of operating data, exponential curve can be fit to obtain a pump characteristic curve describing the pump operation of the form (Kevin E.Lansey and Larry W. Mays [30]. J Krope and D Goricanec[51]

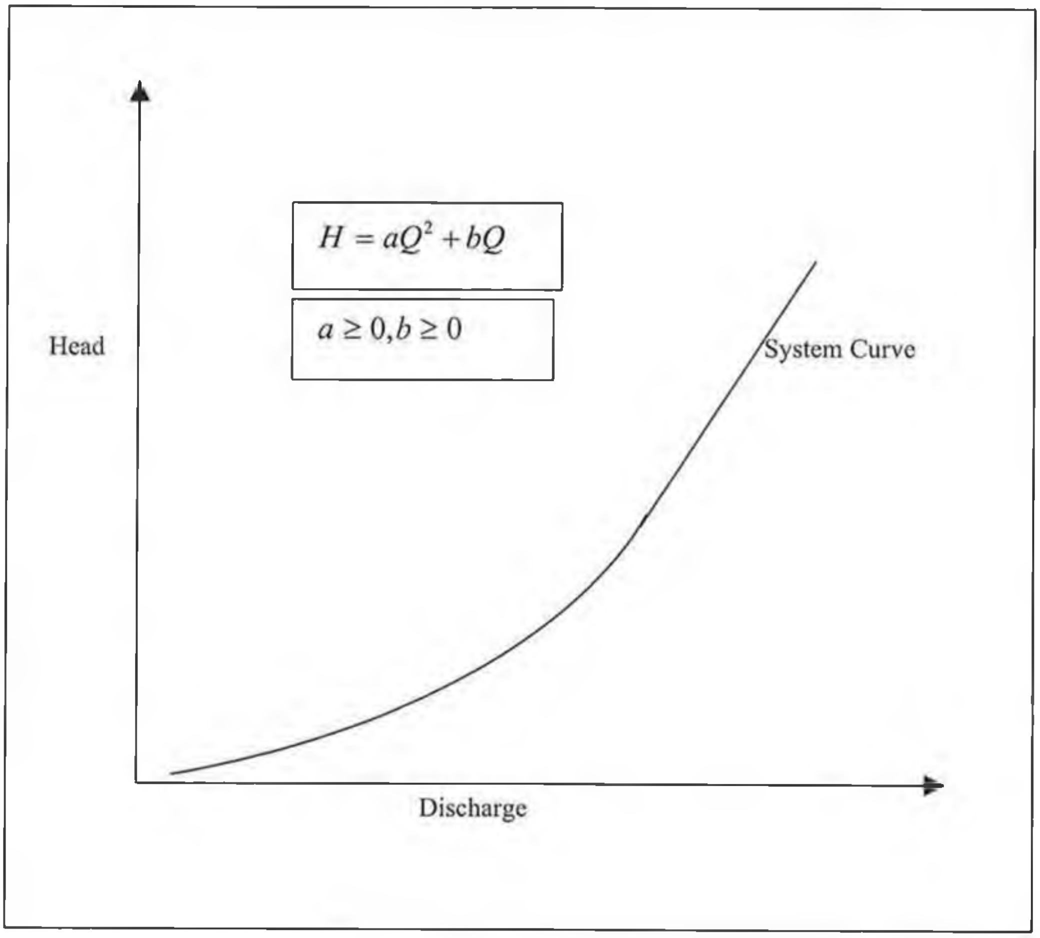


Figure 4.7 system curve

$$Hpmup = AQ^2 + BQ + Hc$$

$$Ep = H1 - CQ^n \text{ ----------(4.9)}$$

where

*H*-pump = pump pressure head.

*Hc* = cut off which represent the pressure head ast zero flow.

*C,n* = constants of the pump characteristic curve Haghighi et al.,[48].

A topically concave curve with H-pump increasing and *Q* decreasing is seen in Fig 4.7. The computer program will determine the coefficient C and exponent **n** for the curve from the pump cut-off head, *H1*, and the two additional points of the opreating data (Head-Discharge) in put for this purpose, Fig 4.8 depicts this representation. The data points are shown along with the curve of the form of Eq (4.9), which passed through these data pints. An example for this is as follows

A pump head *Ep*, flow rate (*Q*) relation is used of the form:

$$Ep = H1 - CQ^n \textbf{ -------------(4.9)}$$

where **C** and **n** are determine by passing the curve through *H2, Q2* and *H3, Q3*

These are computed as

$$n = \frac{\log(\frac{H1-H3}{H1-H2})}{\log(\frac{Q3}{Q2})} \textbf{ ------(4.10)}$$

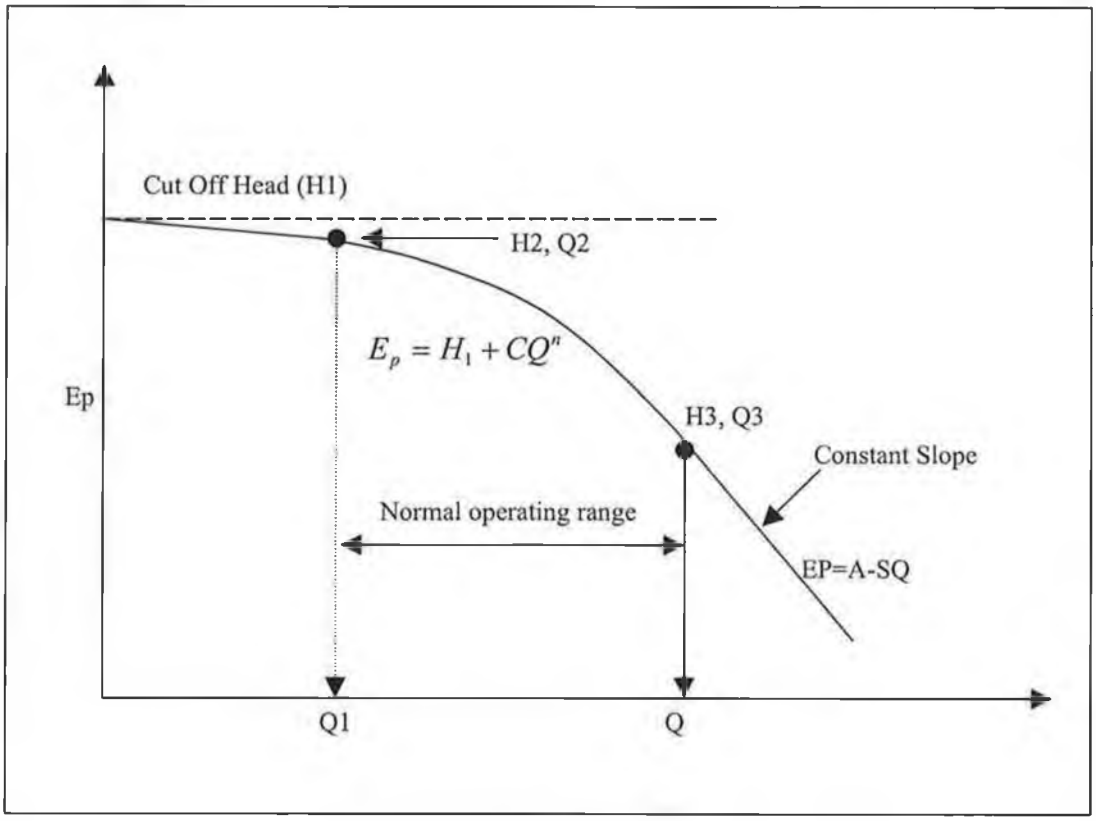


Figure (4.8) system curve

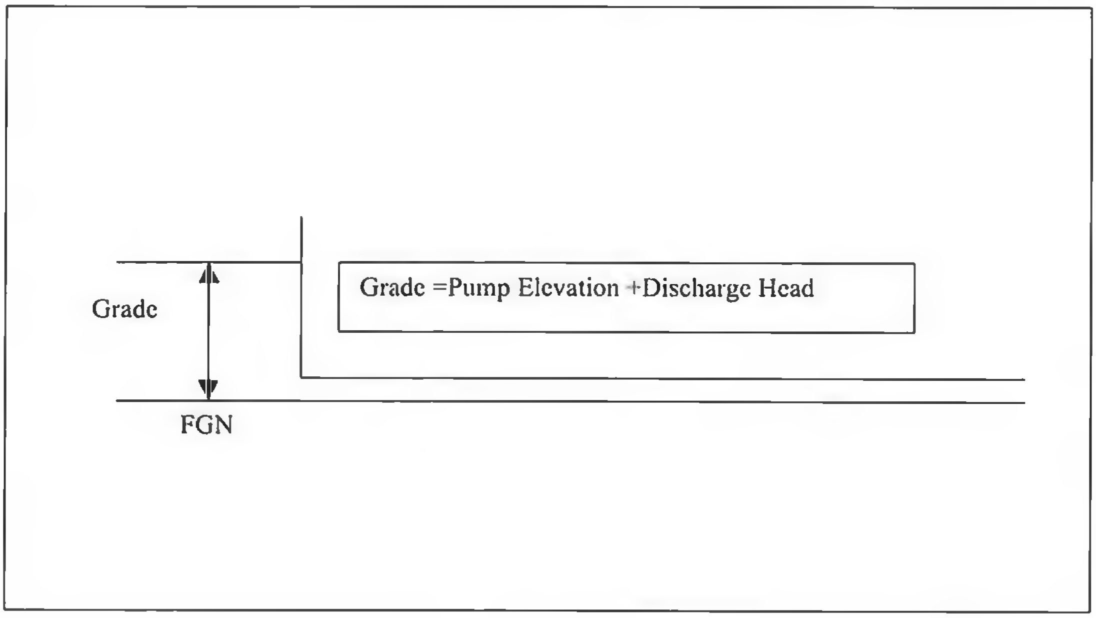


Figure 4.9

And

$$C = \frac{(H1 - H2)}{Q3^{N}} \qquad (4.11)$$

This expression holds in range (*0<=Q<=Q3*). Above *Q3* the characteristic is extended at a constant slope (as shown in figure 5) equal to the slope of Equation evaluated at *Q=Q3*.

The characteristic has the form:

$$Ep = A + SQ \text{---------}(4.12)$$

where

$$S = -nCQ3^{n-1} \text{---------------}(4.13)$$

$$A = H3 - SQ3 \text{--------------}(4.14)$$

The exponential relation given in Eq (4.9) represents the pump very well between zero flow and the third data point *H3* and *Q3* but is not suitable for negative flow and may not be suitable for flow in excess of *Q3*.

It possible, however, the solution of the hydraulic equation required *Q* pump flow rate outside the range of adequate pump representation by Eq (4.9) (the pump is not suitable for the conditions specified. A pump described by the opreating data has a check valve, which is closed if the flow reversal occurs. This indicates that the pump cut off head is not adequate to overcome the system grade caused by other factors, and the pump as described is unable to operate. A message that the check valve is closed will be generated if this situation occurs.

If the solution indicates that the pump is operating at flow rate above the third data point *Q3* then it operates on a straight line described by *EP=A-SQ* where the slop is the same as given by equation 2 at *Q=Q3*. The computer determines the values of a and s as described on the next page. This gives a characteristic, which is more realistic than Eq (4.9) gives for *Q>Q3* and will better simulate a typical pump curve in this region.

If the pump operates out of the range *Q1<Q<Q3* then the pump is considered to be operating out of it normal range and a message to this effect it generated.

Because using pump data, which is not compatible with the system requirement, may lead to poor results, it is suggested that initial analysis be carried out with constant power specified.

A third method of incorporating the effect of the pump into the system may be desirable if the analysis is to be made for a situation where the discharge pressure is to be specified or fairly closely known. For this application the pump discharge is taken as a fixed grade node with a grade designated using the specified pump discharge pressure. If the pump is on external line this fixed grade is simply a supply reservoir. If the pump is on an internal line (booster pump) then this application is identical to that for a pressure regulator with the regulated pressure being the discharge pressure of the pump.

Finally, a pump of unknown size is to be selected to input a specified amount of water into the water distribution system the pump station can be represented as a junction node. The desired inflow can be specified at this node and the analysis will compute the pump discharge, which would be required to produce the specified inflow. This application are illustrated in Fig 4.9

### 4.4.3.Minor Loss Components

A number of components in the pipe system such as valve, junction, bends, meter…etc , alter the flow pattern in the pipe creating additional turbulence which result in head loss in excess of the normal frictional losses in the pipe and this kind of losses may be substantial and should be, included in an analysis of the flow distribution of that system. The need to include such losses depends on the relative importance of theses losses to the line loses and the user must make this judgement.

Theses additional head losses are termed ***minor losses (m).*** The minor loss may vary somewhat with flow condition but it is usually sufficient to consider this to be constant for a certain component. If the pipelines are relatively long these losses are truly minor and can be neglected. In short pipe lines they may represent the major losses in the system, or if the device causes a large loss, such as a partly closed valve, its presence has dominant influence on the flow rate. In practice the engineer must use professional judgement in deciding if and how many "minor losses" should be in this analysis of fluid distribution system.

Using the concept of a minor loss coefficient M, which is a term, used to multiply the velocity head to give the concentrated head loss at the component, includes the losses. Hence, the loss is given by:

$$Hlm = \frac{MV^2}{2g} \text{--------}(4.15)$$

where

*Hlm* = fitting head loss in feet

*M* = a fitting loss due coefficient obtained experimentally Miller,[39]

*V*= the line velocity in FPS, M/S

g= gravitational acceleration $32.2\ \frac{Feet}{Sec^2}.(9.81\frac{m}{Sec^2})$.

The minor loss coefficient may vary somewhat with the flow condition but is usually sufficient to consider this to be constant for a certain component. Some values, which are normally used for common fittings, are given in Table 4.3. Numerically *M* has the same value in (E) and (SI) units.

It is often necessary to compute a value for *M* data (observed or furnished by the manufacturer) for a particular component. If the pressure drop a cross the component is known for a specific flow, the value of *M* can be easily computed. As an illustration of this, suppose the sprinklers for a watering system were known to require a pressure of 13.2 *Psig* to discharge 90 *GPM* in a 2-inch line.

The pressure drop across the sprinkler is 13.1 *PSIG* or 30.23 feet head. For 90*GPM*,the velocity in a 2'' line is 9.19 *FPS*. Hence the minor loss coefficient for the sprinkler is

$$M = \left(\frac{\frac{H_{lm}}{v^2}}{2g}\right) = 30.23/(9.19*9.19/64.4 = 23$$

Table 4.3-loss coefficient for common fitting

| Fitting | M |
|---|---|
| Globe valve, fully open | 10.0 |
| Angle valve, fully open | 5.0 |
| Swing check valve, fully open | 2.5 |
| Gate valve, fully open | 0.2 |
| Gate valve, ¾ open | 1.0 |
| Gate valve, ½ open | 5.6 |
| Gate valve, ¼ open | 24.0 |
| Short-radius elbow | 0.9 |
| Medium-radius elbow | 0.8 |
| Long-radius elbow | 0.6 |
| 45 Elbow | 0.4 |
| closed return bend | 2.2 |
| Tee, through side outlet | 1.8 |
| Tee, straight run | 0.3 |
| Coupling | 0.3 |
| 45 Wye, through side outlet | 0.8 |
| 45 Wye, straight run | 0.3 |
| Entrance | |
| Square | 0.5 |
| Bell mouth | 0.1 |
| Re-entrant | 0.9 |
| Exit | 1.0 |

### 4.4.4.Check Valve

These valves allow flow only in the specified direction. If conditions exist for flow reversal, the valves shut and the line carries no flow.

### 4.4.5.Pressure Regulator Valve (PRV)

These valves are designed to maintain a specified discharge pressure $P_R$, which is lower than the pressure upstream from the PRV. Inclusion of PRV's require only that a junction node be designated at each PRV location and the data is prepared for this representation of the pipe system. A PRV is, actually modelled in the computer simulation as shown in Fig 4.10 as two nodes. The upstream node is a junction node with a flow demand set equal to the flow through the PRV. The downstream node is a fixed grade node with a total grade equal to the elevation of the PRV plus the set pressure head $\frac{P_R}{\gamma}$. The head loss through the valve is dependent upon the downstream pressure and not on the flow in the pipe.

Two situations can occur which can keep the *PRV* from maintaining the set pressure.

Due to *PRV* bypasses in the system connections from the high to low pressure regions the *PRV* cannot control the downstream pressure which will exceed the set pressure. In this situation the flow will reverse through the PRV. A check valve may be designated in the line downstream from the PRV, which will close in this situation.

The upstream pressure drop below the set pressure. In this case the *PRV* is increasing the pressure and acting as a booster pump, which is incorrect, unless this specific application (which is discussed in a later section) is desired. The analysis should be repeated with the PRV removed if this situation occurs. If either of the above

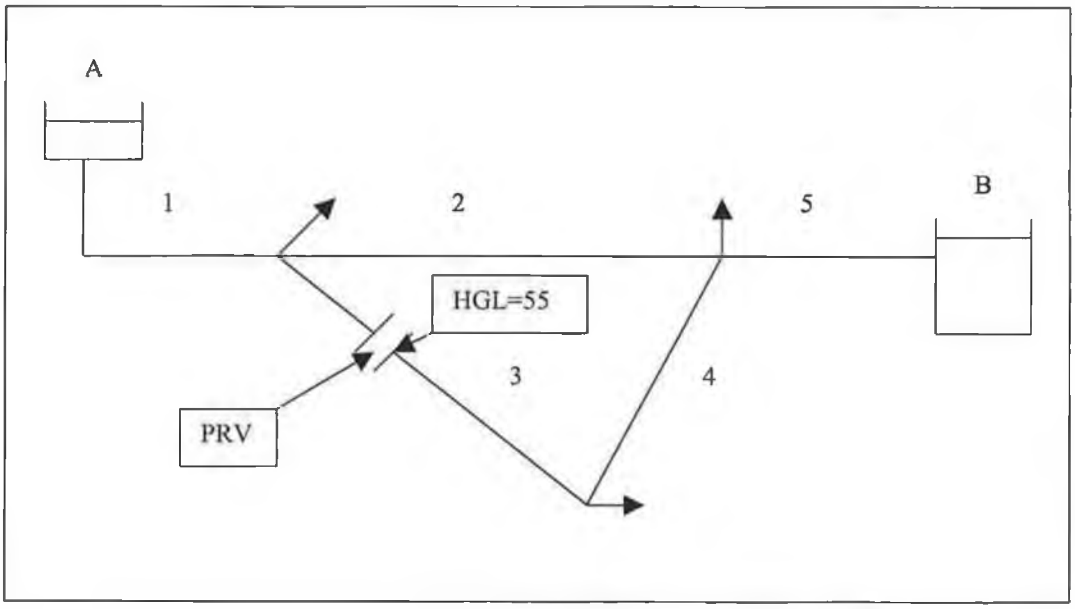


Figure 4.10 PRV in line

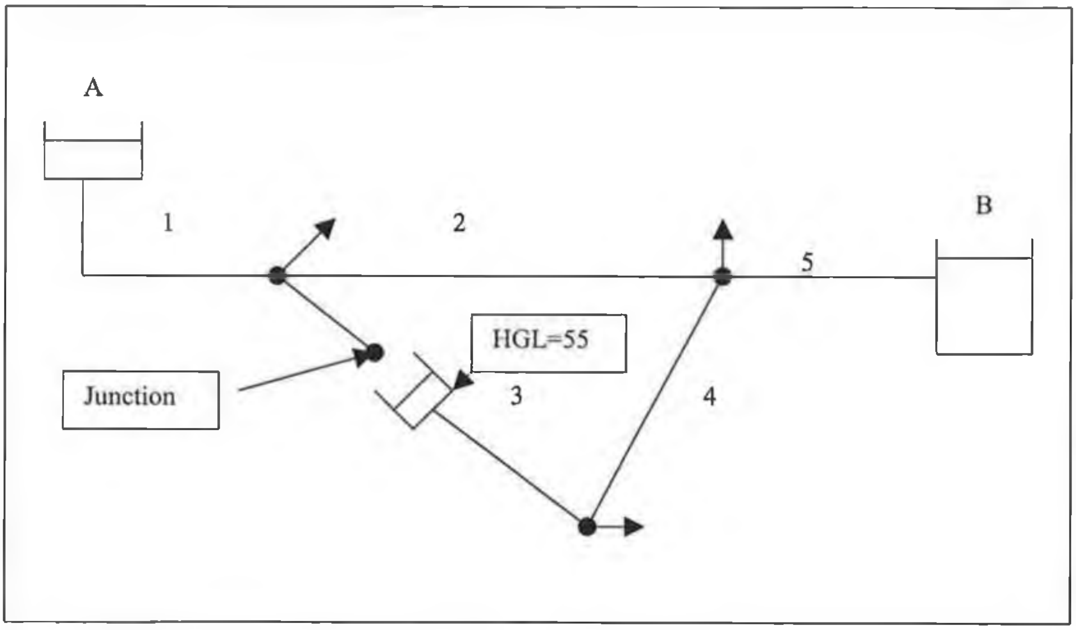


Figure 4.11 Representation as junction node and fixed grade node for solution of a network equation

situations occur the user will be alerted to this fact. The analysis of systems with PRV is discussed more fully in a later section on specific application of the program.

### 4.4.6.Storage Tanks

For regular simulation a connection to storage tank represents a fixed grade node with grade specified as the elevation of the water surface.

## 4.5.Pressure and Flow Specifications

Certain data is required to describe system pressure and flow specification. The most important of these are the flow demands at the junction node for the situation being investigated. For a system, analyses are carried out with no inflows or outflows specified but for most certain flow requirements are specified at designated junction nodes and the pressure and flow distribution is determined for this situation. At any junction node the external inflow or outflow requirement may be specified. Also the elevation of junction nodes must be specified if the pressure are to be calculated.

Values for the elevation of junction nodes are not required to compute the flow distribution and only effect the pressure calculation at the junction nodes. Thus, elevation need only be specified where values of pressure are desired.

At each fixed grade node the total grade (pressure head plus elevation) is known and must be specified. These values are shown on the schematics. This means that the elevation of surface levels reservoir and storage tanks must be specified for regular simulation.

If there are pressure requirement at fixed grade nodes these are incorporated into the grade and must be specified. If there are pressure regulating valves or pressure sustaining valves in the system the regulated pressure must be specified.

Flow direction for lines with pumps, check valves and pressure regulating valves must be specified in the data input and this is done by the order the connection nodes for the pipe section are input. Flow direction is assumed to be from the first to the second node n the order the data is input.

## 4.6.Formulation and Solution of System Equation

### 4.6.1.Basic Equation

Pipe network equations for steady state analysis have been commonly expressed in two ways. Equations, which express mass continuity and energy conservation in terms of the discharge in each pipe section, have been referred to as loop equation and this terminology will be followed here. Second formulations, which express mass continuity in term of grades at junction produces a set of equations, referred to as node equation. It has been shown that the loop equations have superior convergence characteristics and these relationship are utilised in this program:

### 4.6.2.Loop Equation

Equation (4.1) which define the relationship between the number of pipes, primary loops, junction nodes and fixed grade nodes offers a basis for formulating a set of hydraulic equations to describe a pipes system. In terms of the unknown discharge in each pipe, a number of mass continuity and energy conservation equations can be written equalling the number of pipe in the system. For each junction node a continuity relationship equating the flow into the junction (*Qin*) to the flow (*Qout*) is written as

$$\sum Qin - \sum Qout = Qe \qquad \text{J-Equation (4.16)}$$

$$\sum_{i=1}^{NP} \lambda_{j,i} Q_i - q_j = 0 \qquad \text{j=1,2,........NJ}$$

Here *Qe* represents the external inflow or demand at the junction node. For each primary loop the energy conservation equation can be written for pipe section in the loop as follows;

$$\sum Hl = \sum Eb \text{---------------} \textit{L-Equation} \text{ (4.17)}$$

where

*Hl.* energy loss in each pipe (including minor loss)

*Ep.* energy put in to the liquid by as pump

If there are no pumps in the loop then the energy equation states that the sum of the energy loss around the loop equals to zero.

If there are fixed grade node, *f–1* independent energy conservation equations can be written for a path of pipe section between any two fixed grade nodes as follows:

$$\Delta E = \sum Hl - \sum Eb \qquad F-1 \text{ equation (4.18)}$$

where E is the difference in total grade between the two-fixed grade nodes. Any connected path of pipes within the pipe system can be chosen between thesis nodes. When identifying these (*f-1*) energy equations cares must be taken to avoid redundancy. The best method to avoid this difficulty is to either choose all parallel path starting at a common node like A-B, A-C, A-D...etc or to use a series arrangement where the previous ending node for a path is the stating node for the next path like A-B, B-C, C-D.... etc. Either of these methods will result in (f-1) equations with no redundancy

As an additional generalisation Eq(4.17) can be considered to be a special case of Eq(4.18),where the difference in total gradeE is zero for a path which forms a close loop. Thus, the energy conservation relationship for a pipe network is expressed by L + f-1 path equation of the form given by Eq (4.18).

The junction and path equations constitute a set of P simulation non-linear algebraic equations referred as loop equations which describe steady state flow analysis based on the loop equation and requires the solution of this set of equation for the flow rate in each line. To do this the path equations must be expressed in terms of the flow rate, which is done as follows.

Note that the energy loss in a pipe is *Hl*, the sum of the line loss is *Hlp* and the minor loss is *Hlm*. The line loss *Hlp* expressed in terms of the flow rate is given by:

$$Hlp = KpQ^{n} \text{----------(4.19)}$$

where Kp is a pipeline constant which is a function of line length L, diameter d, and roughness C, or friction factor F, and n are an exponent. The value of Kp and n depends on the energy loss expression used for the analysis.

For the Hazen-Williames Equation

$$K_p = \frac{X * L}{C^{1.852} d^{4.87}} \text{------(4.20)}$$

and the exponent n=1.852 .The constant X depends on the unit used and has a value X is equal to 4.73 for English units and X= 10.69 for SI units.

For the Darcy-Weisbach Equation

$$Kp = f\frac{8*L}{gd^5\tau^2}$$

and the exponent n= 2

The minor loss in a pipe section (Hlm) is given as

$$Hlm = K_m Q^2 \text{ ----------(4.21)}$$

where Km is a function of the sum of the minor loss coefficients for the fitting in the pipe section $\sum M$ and the pipe diameter is given by

$$Km = 0.2517\Sigma\frac{m}{d^4} \qquad (4.22)$$

For a pump described by the useful power input the pump energy term *Ep* is given by

$$Ep = \frac{Z}{Q} \qquad (4.23)$$

where Z is a function of the useful power of the pump (*Pu*) and the specific gravity of the liquid (S) and is:

$$Z = \frac{8.814Pu}{S} \qquad \text{English unit (4.24)}$$

$$Z = \frac{0.10197Pu}{S} \qquad \text{(SI units)}$$

If the pump is described by the operating data, The in range operation is described by

$$Ep = A + BQ + CQ^2 \text{ ------(4.25)}$$

Utilising equations (4.19-4.25), the path (energy) equation expressed in term of the flow rate is:

$$\Delta E = \Sigma(KpQ^n + KmQ^2) - \Sigma\frac{Z}{Q} \text{------------(4.26)}$$

Or

$$\Delta E = \Sigma(KpQ^n + KmQ^2) - \Sigma(A + BQ + CQ^2)$$

The continuity equation (4.16) and the energy equation (4.26) form the P simultaneous equations in terms of unknown flow rates, which are labelled the loop equations. Since these are non-linear algebraic relationships no direct solution is possible. Several algorithms for solving the loop equations have been developed and the linear method is the most reliable method.

## 4.7.Solution of loop Equation

This algorithm makes use of gradient method to handle the non-linear flow rate (*Q*) term in the energy equation (4.26). The right side of the equation (4.26) represents the grade difference across a pipe section carrying a flow rate (*Q*) so that,

$$f(Q) = \Sigma(KpQ^n + KmQ^2) - \Sigma\frac{Z}{Q} \text{-----------}(4.27)$$

If the pump is described by points of operating data the energy equation can be written as

$$f(Q) = \Sigma(KpQ^n + KmQ^2) - \Sigma(A + BQ + CQ^n)$$

The function and its gradient is evaluated at an approximate value of the flow rate *Qi*, are used in the algorithms presented for solving the loop equation. The grade difference in a pipe section based on *Q*=*Qi* is:

$$f(Qi) = Hi = \Sigma(KpQi^2 + KmQi^2) - \Sigma\frac{Z}{Qi} \text{----------}(4.28)$$

$$Hi = \Sigma(KpQi^n + KmQi^2) - \Sigma(A + BQi + CQi^n)$$

and the gradient evaluated at *Q*=*Qi* is:

$$f'(Qi) = Gi = \frac{\delta f}{\delta Q}|Q = Qi = nKpQi^{n-1} + 2KmQi + \frac{Z}{Qi^2}$$

Or

$$Gi = nKpQi^{n-1} + 2KmQi - (B + 2CQi)\text{-----------}(4.29)$$

The term Hi and *Gi* as defined above are used in the following discussion of the linear method for solving the loop equation and in the computer program developed herein. The linear method is based on a simultaneous solution of the basic hydraulic equation for the pipe system and has been reported for closed loop system Eq (4.1). Since the energy equation for the paths are non-linear these equations are first linearized in terms of an approximate flow rate *Qi* in each pipe. This is done by taking the derivative of the variable in Eq (4.26) with respect to the flow rate and evaluating them at *Q*=*Qi* using the following approximation

$$f(Q_{l+1}) = f(Q_l) + \frac{\delta f}{\delta Q}|(Q_{l+1} - Q_l) \qquad (4.30)$$

$$= f(Q_l) + G_l(Q_{l+1} - Q_l)$$

The path equation can be written as

$$\Delta E = \sum f(Q_{i+1}) = \sum f(Q_i) + \sum Gi(Q_{l+1} - Q_i)$$

where the summation refers to summing over each pipe. For loop $\Delta E$=0,so that

$$\sum G_i Q_{i+1} = \sum (G_i Q_i - f(Q_i))$$

For a path between two-fixed grade nodes the $\Delta E$ is a constant, then

$$\sum G_i Q_{i+1} = \sum (G_i Q_i - f(Q_i)) + \Delta E$$

when this relationship is applied to the energy equation Eq (4.26), the following linearized equation result:

$$\Sigma GiQ = \Sigma (GiQi - Hi) \text{---------}(4.31)$$

The sum refers to each pipe in the path. Eq (4.31) is employed to formulate (*L+F-1*) energy equations, which one combined with the junction Eq (4.16), to form a set of (*Np*) simultaneous linear equations in terms of the flow rate in each pipe.

$$q_j + \sum_{n=1}^{NP} \lambda_{j,n} Q_{n=} = 0$$

For each junction in the system ”j”, j=1,2…NJ

$$\sum G_i Q_{i+1} = \sum (G_i Q_i - f(Q_i))$$

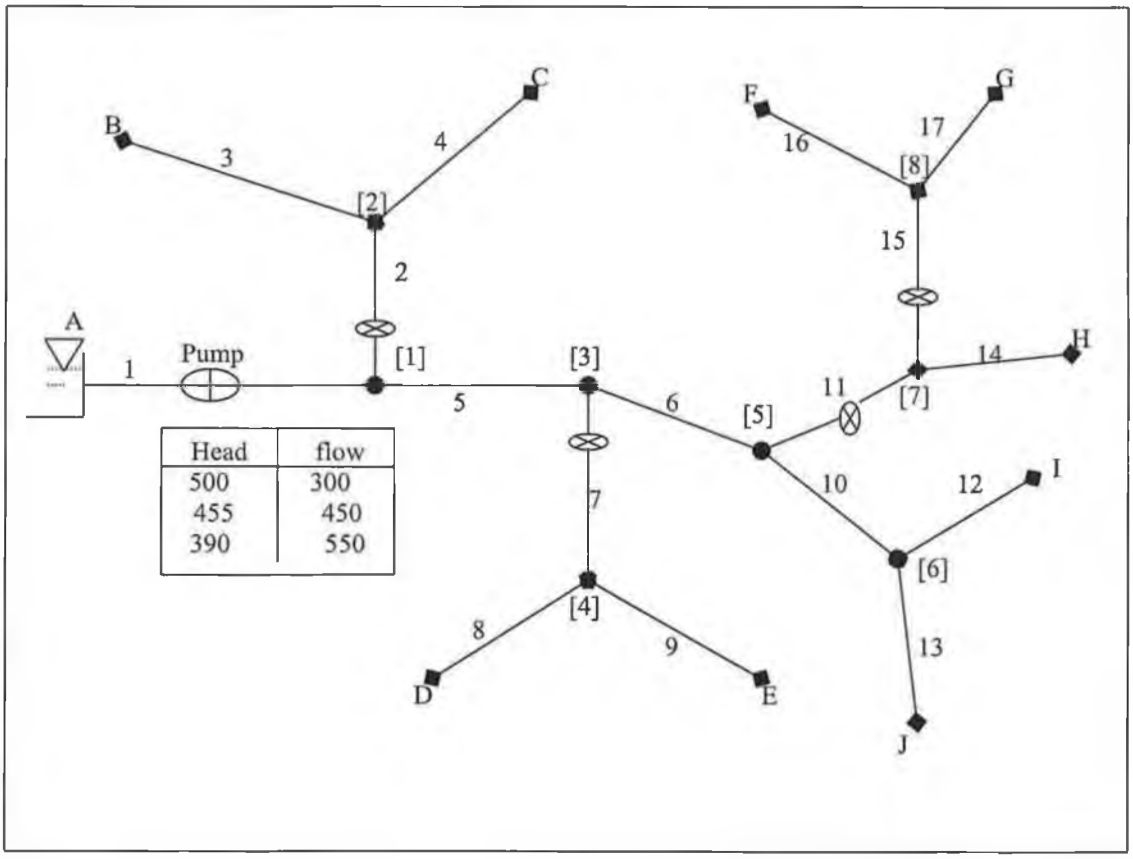
$$\begin{bmatrix} \frac{\delta F_1}{\delta Q_1} & \frac{\delta F}{\delta Q_2} & \cdot & \frac{\delta F}{\delta Q_n} \\ \frac{\delta F_2}{\delta Q_1} & \frac{\delta F_2}{\delta Q_2} & \cdot & \frac{\delta F_2}{\delta Q_n} \\ \cdot & \cdot & \cdot & \cdot \\ \frac{\delta F_n}{\delta Q_1} & \frac{\delta F_n}{\delta Q_2} & \cdot & \frac{\delta F_n}{\delta Q_n} \end{bmatrix} \begin{Bmatrix} Q_{i+1} \\ Q_{i+2} \\ \cdot \\ Q_{i+n} \end{Bmatrix} = \begin{Bmatrix} F_1 \\ F_2 \\ \cdot \\ F_n \end{Bmatrix}$$

Using a set of initial flow rate Qi in each pipe, the system of linear equations is solved for Qi+1 using matrix procedure. This new flow rate Qi+1 is used as the known values to obtain a second solution of the linear equations. This procedure is repeated until the change in flow rates (Qi+1-Qi) in insignificant.

The technique used to solve the system of equations is as follows. Based on an arbitrary initial value for the flow rate in each pipe in the system the linearized equations are solved using matrix for solving the linear equations. This set of the flow rate is used to linearize the equations and the second solution is obtained .The procedure is repeated till the change in the flow rate obtained in successive trials is insignificant. Because all flows are computed simultaneously convergence is expected and occurs very fast compared to other procedures. Usually few trials are required for a high degree of accuracy even for large system.

Now some details of how the method works in practice is shown. The network in figure 4.12 is considered and the data in the figure and the table 4.4-4.6 are used to form and solve the system of equations.

These seventeen-pipe tree type systems shown represent a green watering system for a nine-hole golf course. The system discharge water through sprinklers at known elevation into the atmosphere. Therefore, each discharge point is a fixed grade node with the grade equal to the sprinkler elevation (discharge pressure =0). The analysis will compute how much flow can be delivered at each sprinkler. The data for the system are shown and three operating data points describe the pump



| Head | flow |
|---|---|
| 500 | 300 |
| 455 | 450 |
| 390 | 550 |

*Figure 4.12 17 pipe tree type watering system p=17,l=0,j=8,f=10*

The corresponding 8 junction continuity equations are:

$$F_1 = Q_1 - Q_2 - Q_5 = 0$$

$$F_2 = Q_2 - Q_3 - Q_4 = 0$$

$$F_3 = Q_5 - Q_6 - Q_7 = 0$$

$$F_4 = Q_5 - Q_6 - Q_7 = 0$$

$$F_5 = Q_6 - Q_{10} - Q_{11} = 0$$

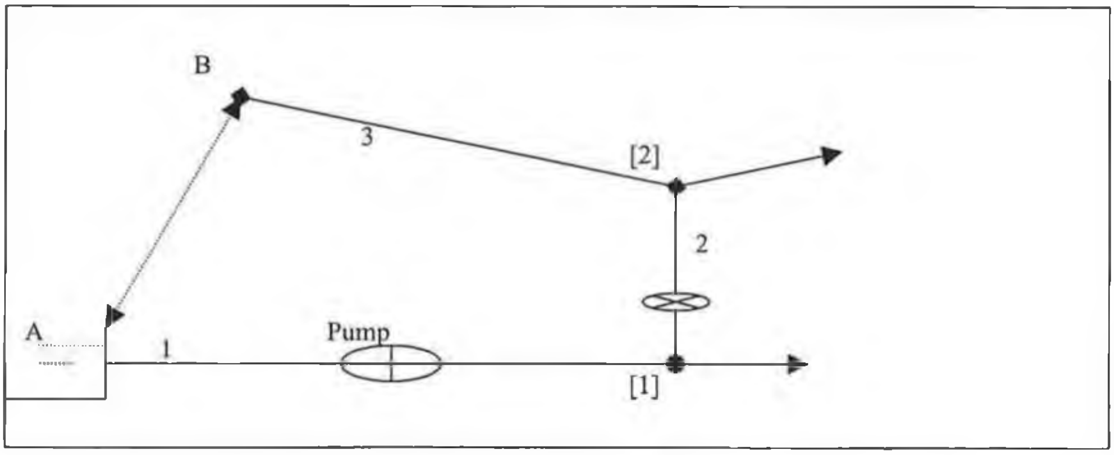
$$F_6 = Q_{10} - Q_{12} - Q_{13} = 0$$

$$F_7 = Q_{11} - Q_{14} - Q_{15} = 0$$

$$F_8 = Q_{15} - Q_{16} - Q_{17} = 0$$

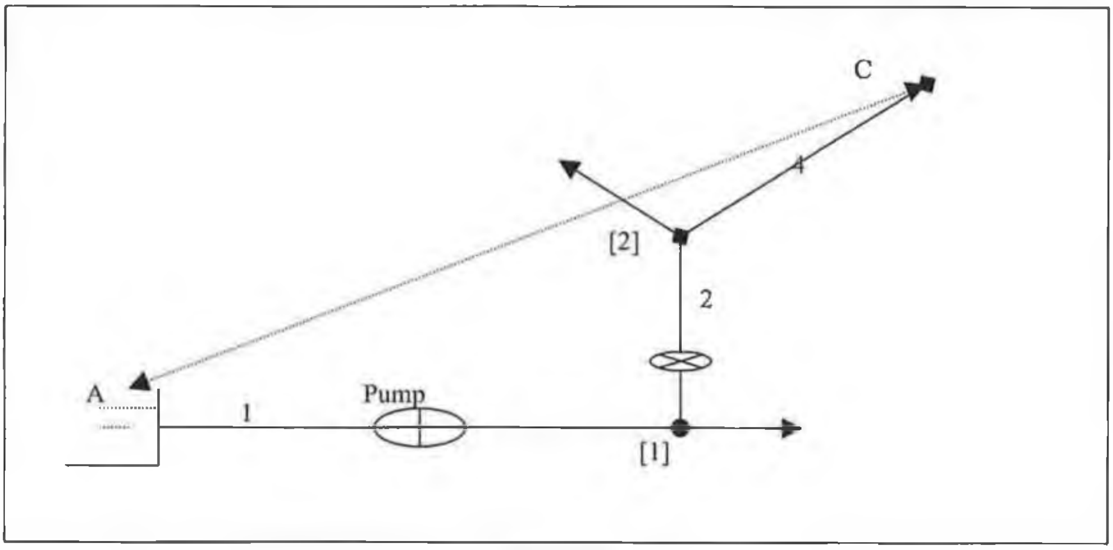
For the linear method an initial flow rate is needed but are not required to satisfy the continuity equations Wood [25]. The initial flow rate for the linear method is based on a constant flow velocity in an arbitrary direction.

The energy equations around each loop or path in the systems are written as follow.



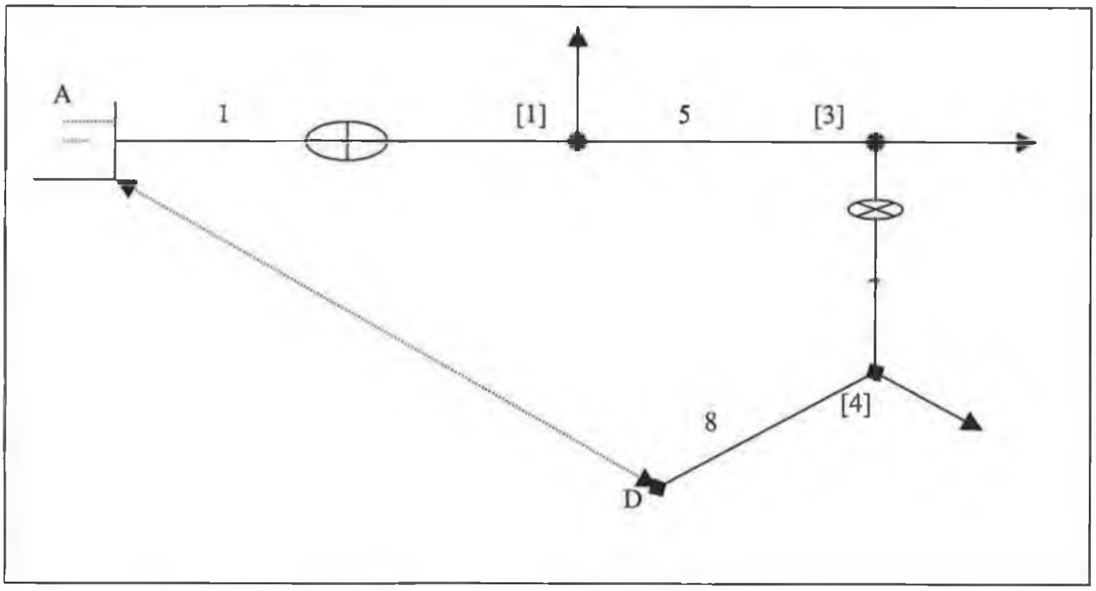
$$f_1 = K_{p_1} Q_1^n + K_{m_1} Q_1^2 - head + K_{p_2} Q_2^n + K_{m_2} Q_2^2 + K_{p_3} Q_3^n + K_{m_3} Q_3^2 =$$

*Figure 4.13 energy equation between (A-B) pipe [1-2]*



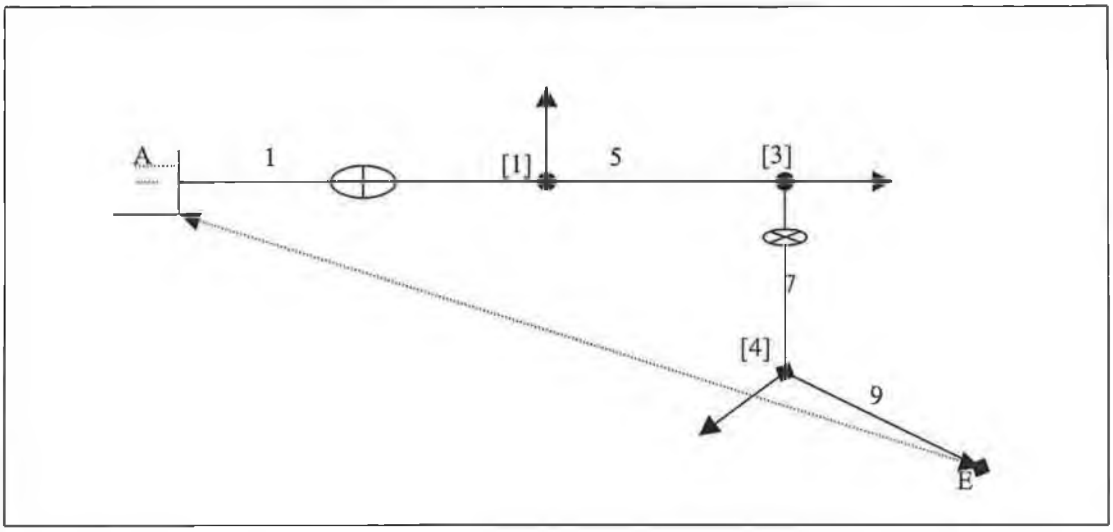
$$f_2 = K_{p_1}Q_1^n + K_{m_1}Q_1^2 - head + K_{p_2}Q_2^n + K_{m_2}Q_2^2 + K_{p_4}Q_4^n + K_{m_4}Q_4^2 =$$

*Figure 4.14 energy equation between (A-C) pipe [1-2-4*



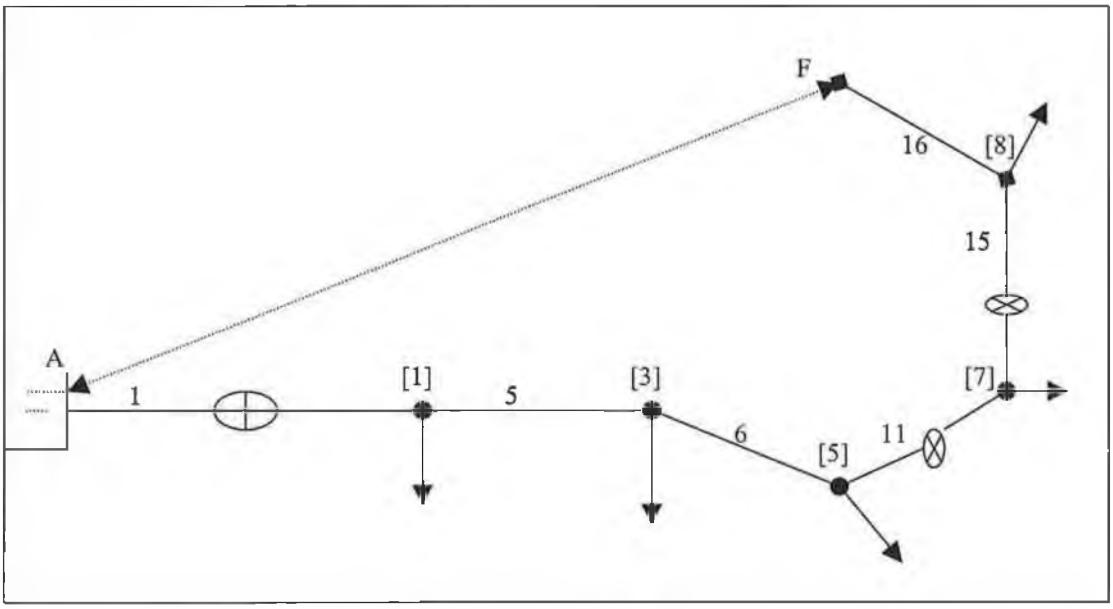
$$f_3 = K_{p_1}Q_1^n + K_{m_1}Q_1^2 - head + K_{p_5}Q_5^n + K_{m_5}Q_5^2 + K_{p_7}Q_7^n + K_{m7}Q_7^2 + K_{p_8}Q_3^n + K_{m_8}Q_8^2 =$$

*Figure 4.15 Energy equation between (A-D) pipe [1-5-7-8]*



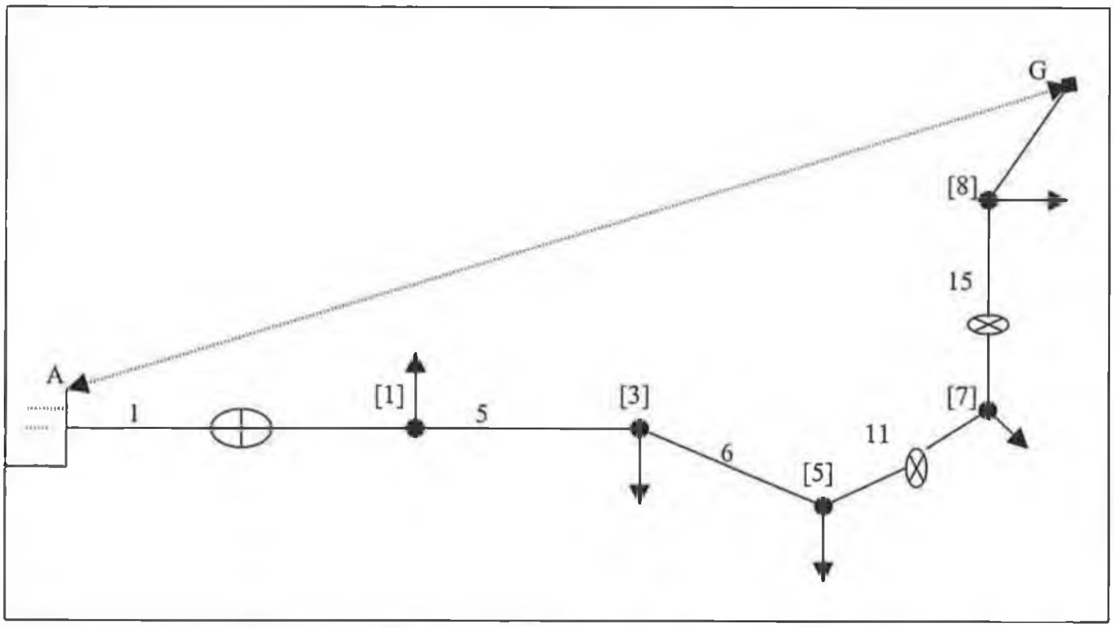
$$f_4 = K_{p_1}Q_1^n + K_{m_1}Q_1^2 - head + K_{p_5}Q_5^n + K_{m_5}Q_5^2 + K_{p_7}Q_7^n + K_{m_7}Q_7^2 + K_{p_9}Q_9^n + K_{m_9}Q_9^2 =$$

*Figure 4.16 energy equation between (A-E) pipe [1-5-7-9]*



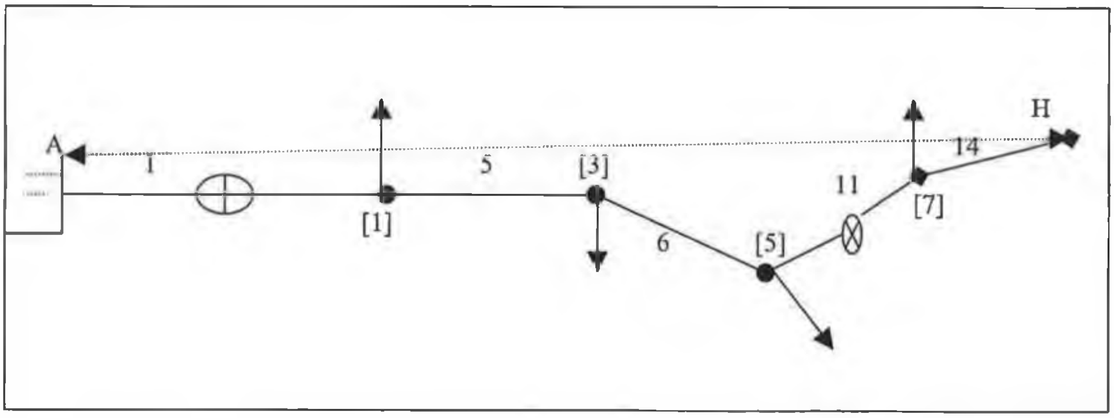
$$f_5 = K_{p_1}Q_1^n + K_{m_1}Q_1^2 - head + K_{p_5}Q_5^n + K_{m_5}Q_5^2 + K_{p_6}Q_6^n + K_{m_6}Q_6^2 + K_{p_{11}}Q_{11}^n + K_{m_{11}}Q_{11}^2 +$$
$$K_{p_{15}}Q_{15}^n + K_{m_{15}}Q_{15}^2 + K_{p_{16}}Q_{16}^n + K_{m_{16}}Q_{16}^2 =$$

*Figure 4.17 energy equation between (A-F) pipe [1-5-6-11-15-16]*



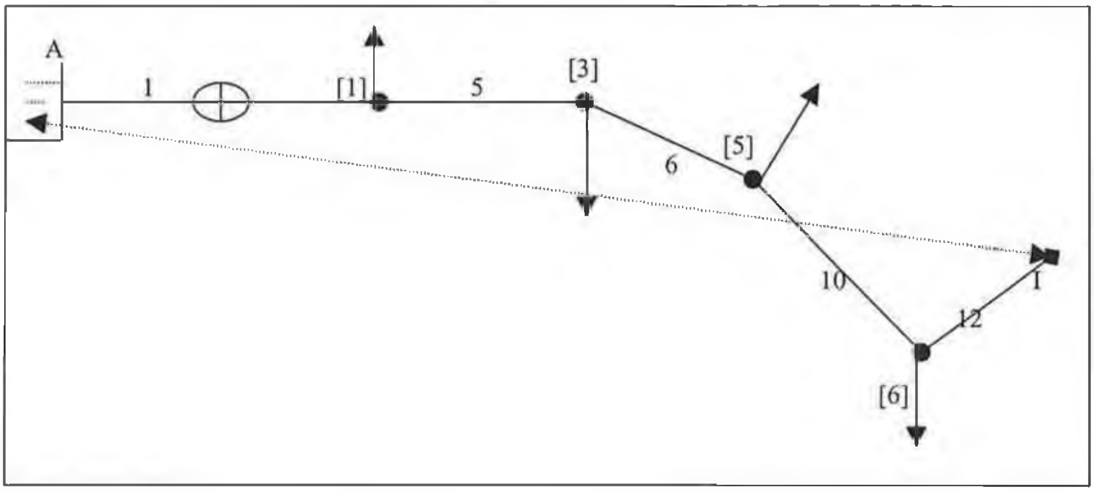
$$f_6 = K_{p_1}Q_1^n + K_{m_1}Q_1^2 - head + K_{p_5}Q_5^n + K_{m_5}Q_5^2 + K_{p_6}Q_6^n + K_{m_6}Q_6^2 + K_{p_{11}}Q_{11}^n + K_{m_{11}}Q_{11}^2 + K_{p_{15}}Q_{15}^n + K_{m_{15}}Q_{15}^2 + K_{p_{17}}Q_{17}^n + K_{m_{17}}Q_{17}^2 =$$

*Figure 4.18 energy equation between (A-G) pipe [1-5-6-11-17]*



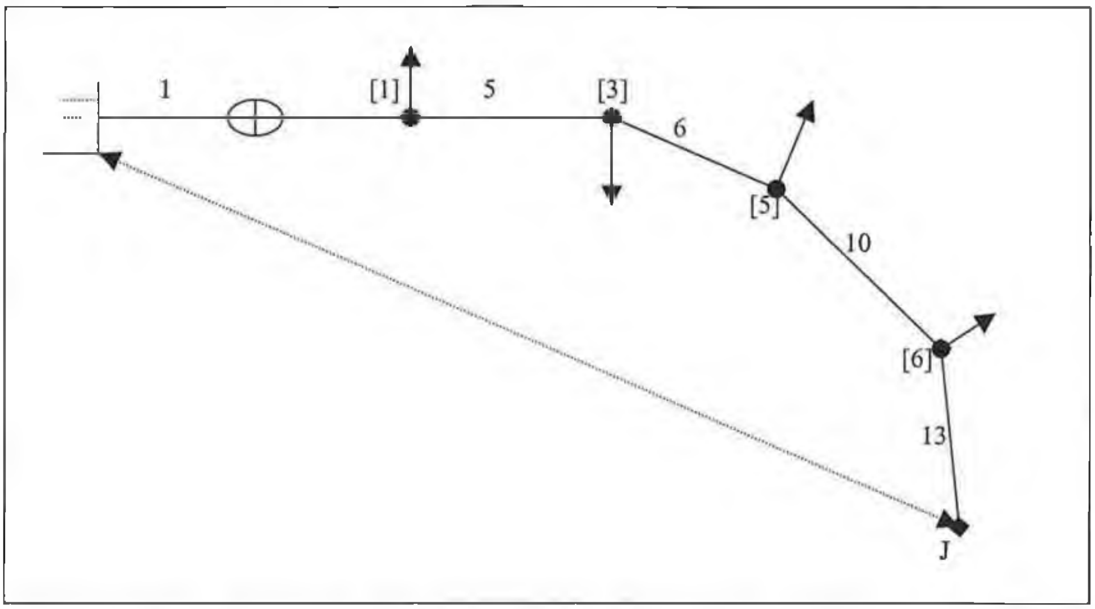
$$f_7 = K_{p_1}Q_1^n + K_{m_1}Q_1^2 - head + K_{p_5}Q_5^n + K_{m_5}Q_5^2 + K_{p_6}Q_6^n + K_{m_5}Q_6^2 + K_{p_{11}}Q_{11}^n + K_{m_{11}}Q_{11}^2 + K_{p_{14}}Q_{14}^n + K_{m_{14}}Q_{14}^2 =$$

*Figure 4.19 energy equation between (A-H) pipe [1-5-6-11-14]*



$$f_8 = K_{p_1}Q_1^n + K_{m_1}Q_1^2 - head + K_{p_5}Q_5^n + K_{m_5}Q_5^2 + K_{p_6}Q_6^n + K_{m_6}Q_6^2 + K_{p_{10}}Q_{10}^n + K_{m_{10}}Q_{10}^2 + K_{p_{12}}Q_{12}^n + K_{m_{12}}Q_{12}^2 =$$

*Figure 4.20 energy equation between (A-I) pipe [1-5-6-10-12]*



$$f_9 = K_{p_1}Q_1^n + K_{m_1}Q_1^2 - head + K_{p_5}Q_5^n + K_{m_5}Q_5^2 + K_{p_6}Q_{10}^n + K_{m_6}Q_6^2 + K_{p_{10}}Q_{10}^n + K_{m_{10}}Q_{10}^2 + K_{p_{13}}Q_{13}^n + K_{m_{13}}Q_{13}^2 =$$

*Figure 4.21 energy equation between (A-J) pipe [1-5-6-10-13]*

Since the energy for the path are non-linear these equations are fist linearized in terms of an approximate flow rate *Qoi* for each pipe. Taking the derivative of the energy equations with respect to the flow rate this can be done. Equation (4.30) is applied on the non-linear equations and the following systems of equations are been obtained.

$$\frac{\partial F_1}{\partial Q_1}Q_1 + \frac{\partial F_1}{\partial Q_2}Q_2 + \frac{\partial F_1}{\partial Q_3}Q_3 = \frac{\partial F_1}{\partial Q_1}Q_1 - F_1(Q_1) + \frac{\partial F_1}{\partial Q_2} - F_1(Q_2) + \frac{\partial F_1}{\partial Q_3} - F_1(Q_3) + \Delta E$$

$$\frac{\partial F_2}{\partial Q_1}Q_1 + \frac{\partial F_2}{\partial Q_2}Q_2 + \frac{\partial F_2}{\partial Q_4}Q_4 = \frac{\partial F_2}{\partial Q_1}Q_1 - F_2(Q_1) + \frac{\partial F_2}{\partial Q_2} - F_1(Q_2) + \frac{\partial F_2}{\partial Q_4} - F_2(Q_4) + \Delta E_2$$

$$\frac{\partial F_3}{\partial Q_1}Q_1 + \frac{\partial F_3}{\partial Q_5}Q_5 + \frac{\partial F_3}{\partial Q_7}Q_7 + \frac{\partial F_3}{\partial Q_8} = \frac{\partial F_3}{\partial Q_1}Q_1 - F_3(Q_1) + \frac{\partial F_3}{\partial Q_5} - F_3(Q_5) + \frac{\partial F_3}{\partial Q_7} - F_3(Q_7) +$$
$$\frac{\partial F_3}{\partial Q_8} - F_8(Q_8) + \Delta E_3$$

$$\frac{\partial F_4}{\partial Q_1}Q_1 + \frac{\partial F_4}{\partial Q_5}Q_5 + \frac{\partial F_4}{\partial Q_7}Q_7 + \frac{\partial F_4}{\partial Q_9}Q_9 = \frac{\partial F_4}{\partial Q_1}Q_1 - F_4(Q_1) + \frac{\partial F_4}{\partial Q_2} - F_4(Q_5) + \frac{\partial F_4}{\partial Q_7} - F_4(Q_7) +$$
$$\frac{\partial F_4}{\partial Q_9}Q_9 - F_4(Q_9) + \Delta E_4$$

$$\frac{\partial F_5}{\partial Q_1}Q_1 + \frac{\partial F_5}{\partial Q_5}Q_5 + \frac{\partial F_5}{\partial Q_6}Q_6 + \frac{\partial F_5}{\partial Q_{11}}Q_{11} + \frac{\partial F_5}{\partial Q_{15}}Q_{15} + \frac{\partial F_5}{\partial Q_{16}}Q_{16} = \frac{\partial F_5}{\partial Q_1}Q_1 - F_5(Q_1) +$$
$$\frac{\partial F_5}{\partial Q_5}Q_5 - F_5(Q_5) + \frac{\partial F_5}{\partial Q_6}Q_6 - F_5(Q_6) + \frac{\partial F_5}{\partial Q_{11}}Q_{11} - F_5(Q_{11}) + \frac{\partial F_5}{\partial Q_{15}}Q_{15} - F(Q_{15}) +$$
$$\frac{\partial F_5}{\partial Q_{16}}Q_{16} - F_5(Q_{16}) + \Delta E_5 = 0$$

$$\frac{\partial F_6}{\partial Q_1}Q_1 + \frac{\partial F_6}{\partial Q_5}Q_5 + \frac{\partial F_6}{\partial Q_6}Q_6 + \frac{\partial F_6}{\partial Q_{11}}Q_{11} + \frac{\partial F_6}{\partial Q_{15}}Q_{15} + \frac{\partial F_6}{\partial Q_{17}}Q_{17} = \frac{\partial F_6}{\partial Q_1}Q_1 - F_6(Q_1) +$$
$$\frac{\partial F_6}{\partial Q_5}Q_5 - F_6(Q_5) + \frac{\partial F_6}{\partial Q_6}Q_6 - F_6(Q_6) + \frac{\partial F_6}{\partial Q_{11}}Q_{11} - F_6(Q_{11}) + \frac{\partial F_6}{\partial Q_{15}}Q_{15} - F_6(Q_{15}) +$$
$$\frac{\partial F_6}{\partial Q_{17}}Q_{17} - F_{17}(Q_{17}) + \Delta E_6 = 0$$

$$\frac{\partial F_7}{\partial Q_1}Q_1 + \frac{\partial F_7}{\partial Q_5}Q_5 + \frac{\partial F_7}{\partial Q_6}Q_6 + \frac{\partial F_7}{\partial Q_{11}}Q_{11} + \frac{\partial F_7}{\partial Q_{14}}Q_{14} = \frac{\partial F_7}{\partial Q_1}Q_1 - F_7(Q_1) + \frac{\partial F_7}{\partial Q_5} - F_7(Q_5) +$$
$$\frac{\partial F_7}{\partial Q_6} - F_7(Q_6) + \frac{\partial F_7}{\partial Q_{11}} - F_7(Q_{11}) + \frac{\partial F_7}{\partial Q_{14}} - F_7(Q_{14}) + \Delta E_7 = 0$$

$$\frac{\partial F_8}{\partial Q_1}Q_1+\frac{\partial F_8}{\partial Q_5}Q_5+\frac{\partial F_8}{\partial Q_6}Q_6+\frac{\partial F_8}{\partial Q_{10}}Q_{10}+\frac{\partial F_8}{\partial Q_{12}}Q_{12}=\frac{\partial F_8}{\partial Q_1}Q_1-F_8(Q_8)+\frac{\partial F_8}{\partial Q_5}Q_5-F_8(Q_5)+$$
$$\frac{\partial F_8}{\partial Q_6}-F_8(Q_6)+\frac{\partial F_8}{\partial Q_{10}}-F_8(Q_{10})+\frac{\partial F_8}{\partial Q_{12}}-F_8(Q_{12})+\Delta E_8$$
$$\frac{\partial F_9}{\partial Q_1}Q_1+\frac{\partial F_9}{\partial Q_5}Q_5+\frac{\partial F_9}{\partial Q_6}Q_6+\frac{\partial F_9}{\partial Q_{10}}Q_{10}+\frac{\partial F_9}{\partial Q_{13}}Q_{13}=\frac{\partial F_9}{\partial Q_1}Q_1-F_9(Q_1)+\frac{\partial F_9}{\partial Q_5}Q_5-F_9(Q_5)+$$
$$\frac{\partial F_9}{\partial Q_6}Q_6-F_9(Q_6)+\frac{\partial F_9}{\partial Q_{10}}Q_{10}-F_9(Q_{10})+\frac{\partial F_9}{\partial Q_{12}}Q_{13}-F_9(Q_{13})+\Delta E_9$$

Now the solution of the equations is started by using the equation [*D*]{*Z*}={*F*}, {*Qm*}={*Qm+1*}{*Z*}. In this case the system of equations are solved repeatedly for updated vector {z} until it is close to be sufficiently small. *D* is the Jacobian matrix which is contain all the information about the system.

$$[D]=\begin{bmatrix}
-1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 1 \\
\frac{\partial F_9}{\partial Q_1} & \frac{\partial F_9}{\partial Q_2} & \frac{\partial F_9}{\partial Q_3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{\partial F_{10}}{\partial Q_1} & \frac{\partial F_{10}}{\partial Q_2} & 0 & \frac{\partial F_{10}}{\partial Q_4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{\partial F_{11}}{\partial Q_1} & 0 & 0 & 0 & \frac{\partial F_{11}}{\partial Q_5} & 0 & \frac{\partial F_{11}}{\partial Q_7} & \frac{\partial F_{11}}{\partial Q_8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{\partial F_{12}}{\partial Q_1} & 0 & 0 & 0 & \frac{\partial F_{12}}{\partial Q_5} & 0 & \frac{\partial F_{12}}{\partial Q_7} & 0 & \frac{\partial F_{12}}{\partial Q_9} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{\partial F_{13}}{\partial Q_1} & 0 & 0 & 0 & \frac{\partial F_{13}}{\partial Q_5} & \frac{\partial F_{13}}{\partial Q_6} & 0 & 0 & 0 & 0 & \frac{\partial F_{13}}{\partial Q_{10}} & 0 & 0 & 0 & \frac{\partial F_{13}}{\partial Q_{15}} & \frac{\partial F_{13}}{\partial Q_{16}} & 0 \\
\frac{\partial F_{14}}{\partial Q_1} & 0 & 0 & 0 & \frac{\partial F_{14}}{\partial Q} & \frac{\partial F_{14}}{\partial Q} & 0 & 0 & 0 & 0 & \frac{\partial F_{14}}{\partial Q} & 0 & 0 & 0 & \frac{\partial F_{14}}{\partial Q} & 0 & \frac{\partial F_{14}}{\partial Q_{17}} \\
\frac{\partial F_{15}}{\partial Q_1} & 0 & 0 & 0 & \frac{\partial F_{15}}{\partial Q_5} & \frac{\partial F_{15}}{\partial Q_6} & 0 & 0 & 0 & 0 & \frac{\partial F_{15}}{\partial Q_{11}} & 0 & 0 & \frac{\partial F_{15}}{\partial Q_{14}} & 0 & 0 & 0 \\
\frac{\partial F_{16}}{\partial Q_1} & 0 & 0 & 0 & \frac{\partial F_{16}}{\partial Q_5} & \frac{\partial F_{16}}{\partial Q_6} & 0 & 0 & 0 & \frac{\partial F_{16}}{\partial Q_{10}} & 0 & \frac{\partial F_{16}}{\partial Q_{12}} & 0 & 0 & 0 & 0 & 0 \\
\frac{\partial F_{17}}{\partial Q_1} & 0 & 0 & 0 & \frac{\partial F_{17}}{\partial Q_5} & \frac{\partial F_{17}}{\partial Q_6} & 0 & 0 & 0 & \frac{\partial F_{17}}{\partial Q_{10}} & 0 & 0 & \frac{\partial F_{17}}{\partial Q_{13}} & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$\{F\} = \left\{\begin{matrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \\ F_8 \\ F_9 \\ F_{10} \\ F_{11} \\ F_{12} \\ F_{13} \\ F_{14} \\ F_{15} \\ F_{16} \\ F_{17} \end{matrix}\right\} \qquad \text{and} \qquad \{Z\} = \left\{\begin{matrix} Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \\ Z_7 \\ Z_8 \\ Z_9 \\ Z_{10} \\ Z_{11} \\ Z_{12} \\ Z_{13} \\ Z_{14} \\ Z_{15} \\ Z_{16} \\ Z_{17} \end{matrix}\right\}$$

where is:

$$F_9 = \frac{\partial F_9}{\partial Q_1} Q_1 - F_9(Q_1) + \frac{\partial F_9}{\partial Q_2} Q_2 - F_9(Q_2) + \frac{\partial F_9}{\partial Q_3} Q_3 - F_9(Q_3) + \Delta E_1$$

$$F_{10} = \frac{\partial F_{10}}{\partial Q_1} Q_1 - F_{10}(Q_1) + \frac{\partial F_{10}}{\partial Q_2} Q_2 - F_{10}(Q_2) + \frac{\partial F_{10}}{\partial Q_4} Q_4 - F_{10}(Q_4) + \Delta E_2$$

$$F_{11} = \frac{\partial F_{11}}{\partial Q_1} Q_1 - F_{11}(Q_1) + \frac{\partial F_{11}}{\partial Q_5} Q_5 - F_{11}(Q_5) + \frac{\partial F_{11}}{\partial Q_7} Q_7 - F_{11}(Q_7) + \frac{\partial F_{11}}{\partial Q_8} Q_8 - F_{11}(Q_8) + \Delta E_3$$

$$F_{12} = \frac{\partial F_{12}}{\partial Q_1} Q_1 - F_{12}(Q_1) + \frac{\partial F_{12}}{\partial Q_5} Q_5 - F_{12}(Q_5) + \frac{\partial F_{12}}{\partial Q_7} Q_7 - F_{12}(Q_7) + \frac{\partial F_{12}}{\partial Q_9} Q_9 - F_{12}(Q_9) + \Delta E_4$$

$$F_{13} = \frac{\partial F_{13}}{\partial Q_1} Q_1 - F_{13}(Q_1) + \frac{\partial F_{13}}{\partial Q_5} Q_5 - F_{13}(Q_5) + \frac{\partial F_{13}}{\partial Q_6} Q_6 - F_{13}(Q_6) + \frac{\partial F_{13}}{\partial Q_{11}} Q_{11} - F_{13}(Q_{11}) + \\ \frac{\partial F_{13}}{\partial Q_{15}} Q_{15} - F_{13}(Q_{15}) + \frac{\partial F_{13}}{\partial Q_{17}} Q_{17} - F_{13}(Q_{17}) + \Delta E_5$$

$$F_{14} = \frac{\partial F_{14}}{\partial Q_1} Q_1 - F_{14}(Q_1) + \frac{\partial F_{14}}{\partial Q_5} Q_5 - F_{14}(Q_5) + \frac{\partial F_{14}}{\partial Q_6} Q_6 - F_{14}(Q_6) + \frac{\partial F_{14}}{\partial Q_{11}} Q_{11} - F_{14}(Q_{11}) + \\ \frac{\partial F_{14}}{\partial Q_{15}} Q_{15} - F_{14}(Q_{15}) + \frac{\partial F_{14}}{\partial Q_{17}} Q_{17} - F_{14}(Q_{17}) + \Delta E_6$$

$$F_{15} = \frac{\partial F_{15}}{\partial Q_1} Q_1 - F_{15}(Q_1) + \frac{\partial F_{15}}{\partial Q_5} Q_5 - F_{15}(Q_5) + \frac{\partial F_{15}}{\partial Q_6} Q_6 - F_{15}(Q_6) + \frac{\partial F_{15}}{\partial Q_{11}} Q_{11} - F_{15}(Q_{11}) +$$
$$\frac{\partial F_{15}}{\partial Q_{14}} Q_{14} - F_{15}(Q_{14}) + \Delta E_7$$

$$F_{16} = \frac{\partial F_{16}}{\partial Q_1} Q_1 - F_{16}(Q_1) + \frac{\partial F_{16}}{\partial Q_5} Q_5 - F(Q_5) + \frac{\partial F_{16}}{\partial Q_6} Q_6 - F_{16}(Q_6) + \frac{\partial F_{16}}{\partial Q_{10}} Q_{10} - F_{16}(Q_{10}) +$$
$$\frac{\partial F_{16}}{\partial Q_{12}} Q_{12} - F_{16}(Q_{12}) + \Delta E_8$$

$$F_{17} = \frac{\partial F_{17}}{\partial Q_1} Q_1 - F_{17}(Q_1) + \frac{\partial F_{17}}{\partial Q_5} Q_5 - F_{17}(Q_5) + \frac{\partial F_{17}}{\partial Q_6} Q_6 - F_{17}(Q_6) + \frac{\partial F_{17}}{\partial Q_{10}} Q_{10} - F_{17}(Q_{10}) +$$
$$\frac{\partial F_{17}}{\partial Q_{13}} Q_{13} - F_{17}(Q_{13}) + \Delta E_9$$

Because the initial flow is needed and it is not required to satisfied the continuity equation, it is based on constant flow velocity, which can be selected.

$$Q_{on} = \pi D_n^2 \qquad \text{n =number of pipe 1,2,3...NP}$$

Based on an arbitrary initial value for the flow rate *Qoi* in each pipe the energy equations are linearized and can be solved using matrix for solving the linear equation.The coefficient matrix is a square matrix with *NP* rows and columns. The coefficient matrix will have the values 0,1,or –1 for the junction equation and the value for the derivative of the energy equations. The row number corresponds to the junction number and the path for the system and the column numbers corresponded to the pipe number. Upon properly defining the coefficient matrix and the known vector, a standard linear algebra is called to solve the linear system.

## 4.8.Computer Program

The computer program is written in *C*, to solve the basic pipe system equations using the linear method just described. Basically the program reads input data defining parameter value for each pipe and pressure and flow specification. The only geometric data input is the connecting node number for each pipe. From this input, the basic system equations are generated and a number of checks are made to make sure the system is geometrically feasible (not disconnected).

Several items should be noted about the program:

- Water is assumed for the liquid unless otherwise specified.
- The Hazen-Williams equation is used for a simulation unless otherwise specified
- A maximum of 20 trial is allowed for simulation unless otherwise specified
- The calculation continues until a relative accuracy of 0.005 is attained unless otherwise specified. The relative accuracy is defined as the sum of the change in flow rate ( absolute) between the last two trial divided by the sum of the flow rate (absolute) .this is given by,

$$\mathrm{Re}\textit{lative.Accuracy} = \frac{\sum Q - Q_I}{\sum Q} < 0.005$$

The basic system equations are solved using linear algebra solver. Appendix (A) A complete listing of the program is given in on the appendix (B)

3. No of pipes
4. No. of junction nodes
5. No. of PRV's
6. Additional data keying various programs option

#### 4.8.2.2.Pipeline Data

1. Connecting node length. Node1#, Node2#.
2. Diameter
3. Roughness
4. Minor loss coefficients
5. Pump power (pump data)
6. Grade (if this connects a FGN)

#### 4.8.2.3.Pump Data

Pump is described by pump data and head-discharge data for three operating points

$$(Q1, H1, Q2, H2, Q3, H3)$$

#### 4.8.2.4.PRV's Data

1. Junction node number for PRV
2. Pipe number –downstream for PRV
3. Grade set by PRV

#### 4.8.2.5.Junction Node Data

1. Demand (Qj)
2. Elevation

3. Junction node number

#### 4.8.2.6.Output Option

1. Key for full or limited output
2. No. of junction nodes summary of maximum and minimum pressures
3. No. of pipes for limited output
4. No. of junction for limited output

### 4.8.3.Discussion of Data Coding

#### 4.8.3.1.Junction Node Data

The only data junction node which is essential to the solution are flow rates entering or leaving the system at junction nodes and this data must be input for all junction nodes with external flows in the system. If pressure calculations are desired, the junction node elevation must be input. However, if there is no demand at a junction node and a pressure calculation for that junction node is not needed then no data input for that junction node is necessary.

#### 4.8.3.2.Output Option

The users select full or limited output. If full output is selected, result for all pipes and junction nodes will be output for all simulation. If limited output is selected, by the user only the results the pipes and junction selected in the summary of the result. The program is also designed to summarise the maximum and minimum pressure in the system and the user can input the number of junction nodes to be included in the summary.

### 4.8.3.3.Data Check

This option will allow the computer to read and check all the input data. This option is useful for checking physical data by hand before going to the expense of the analysis.

### 4.8.3.4.Geometric Data Verification

While inputting a minimum of geometric data is convenient, it creates a situation where checking the geometry described is not possible. The only geometric data, which is input, are the connecting nodes for each pipe, and if this is input incorrectly it is probable that the data will be accepted with the result that the system geometry is incorrectly represented. The computer will detect and identify disconnected system. The input data summary includes a list of pipes connecting each junction node which can be checked by the user against system geometry and, if verified, will assure that the input data for connecting nodes is correct. An option is available for computer verification of this data. If this option is used the computer will check pipes connecting junction nodes generated using the input data against additional data input for this purpose. A successful check of this data will assure the user that this system is geometrically correct. The use of this option is keyed by non-zero entry in the system data and this requires that the connecting pipes at each junction node be input in ascending numerical order on the junction data. A successful verification will produce a verifying message while an unsuccessful verification will produce a message identifying this error and the junction node where it occurred and will suppress the analysis until this discrepancy is removed.

#### 4.8.3.5.Maximum Number of Trials

This limit is set at 20 unless a different limit is specified as the ninth number on the system data. It is unlikely that this limit will ever be reached, but it is imposed to guard against an unforeseen convergence problem (this conceivably could be caused by a check valve or pump operating extremely close to its boundary condition). This option will also allow a small number of trials to be run if desired.

#### 4.8.3.6.Relative Accuracy

This parameter determines when the solution is accepted. It is defined as the total (absolute) change in flow rate in the pipes from the previous trial divided by the total (absolute) flow rate and is set at 0.005 unless this option is employed to change this value. Inserting the desired relative accuracy as the tenth number on the system data does this. If this field is left blank the value of 0.005 is used which provides an extremely accurate result. A summary of the result of many problems indicates that an average of six trials will produce this accuracy although more (or less) may be required for certain problems. It is unlikely that the user will want to exercise this option and change the relative accuracy and such a change is not recommended.

#### 4.8.3.7.Specific Gravity of Liquid

Unless specified by the user, water is assumed to be the liquid being transported. Other liquids are considered by inserting a non-zero entry as the eleventh number on the system data. This is the specific gravity of the liquid being considered (ratio of liquid density to water density).

#### 4.8.3.8.Kinematics Viscosity of Liquid

A non-zero input in this field for this parameter (system data) allows the use of the Darcy-Weisbach equation for head loss calculation and the kinematics viscosity, which is needed to employ this relationship. If the filed is left blank the Hazen-Williams head loss equation is used which is appropriate for water distribution system. For other liquids (and for water, if desired) the Darcy-Weisbach equation should be used and this option requires the input of the value for the kinematics viscosity (in ft^2/s or m^2/s for the SI units).

#### 4.8.3.9.Non-consecutive Pipe Numbering

Non-consecutive numbering of the junction nodes is always acceptable. However, it is assumed that the pipes are numbered 1-p and the data is input in this order unless this option is employed. Using option which is keyed by a non-zero entry in the system data a pipe number is input ending in the pipe line data for each pipe. This pipe number used for all subsequent input-output operation.

#### 4.8.3.10.Pump Operating Data Input

The description of a pump by three points of operating data (pairs of head-discharge data) is keyed by a negative one (-1) entry as the eighth number on the pipe line data. The three points should represent a wide variety of operating data. This is a very effective means of describing a pump and also accounts for out of range pump operation. The input data defines the normal range of operation as well as providing data within that range. A representative useful power will be computed from the operating data if a negative two (-2) is used, and this feature is useful if convergence problems are noted.

### 4.8.3.11.Geometric Data Generation and Input

Each time a simulation is made computer time is used to generate the loops and path for energy equations and this produces additional time consumption for large systems. In order to avoid this an option allows the user to generate this data as input data

### 4.8.4.Computer Messages and Warning

A number of checks of geometric conditions and other data input are made and a message is produced if an error is detected. The following is a list of messages, which is a result of a condition, which makes an analysis impossible, and execution termination results.

### 4.8.4.1.System is Disconnected-Check-Input Data

This message means that part of the system is not connected to the rest by a pipe section, which is not acceptable and probably caused, by incorrect input data for connecting nodes or, perhaps, the missing of a pipe. The input data summary must be checked and the data corrected.

### 4.8.4.2.Data Input for Pipes Connecting Junction –Do not Check

This message is produced if the data input for connecting pipes at the indicated junctions nodes does not check the data generated from connecting node input.

### 4.8.4.3.The Relation P=J+L+T-1 is not Satisfied –Check Input Data

The computer uses the connecting node data input for the pipe to generate Loop and path data. If this basic geometric relationship is not satisfied after generation of the loop and path data this message is generated. It is likely that one of the earlier messages connecting node data will also be generated if these conditions accrue. The input summary must be checked, particularly the connecting pipes of the junction, and the data need to be corrected.

# Examples

This application is the principal one, which refers to steady state analysis of pressure and flow in piping system for given set of conditions. The user via the input data can choose a number of different sets of conditions. Some examples are provided to illustrate a variety of program applications for regular simulation. The results for these examples are discussed in this section.

Example 1:

Figure represents a small distribution system, which transports water fed by a large main, which maintains a pressure of 60 psig. The piping system carries water to storage tank and also discharges water at four other locations. The simulation is carried out to determine how much water can be delivered at specified pressure. For this simulation the source of water and all the delivery points are treated as fixed grade nodes with total grades known (elevation plus pressure head). The grade at the source (60 psig at elevation of 50 ft) is for example, computed as 50+60*144/62.8=188.46.and all grades are computed in this manner. For different lengths of pipe, 1 find the discharge in each pipe and the head at all node and the change in head loss for each pipe in the system.

The chart in Figure 4.22 can show the effect on each node and the head loss in the system by changing the length of pipe, 1. The pressure and the head at node (1) is affected more by changing the length of pipe (1) Node if slightly affected by this change and higher head loss occur in pipe one.

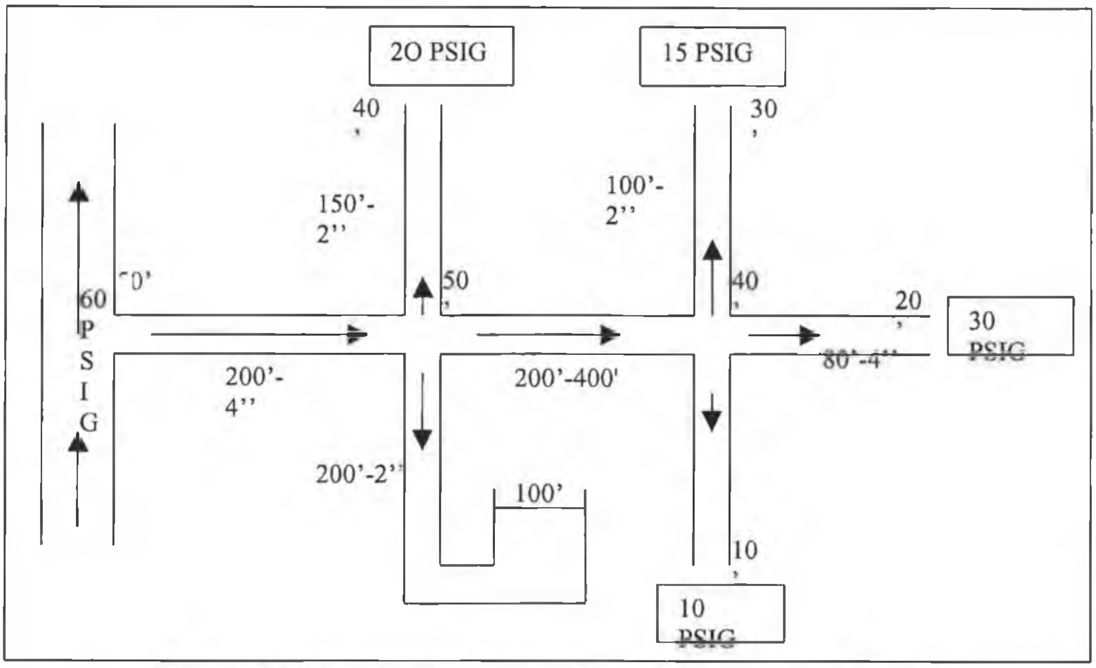


Figure 4.22 distribution system fed by large pressure main

The input and out put file for example 1 is listed in file name (EXAMOUT1-EXAMOUT2).

Example 2:

For the small network below do the following:

1. For the specified physical system, find the discharge in each pipe and the head at all nodes.
2. Find the head that the pump must produce so that the discharge through pipe 5 into the reservoir is $Q_5 = 1.0 ft^3/s$

| Q | 4.5 | 4.0 | 3.5 |
|---|---|---|---|
| Hp | 54 | 50 | 44 |

The 8 unknowns are $Q_1, Q_2, Q_3, Q_4, Q_5, H_1, H_2, H_3$. The solution is as follows. This can be regarded as the solution to an analysis problem since all of the physical features of the networks are known, and the solution describes the performance of this existing network in response to the specified demands. The chart in figure 4.23 shows the head and HGL at each node for different elevations. It is noted that the head at the junction is equal to the HGL-elevation when the elevation at the junction equals to zero

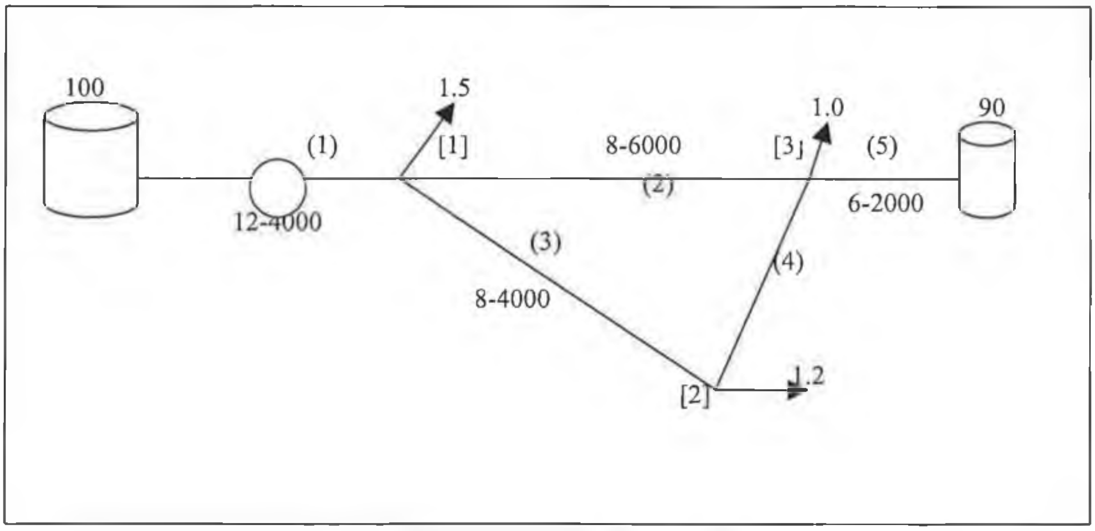


Figure 4.23 Small network with 2 reservoirs and 4 pipes, 3 junction

The input and out put file for example 1 is listed in file name (EXAMIN8-EXAMOP9).

Example 3:

Figure 4.24 depicts a network with 19 pipes and 12 nodes. The nodal demands sum to 10ft^3/s, and this charge must come from two reservoirs .the steady sate solution is listed in file EXAMOP5. All pipe diameters are in inches and lengths in feet. The largest head loss, 24.3 ft, occurs in pipe 1 that supplies the network from a reservoir, and the third largest head loss, 21.7 ft occurs in the other reservoir supply line in pipe 5. The number of simultaneous equations to model this steady flow problem is 19. Six of them real loop energy equations and one is a pseudo loop that connects the two reservoirs.

The solution of the steady problem will be sought for all the demands. At node 9 the demand gradually increases from 1.2 to 1.5 ft3/s.

Results from steady flow are shown in file EXAMOP5. The results from this solution are also compared for each step. It is clear that the pressure at node 9-5-10-11-12 one affected more by changing the demand at node 9 than that of other node in the system and the flow in pipe which is very close to the nodes [13-14-15-16-17-18-19] increase.

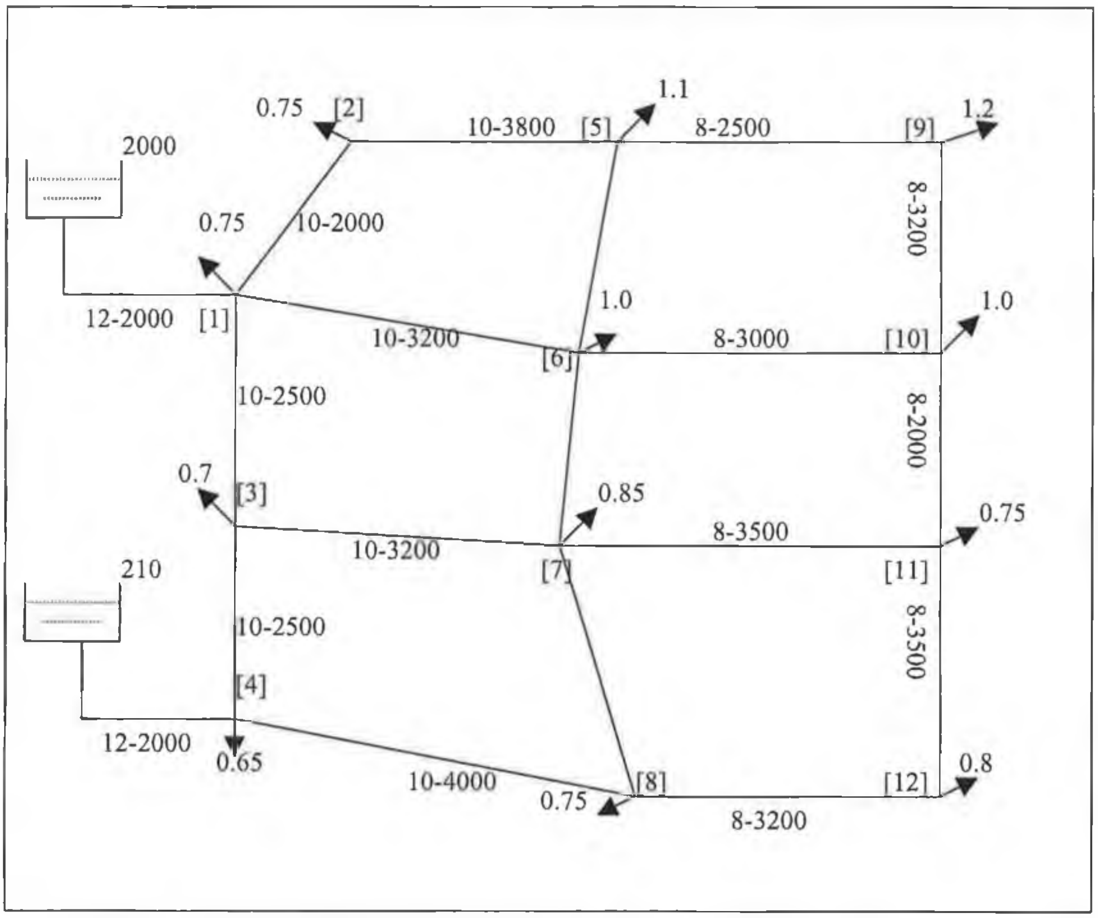


Figure 4.24 system with 2 reservoirs and 19 pipes 12 node

The input and out put file for example 1 is listed in file name (EXAMOUT5-EXAMOP5).

Example 4:

The pipe and other data for 14-pipe network supply with reservoirs are given. A solution is obtained using the input data for the following cases.

1. Finds the heads at all nodes as well as the discharges in all 14-pipes.
2. All the demands are considered unknown, and the heads are given in the input data.
3. Find the solution with a pump inserted in line 14 and a specified reservoir water surface elevation.
4. The solution with different water surface elevation in tank 1.from 2650-2550.

The results are shown in figures 4.25 and it is noted here that the demand at node [1] is =0 when the water surface elevation =2587.58 ft. In this point either one has to supply the junction [1] or keep the water surface elevation over this limit to avoid the flow in the opposite direction. From the result in the table it is clear that when the water surface level changes, the head loss in pipe one also changes till the head at node [1] get value higher than the water level in the tank. At this point supply to the system.

64.144+2548.6=2612.744<2650 (water surface)

51.418+2548.6=2600.00==2600 (water surface)

48.634+2548.6=2597.00>2587.58 (water surface)

40.886+2548.6=2589.486>2550 (water surface)

The head at node [1] is equal to the water surface level at 2600 feet.

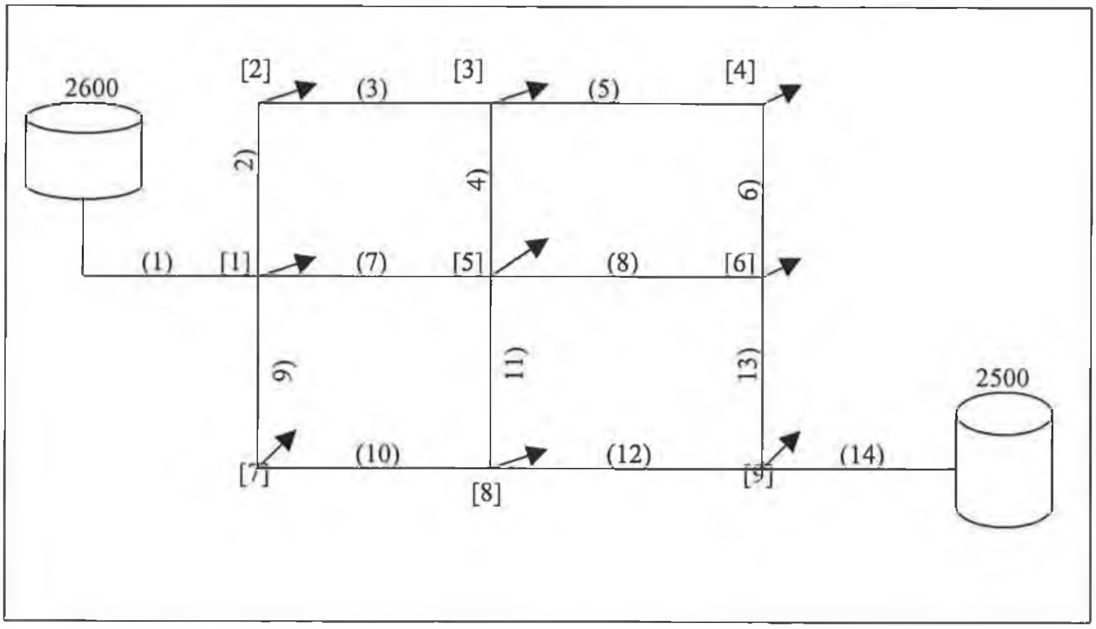


Figure 4.25 pipe network with 14 pipe and 9 junction

The input and out put file for example 1 is listed in file name (EXAMIN-EXAMPLE1).

# File Input & Output
# Example [1]

```
7 2 6 0
1 188.46 2 86.46 4 64.62 5 100.0 6 33.08 7 89.23
0 1 300 0.333 110 0.349
0 1 150 0.166 110 0.0872
1 2 200 0.333 110 0.349
0 2 100 0.166 110 0.0872
0 1 200 0.166 110 0.0872
0 2 300 0.166 110 0.0872
0 2 80 0.333 110 0.349

0 130.43 50
0 97.44 40
```

```
Pipe Data:
-------------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                       Coef.
-------------------------------------------------------------------
   1    0    1     200.0     0.333 110.000000     1.374    59.830
   2    0    1     150.0     0.166 110.000000    -0.213   -42.171
   3    1    2     200.0     0.333 110.000000     1.013    34.032
   4    0    2     100.0     0.166 110.000000    -0.221   -29.980
   5    0    1     200.0     0.166 110.000000    -0.148   -28.631
   6    0    2     300.0     0.166 110.000000    -0.180   -61.519
   7    0    2      80.0     0.333 110.000000    -0.613    -5.420
Node Data:
--------------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
--------------------------------------------------------

    1      0.000    50.000    78.631    34.073   128.631
    2      0.000    40.000    54.600    23.660    94.600
Pipe Data:
-------------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                       Coef.
-------------------------------------------------------------------
   1    0    1     100.0     0.333 110.000000     1.665    42.654
   2    0    1     150.0     0.166 110.000000    -0.256   -59.349
   3    1    2     200.0     0.333 110.000000     1.218    47.817
   4    0    2     100.0     0.166 110.000000    -0.234   -33.376
   5    0    1     200.0     0.166 110.000000    -0.191   -45.808
   6    0    2     300.0     0.166 110.000000    -0.185   -64.915
   7    0    2      80.0     0.333 110.000000    -0.799    -8.848
Node Data:
--------------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
--------------------------------------------------------

    1      0.000    50.000    95.809    41.517   145.809
    2      0.000    40.000    57.996    25.132    97.996
Pipe Data:
-------------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                       Coef.
-------------------------------------------------------------------
   1    0    1     300.0     0.333 110.000000     1.193    69.043
   2    0    1     150.0     0.166 110.000000    -0.186   -32.957
   3    1    2     200.0     0.333 110.000000     0.886    26.562
   4    0    2     100.0     0.166 110.000000    -0.214   -28.236
   5    0    1     200.0     0.166 110.000000    -0.120   -19.417
   6    0    2     300.0     0.166 110.000000    -0.177   -59.776
   7    0    2      80.0     0.333 110.000000    -0.496    -3.660
Node Data:
--------------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
--------------------------------------------------------

    1      0.000    50.000    69.418    30.081   119.418
    2      0.000    40.000    52.856    22.904    92.856
```

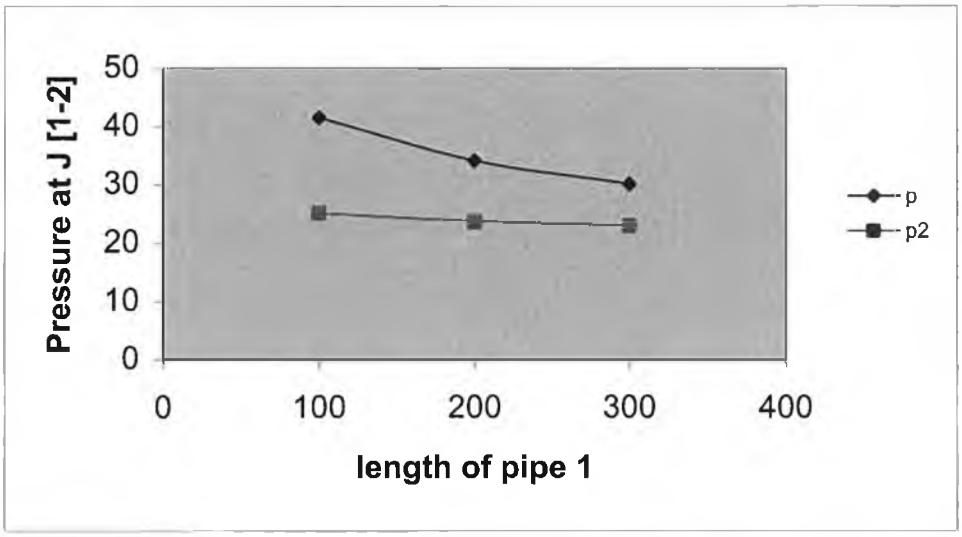


Figure.4.A1

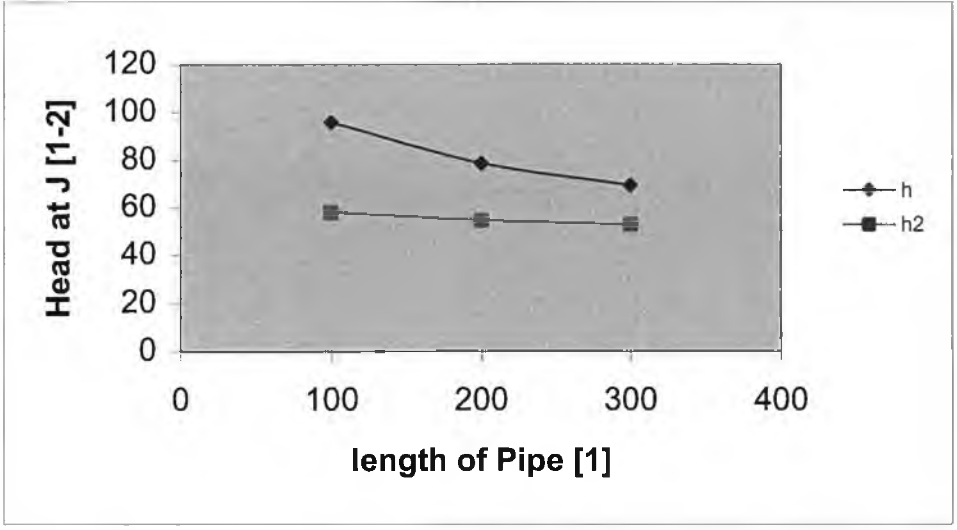


Figure 4.A2

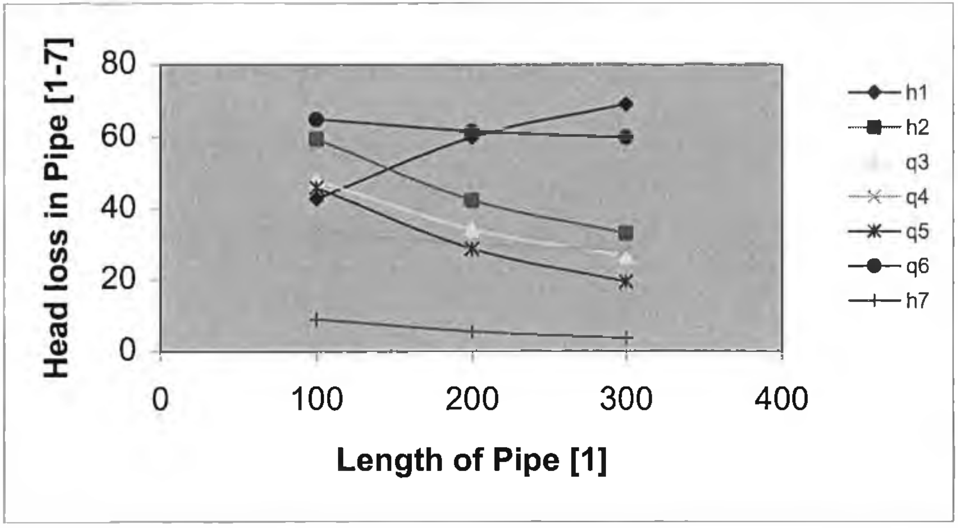


Figure 4.A3

# File Input & Output
# Example [2]

```
5 4 1 0
5 90
4 1 4000 1.0 .000167 4.2
1 3 6000 .667 .000167 1.3
1 2 4000 .667 .000167 1.5
2 3 3000 .500 .000167 0.5
0 3 2000 .5 .000167 1.0
1.5 126 0
1.2 98 0
1.0 95 0
-4.7 130 0
```

```
5 4 1 0
5 90
4 1 4000 1.0 .000167 4.2
1 3 6000 .667 .000167 1.3
1 2 4000 .667 .000167 1.5
2 3 3000 .500 .000167 0.5
0 3 2000 .5 .000167 1.0
1.5 126 50
1.2 98 40
1.0 95 30
-4.7 130 20
```

```
Devices caused the following changes in heads
Device   1 in pipe    1 Change in head =   50.985
Pipe Data:
-----------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                       Coef.
-----------------------------------------------------------------
   1    0    1    4000.0     1.000  0.000167     4.103    26.459
   2    1    3    6000.0     0.667  0.000167     1.192    29.129
   3    1    2    4000.0     0.667  0.000167     1.411    26.625
   4    2    3    3000.0     0.500  0.000167     0.211     2.503
   5    0    3    2000.0     0.500  0.000167    -0.403    -5.447
Node Data:
-------------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
-------------------------------------------------------

    1      1.500     0.000   124.526    53.961   124.526
    2      1.200     0.000    97.901    42.424    97.901
    3      1.000     0.000    95.398    41.339    95.398
Devices caused the following changes in heads
Device   1 in pipe    1 Change in head =   47.013
Pipe Data:
-----------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                       Coef.
-----------------------------------------------------------------
   1    0    1    4000.0     1.000  0.000167     3.730    27.327
   2    1    3    6000.0     0.667  0.000167     1.232    31.000
   3    1    2    4000.0     0.667  0.000167     1.450    28.000
   4    2    3    3000.0     0.500  0.000167     0.233     3.000
   5    0    3    2000.0     0.500  0.000167    -0.386    -5.046
Node Data:
-------------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
-------------------------------------------------------

    1      1.049     0.000   126.000    54.600   126.000
    2      1.216     0.000    98.000    42.467    98.000
    3      1.079     0.000    95.000    41.167    95.000
Pipe Data:
-----------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                       Coef.
-----------------------------------------------------------------
   1    4    1    4000.0     1.000  0.000167     4.700    34.213
   2    1    3    6000.0     0.667  0.000167     1.536    46.799
   3    1    2    4000.0     0.667  0.000167     1.664    36.285
   4    2    3    3000.0     0.500  0.000167     0.464    10.513
   5    0    3    2000.0     0.500  0.000167    -1.000   -29.499
Node Data:
-------------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
-------------------------------------------------------

    1      1.500     0.000   166.022    71.943   166.022
    2      1.200     0.000   129.736    56.219   129.736
    3      1.000     0.000   119.223    51.663   119.223
    4     -4.700     0.000   200.234    86.768   200.234
Pipe Data:
-----------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
```

```
  No.  #1   #2                         Coef.
------------------------------------------------------------------
   1    4    1    4000.0     1.000  0.000167      1.488     4.000
   2    1    3    6000.0     0.667  0.000167      1.232    31.000
   3    1    2    4000.0     0.667  0.000167      1.450    28.000
   4    2    3    3000.0     0.500  0.000167      0.233     3.000
   5    0    3    2000.0     0.500  0.000167     -0.386    -5.046
Node Data:
-------------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
-------------------------------------------------------

    1     -1.194     0.000   126.000    54.600   126.000
    2      1.216     0.000    98.000    42.467    98.000
    3      1.079     0.000    95.000    41.167    95.000
    4     -1.488     0.000   130.000    56.333   130.000
Pipe Data:
------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                         Coef.
------------------------------------------------------------------
   1    4    1    4000.0     1.000  0.000167      4.700    34.213
   2    1    3    6000.0     0.667  0.000167      1.536    46.799
   3    1    2    4000.0     0.667  0.000167      1.664    36.285
   4    2    3    3000.0     0.500  0.000167      0.464    10.513
   5    0    3    2000.0     0.500  0.000167     -1.000   -29.499
Node Data:
-------------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
-------------------------------------------------------

    1      1.500    50.000   116.022    50.276   166.022
    2      1.200    40.000    89.736    38.886   129.736
    3      1.000    30.000    89.223    38.663   119.223
    4     -4.700    20.000   180.234    78.102   200.234
Pipe Data:
------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                         Coef.
------------------------------------------------------------------
   1    4    1    4000.0     1.000  0.000167      1.488     4.000
   2    1    3    6000.0     0.667  0.000167      1.232    31.000
   3    1    2    4000.0     0.667  0.000167      1.450    28.000
   4    2    3    3000.0     0.500  0.000167      0.233     3.000
   5    0    3    2000.0     0.500  0.000167     -0.386    -5.046
Node Data:
-------------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
-------------------------------------------------------

    1     -1.194    50.000    76.000    32.933   126.000
    2      1.216    40.000    58.000    25.133    98.000
    3      1.079    30.000    65.000    28.167    95.000
    4     -1.488    20.000   110.000    47.667   130.000
```

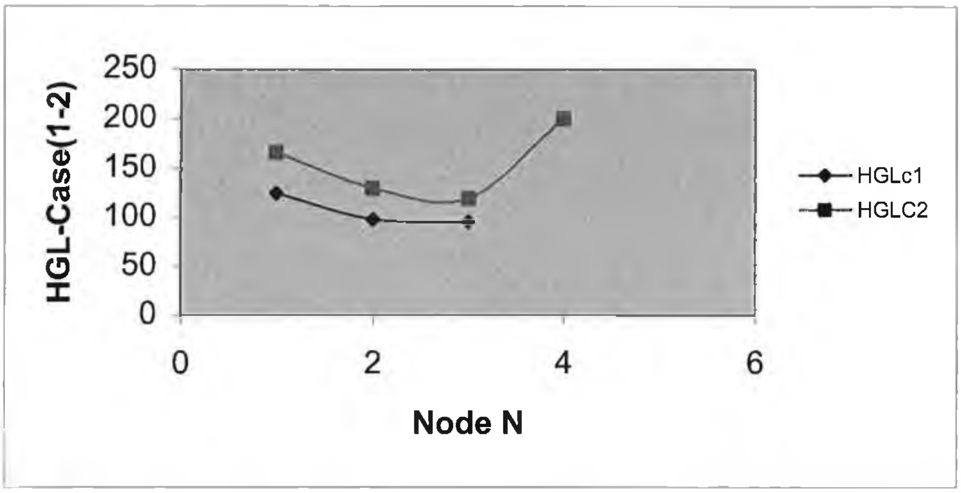


Figure 4.A4

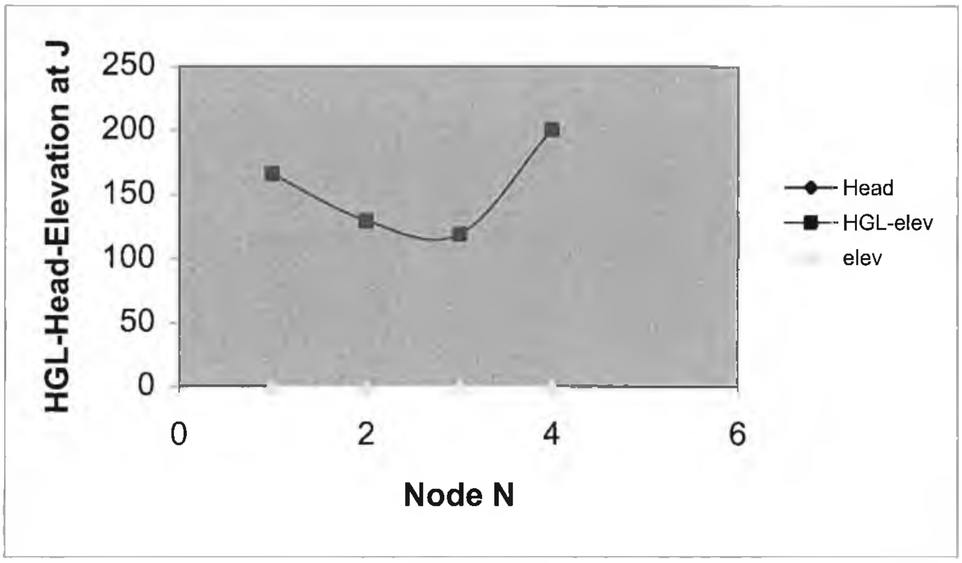


Figure 4.A5

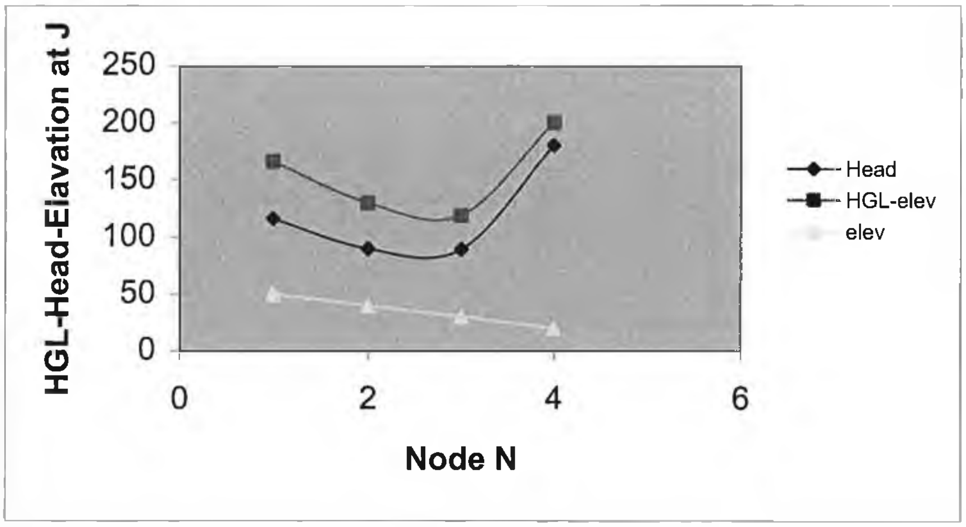


Figure 4.A6

# File Input & Output
# Example [3]

```
19 12 2 0
1 200 5 210
0 1 2000 1 0.000417 1.1
1 2 2000 .8333 .000417 1.1
1 3 2500 .8333 .000417 1.1
4 3 2500 .8333 .000417 1.1
0 4 2000 1 0.000417 1.1
2 5 3800 .8333 .000417 1.1
1 6 3500 .8333 .000417 1.1
3 7 3200 .8333 .000417 1.1
4 8 4000 .8333 .000417 1.1
6 5 3500 .6666 .000417 1.1
7 6 2500 .6666 .000417 1.1
8 7 3800 .6666 .000417 1.1
5 9 2500 .6666 .000417 1.1
6 10 3000 .6666 .000417 1.1
7 11 3500 .6666 .000417 1.1
8 12 3200 .6666 .000417 1.1
10 9 3200 .6666 .000417 1.1
11 10 2000 .6666 .0000417 1.1
12 11 3500 .6666 .000417 1.1

0.75 175.74 0.0
0.75 164.78 0.0
0.70 175.14 0.0
0.65 188.33 0.0
1.10 155.49 0.0
1.00 159.79 0.0
0.85 162.44 0.0
0.75 165.44 0.0
2.50 148.15 0.0
1.00 149.74 0.0
0.75 151.15 0.0
0.80 152.11 0.0
```

```
Pipe Data:
-------------------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
 No.   #1   #2                         Coef.
-------------------------------------------------------------------------------
   1    0    1     2000.0     1.000  0.000417     5.300    24.270
   2    1    2     2000.0     0.833  0.000417     2.171    10.960
   3    1    3     2500.0     0.833  0.000417     0.407     0.599
   4    4    3     2500.0     0.833  0.000417     2.128    13.189
   5    0    4     2000.0     1.000  0.000417     5.000    21.681
   6    2    5     3800.0     0.833  0.000417     1.421     9.289
   7    1    6     3500.0     0.833  0.000417     1.972    15.949
   8    3    7     3200.0     0.833  0.000417     1.835    12.720
   9    4    8     4000.0     0.833  0.000417     2.222    22.915
  10    6    5     3500.0     0.667  0.000417     0.544     4.300
  11    7    6     2500.0     0.667  0.000417     0.501     2.630
  12    8    7     3800.0     0.667  0.000417     0.428     2.994
  13    5    9     2500.0     0.667  0.000417     0.866     7.327
  14    6   10     3000.0     0.667  0.000417     0.928    10.029
  15    7   11     3500.0     0.667  0.000417     0.913    11.349
  16    8   12     3200.0     0.667  0.000417     1.043    13.364
  17   10    9     3200.0     0.667  0.000417     0.334     1.598
  18   11   10     2000.0     0.667  0.000042     0.407     1.310
  19   12   11     3500.0     0.667  0.000417     0.243     0.988
Node Data:
-------------------------------------------------------
  Node     Demand Elevation       Head   Pressure HGL-elev.
-------------------------------------------------------


    1      0.750     0.000    175.730     76.150   175.730
    2      0.750     0.000    164.770     71.400   164.770
    3      0.700     0.000    175.131     75.890   175.131
    4      0.650     0.000    188.319     81.605   188.319
    5      1.100     0.000    155.480     67.375   155.480
    6      1.000     0.000    159.781     69.238   159.781
    7      0.850     0.000    162.410     70.378   162.410
    8      0.750     0.000    165.404     71.675   165.404
    9      1.200     0.000    148.153     64.200   148.153
   10      1.000     0.000    149.751     64.892   149.751
   11      0.750     0.000    151.061     65.460   151.061
   12      0.800     0.000    152.040     65.884   152.040
Pipe Data:
-------------------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
 No.   #1   #2                         Coef.
-------------------------------------------------------------------------------
   1    0    1     2000.0     1.000  0.000417     5.481    25.908
   2    1    2     2000.0     0.833  0.000417     2.261    11.843
   3    1    3     2500.0     0.833  0.000417     0.418     0.629
   4    4    3     2500.0     0.833  0.000417     2.183    13.847
   5    0    4     2000.0     1.000  0.000417     5.119    22.690
   6    2    5     3800.0     0.833  0.000417     1.511    10.429
   7    1    6     3500.0     0.833  0.000417     2.052    17.222
   8    3    7     3200.0     0.833  0.000417     1.901    13.603
   9    4    8     4000.0     0.833  0.000417     2.286    24.198
  10    6    5     3500.0     0.667  0.000417     0.594     5.051
  11    7    6     2500.0     0.667  0.000417     0.537     2.989
  12    8    7     3800.0     0.667  0.000417     0.448     3.252
  13    5    9     2500.0     0.667  0.000417     1.004     9.707
  14    6   10     3000.0     0.667  0.000417     0.995    11.455
  15    7   11     3500.0     0.667  0.000417     0.963    12.543
  16    8   12     3200.0     0.667  0.000417     1.088    14.467
  17    10     9    3200.0     0.667  0.000417     0.496      3.304
  18    11    10    2000.0     0.667  0.000042     0.500      1.901
  19    12    11    3500.0     0.667  0.000417     0.288      1.341
Node Data:
------------------------------------------------------
 Node     Demand Elevation       Head  Pressure HGL-elev.
------------------------------------------------------

    1      0.750     0.000   174.092    75.440   174.092
    2      0.750     0.000   162.249    70.308   162.249
    3      0.700     0.000   173.463    75.167   173.463
    4      0.650     0.000   187.310    81.167   187.310
    5      1.100     0.000   151.819    65.788   151.819
    6      1.000     0.000   156.870    67.977   156.870
    7      0.850     0.000   159.860    69.272   159.860
    8      0.750     0.000   163.111    70.682   163.111
    9      1.500     0.000   142.112    61.582   142.112
   10      1.000     0.000   145.416    63.013   145.416
   11      0.750     0.000   147.316    63.837   147.316
   12      0.800     0.000   148.645    64.413   148.645
Pipe Data:
-------------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
 No.   #1   #2                        Coef.
-------------------------------------------------------------------------
   1     0     1    2000.0     1.000  0.000417     5.632     27.314
   2     1     2    2000.0     0.833  0.000417     2.337     12.623
   3     1     3    2500.0     0.833  0.000417     0.426      0.650
   4     4     3    2500.0     0.833  0.000417     2.229     14.414
   5     0     4    2000.0     1.000  0.000417     5.218     23.549
   6     2     5    3800.0     0.833  0.000417     1.587     11.453
   7     1     6    3500.0     0.833  0.000417     2.119     18.315
   8     3     7    3200.0     0.833  0.000417     1.955     14.349
   9     4     8    4000.0     0.833  0.000417     2.339     25.281
  10     6     5    3500.0     0.667  0.000417     0.637      5.761
  11     7     6    2500.0     0.667  0.000417     0.567      3.317
  12     8     7    3800.0     0.667  0.000417     0.465      3.483
  13     5     9    2500.0     0.667  0.000417     1.124     12.023
  14     6    10    3000.0     0.667  0.000417     1.050     12.679
  15     7    11    3500.0     0.667  0.000417     1.003     13.548
  16     8    12    3200.0     0.667  0.000417     1.124     15.386
  17    10     9    3200.0     0.667  0.000417     0.626      5.104
  18    11    10    2000.0     0.667  0.000042     0.576      2.448
  19    12    11    3500.0     0.667  0.000417     0.324      1.661
Node Data:
------------------------------------------------------
 Node     Demand Elevation       Head  Pressure HGL-elev.
------------------------------------------------------

    1      0.750     0.000   172.686    74.831   172.686
    2      0.750     0.000   160.063    69.361   160.063
    3      0.700     0.000   172.036    74.549   172.036
    4      0.650     0.000   186.451    80.795   186.451
    5      1.100     0.000   148.610    64.398   148.610
    6      1.000     0.000   154.371    66.894   154.371
    7      0.850     0.000   157.688    68.331   157.688
    8      0.750     0.000   161.170    69.840   161.170
    9      1.750     0.000   136.587    59.188   136.587
   10      1.000     0.000   141.692    61.400   141.692
   11      0.750     0.000   144.139    62.460   144.139
   12      0.800     0.000   145.785    63.173   145.785
Pipe Data:
```

```
-----------------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
 No.   #1   #2                      Coef.
-----------------------------------------------------------------------
   1    0    1    2000.0     1.000  0.000417     5.783    28.756
   2    1    2    2000.0     0.833  0.000417     2.415    13.440
   3    1    3    2500.0     0.833  0.000417     0.432     0.668
   4    4    3    2500.0     0.833  0.000417     2.276    14.999
   5    0    4    2000.0     1.000  0.000417     5.317    24.425
   6    2    5    3800.0     0.833  0.000417     1.665    12.540
   7    1    6    3500.0     0.833  0.000417     2.186    19.440
   8    3    7    3200.0     0.833  0.000417     2.008    15.104
   9    4    8    4000.0     0.833  0.000417     2.391    26.377
  10    6    5    3500.0     0.667  0.000417     0.681     6.539
  11    7    6    2500.0     0.667  0.000417     0.599     3.668
  12    8    7    3800.0     0.667  0.000417     0.482     3.727
  13    5    9    2500.0     0.667  0.000417     1.246    14.637
  14    6   10    3000.0     0.667  0.000417     1.104    13.943
  15    7   11    3500.0     0.667  0.000417     1.042    14.569
  16    8   12    3200.0     0.667  0.000417     1.159    16.310
  17   10    9    3200.0     0.667  0.000417     0.754     7.232
  18   11   10    2000.0     0.667  0.000042     0.650     3.042
  19   12   11    3500.0     0.667  0.000417     0.359     2.005
Node Data:
------------------------------------------------------------
 Node     Demand Elevation       Head  Pressure HGL-elev.
------------------------------------------------------------


   1      0.750     0.000    171.244    74.206   171.244
   2      0.750     0.000    157.804    68.382   157.804
   3      0.700     0.000    170.576    73.916   170.576
   4      0.650     0.000    185.576    80.416   185.576
   5      1.100     0.000    145.264    62.948   145.264
   6      1.000     0.000    151.804    65.782   151.804
   7      0.850     0.000    155.472    67.371   155.472
   8      0.750     0.000    159.199    68.986   159.199
   9      2.000     0.000    130.628    56.605   130.628
  10      1.000     0.000    137.860    59.739   137.860
  11      0.750     0.000    140.903    61.058   140.903
  12      0.800     0.000    142.889    61.919   142.889
Pipe Data:
-----------------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
 No.   #1   #2                      Coef.
-----------------------------------------------------------------------
   1    0    1    2000.0     1.000  0.000417     6.085    31.747
   2    1    2    2000.0     0.833  0.000417     2.572    15.178
   3    1    3    2500.0     0.833  0.000417     0.442     0.698
   4    4    3    2500.0     0.833  0.000417     2.371    16.220
   5    0    4    2000.0     1.000  0.000417     5.515    26.225
   6    2    5    3800.0     0.833  0.000417     1.822    14.898
   7    1    6    3500.0     0.833  0.000417     2.320    21.785
   8    3    7    3200.0     0.833  0.000417     2.113    16.652
   9    4    8    4000.0     0.833  0.000417     2.495    28.619
  10    6    5    3500.0     0.667  0.000417     0.773     8.290
  11    7    6    2500.0     0.667  0.000417     0.663     4.436
  12    8    7    3800.0     0.667  0.000417     0.518     4.253
  13    5    9    2500.0     0.667  0.000417     1.495    20.755
  14    6   10    3000.0     0.667  0.000417     1.210    16.606
  15    7   11    3500.0     0.667  0.000417     1.118    16.671
  16    8   12    3200.0     0.667  0.000417     1.227    18.187
  17   10    9    3200.0     0.667  0.000417     1.005    12.439
```

```
   18    11    10     2000.0      0.667  0.000042      0.795      4.371
   19    12    11     3500.0      0.667  0.000417      0.427      2.762
 Node Data:
 -------------------------------------------------------
  Node     Demand Elevation      Head  Pressure HGL-elev.
 -------------------------------------------------------


     1      0.750     0.000   168.252    72.909   168.252
     2      0.750     0.000   153.075    66.332   153.075
     3      0.700     0.000   167.555    72.607   167.555
     4      0.650     0.000   183.775    79.636   183.775
     5      1.100     0.000   138.177    59.877   138.177
     6      1.000     0.000   146.467    63.469   146.467
     7      0.850     0.000   150.903    65.391   150.903
     8      0.750     0.000   155.156    67.234   155.156
     9      2.500     0.000   117.422    50.883   117.422
    10      1.000     0.000   129.861    56.273   129.861
    11      0.750     0.000   134.232    58.167   134.232
    12      0.800     0.000   136.969    59.353   136.969
```

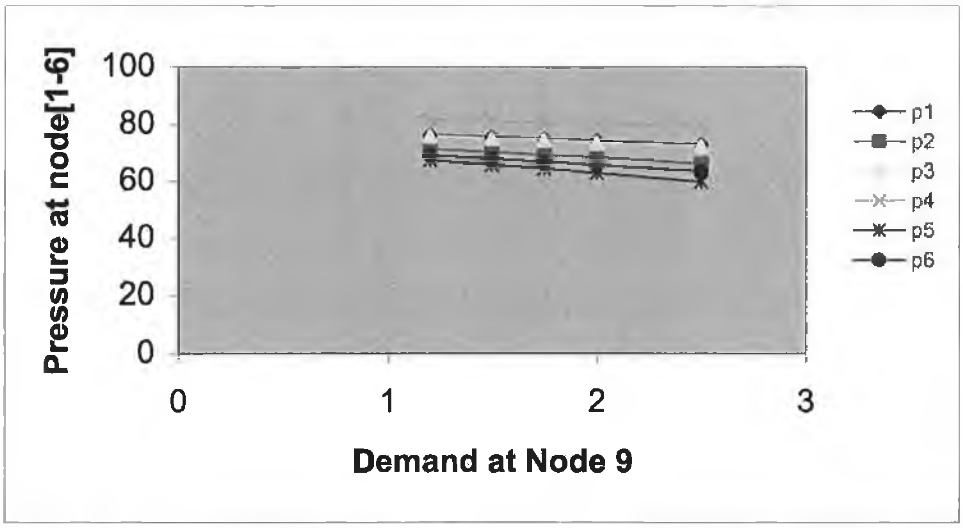


Figure 4.A Pressure at Node [1-2-3-4-5-6]

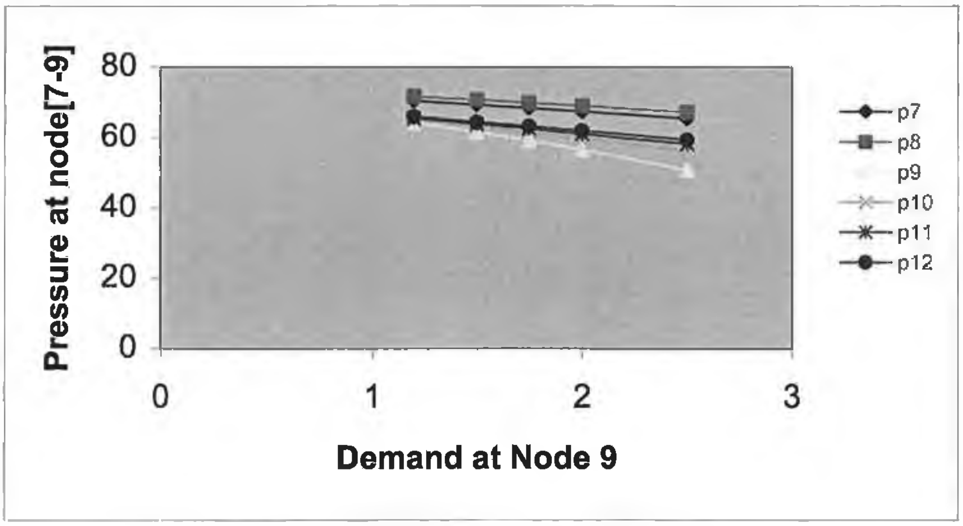


Figure 4.A8 Pressure at Node [7-8-9-10-11-12]

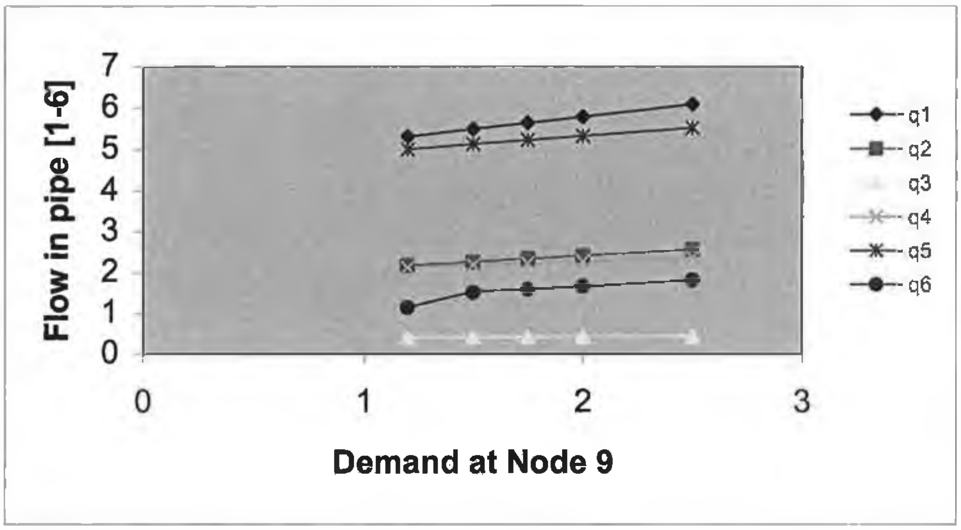


Figure 4.A9 Flow in Pipe [1-2-3-4-5-6]

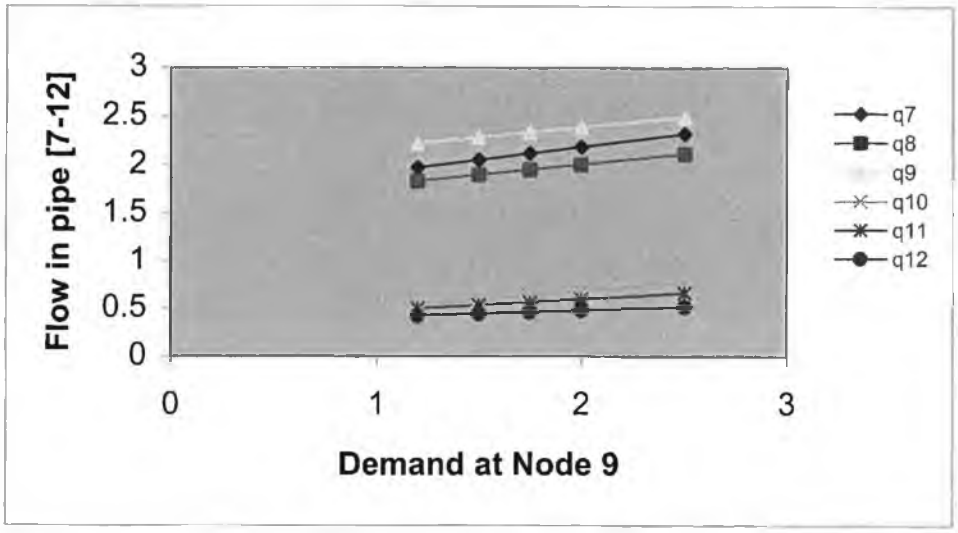


Figure 4.A10 Flow in Pipe[7-8-9-10-11-12]

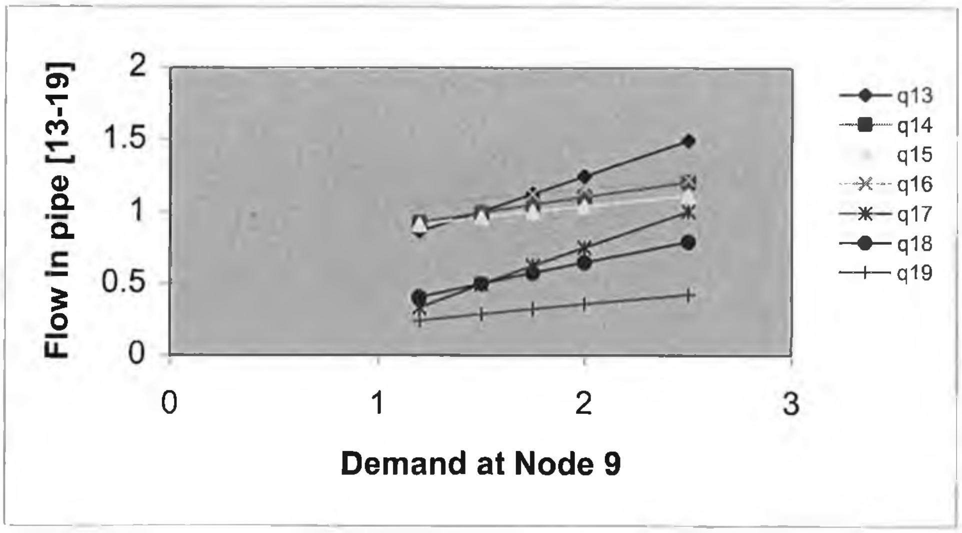


Figure 4.A11 Flow in pipe [13-14-15-16-17-18-19]

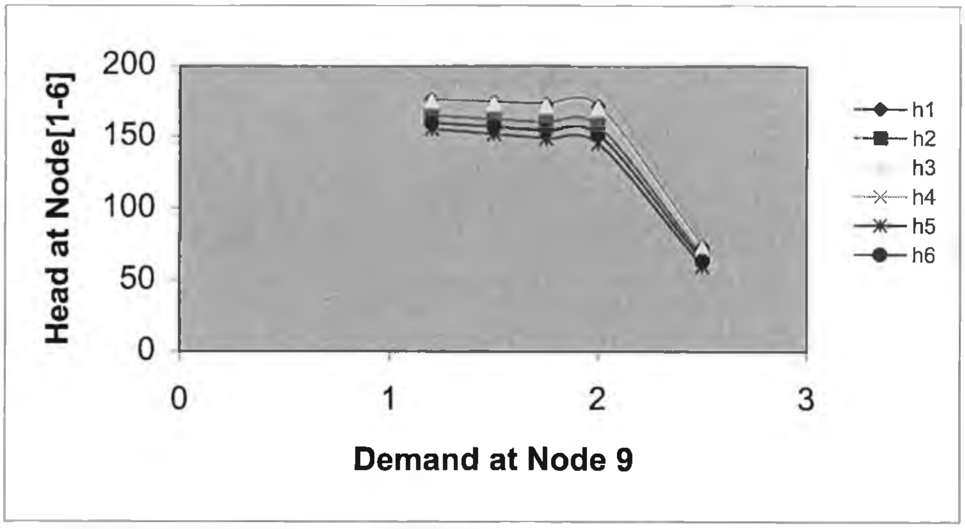


Figure 4.A12 Head at Junction [1-2-3-4-5-6]

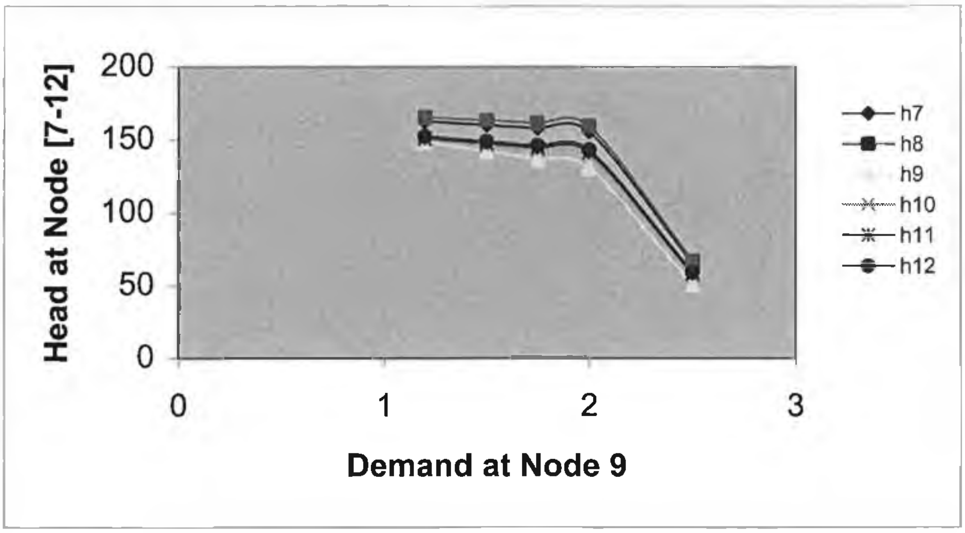


Figure 4.A13 Head at Junction [7-8-9-10-11-12]

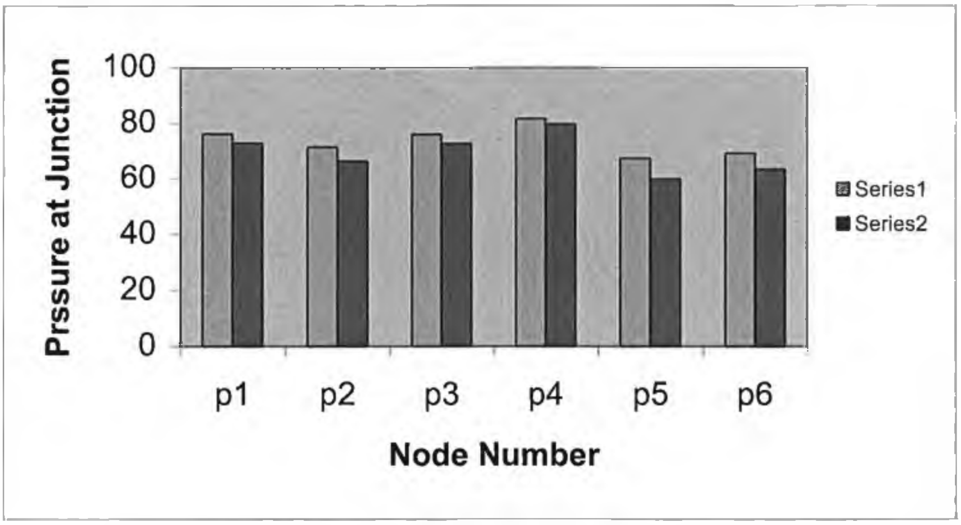


Figure 4.A14 The effect by Changing Q9 on Pressure at [1-6]

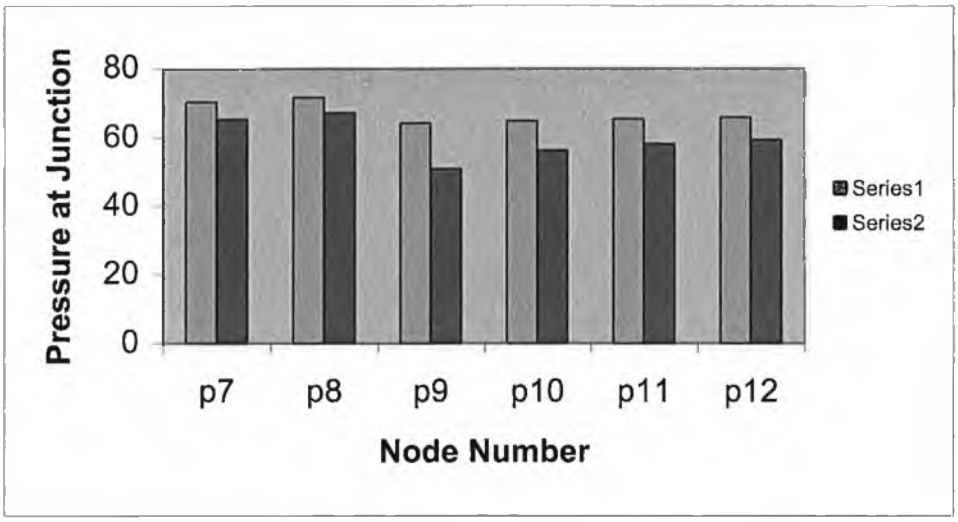


Figure 4.A15 The effect by Changing Qj9 on ressure at[7-12]

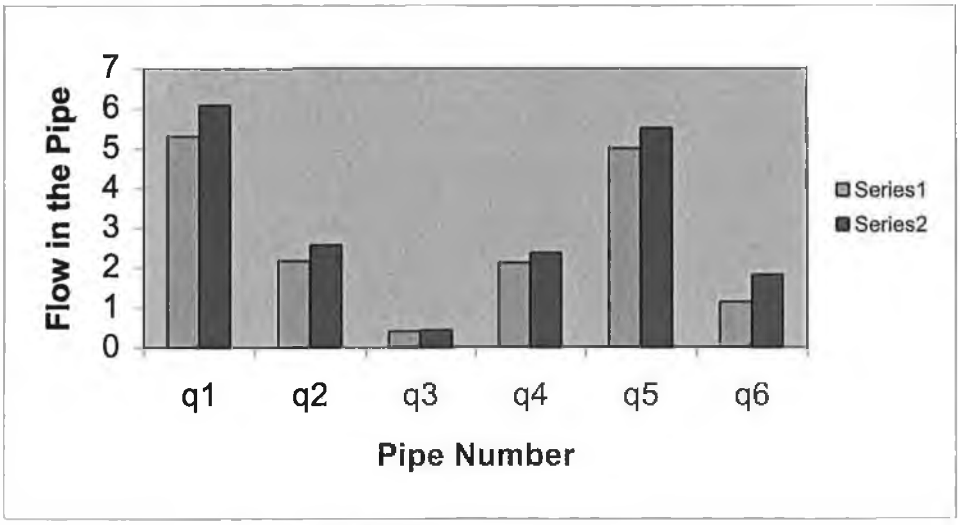


Figure 4.A16

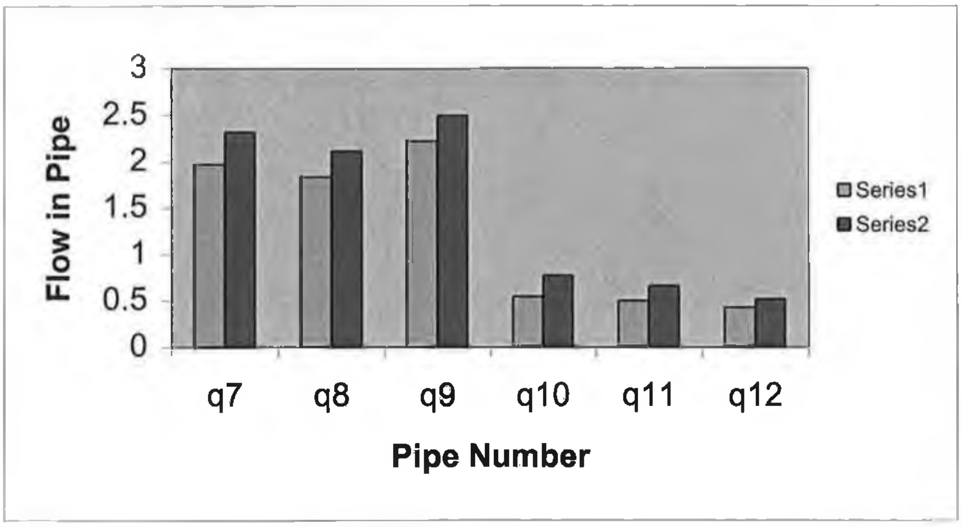


Figure 4.A17

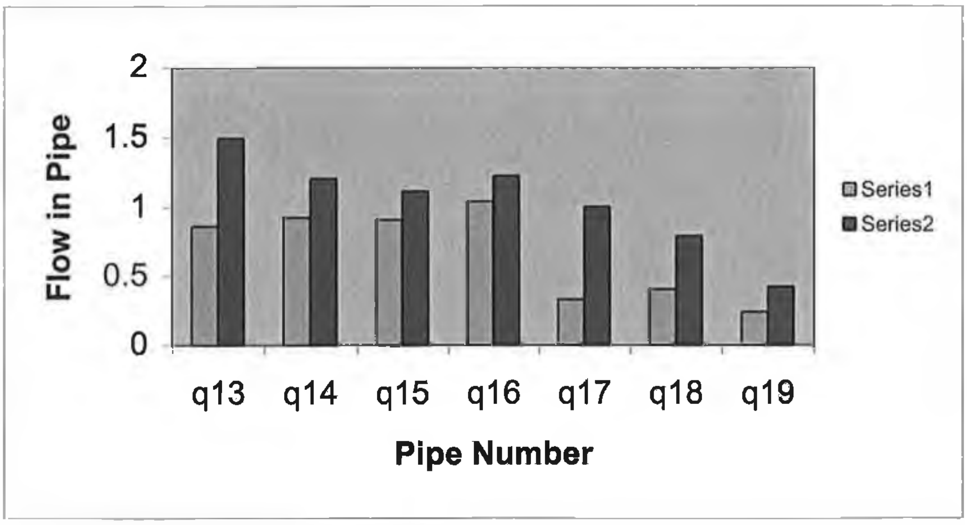


Figure 4.A18

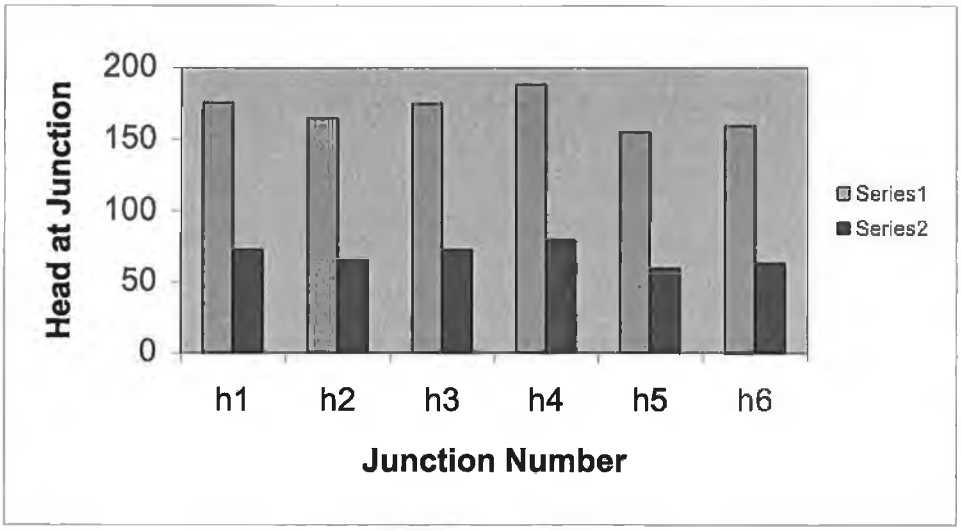


Figure 4.A19

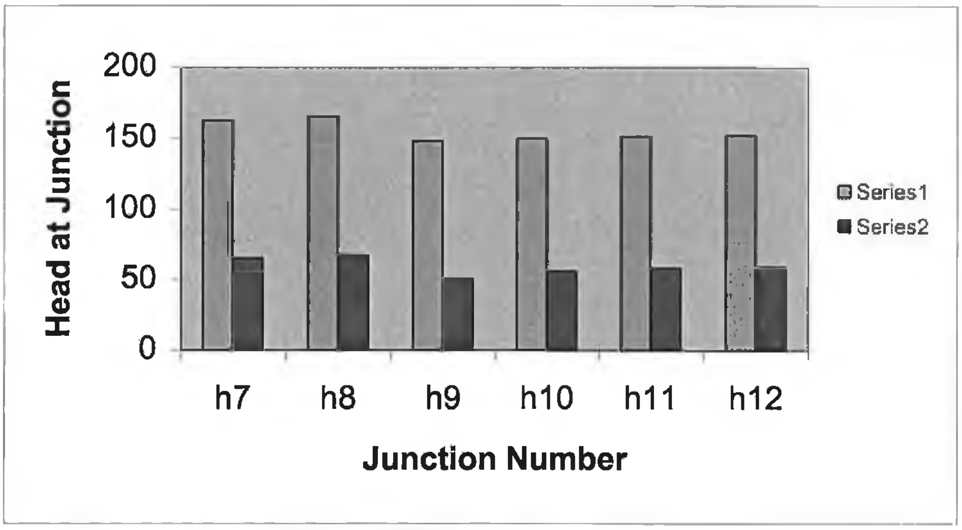


Figure 4.A20

# File Input & Output
# Example [4]

```
14 9 2 0
1 2600 14 2500
0 1 1500 1 .000166667 9.7
1 2 1000 .66667 .000166667 2.9
2 3 2000 .66667 .000166667 1.7
5 3 1000 .66667 .000166667 0.2
3 4 2000 .5 .000166667 .9
6 4 1000 .5 .000166667 .5
1 5 2000 .66667 .000166667 2.7
5 6 2000 .5 .000166667 .8
1 7 1200 .66667 .000166667 2.9
7 8 2000 .66667 .000166667 1.7
5 8 2000 .66667 .000166667 0.8
8 9 1200 .66667 .000166667 1.4
9 6 1200 .66667 .000166667 1.2
0 9 1500 .66667 .000166667 1.3

1.3 2548.6 2410
1.2 2522.8 2405
1.0 2504.1 2400
1.4 2481.1 2340
0.9 2504.3 2405
1.5 2485.4 2350
1.2 2518.1 2405
1.0 2491.6 2400
1.5 2491.6 2370
```

```
Pipe Data:
-------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                          Coef.
-------------------------------------------------------------------
   1    0    1     1500.0     1.000  0.000167     9.714    51.418
   2    1    2     1000.0     0.667  0.000167     2.887    25.810
   3    2    3     2000.0     0.667  0.000167     1.687    18.644
   4    5    3     1000.0     0.667  0.000167     0.194     0.179
   5    3    4     2000.0     0.500  0.000167     0.881    23.016
   6    6    4     1000.0     0.500  0.000167     0.519     4.307
   7    1    5     2000.0     0.667  0.000167     2.663    44.275
   8    5    6     2000.0     0.500  0.000167     0.792    18.889
   9    1    7     1200.0     0.667  0.000167     2.864    30.507
  10    7    8     2000.0     0.667  0.000167     1.664    18.173
  11    5    8     2000.0     0.667  0.000167     0.777     4.405
  12    8    9     1200.0     0.667  0.000167     1.440     8.320
  13    9    6     1200.0     0.667  0.000167     1.227     6.164
  14    0    9     1500.0     0.667  0.000167     1.286     8.496
Node Data:
--------------------------------------------------------
 Node     Demand Elevation       Head   Pressure HGL-elev.
--------------------------------------------------------


    1      1.300  2410.000    138.582     60.052  2548.582
    2      1.200  2405.000    117.772     51.034  2522.772
    3      1.000  2400.000    104.128     45.122  2504.128
    4      1.400  2340.000    141.112     61.148  2481.112
    5      0.900  2405.000     99.307     43.033  2504.307
    6      1.500  2350.000    135.418     58.681  2485.418
    7      1.200  2405.000    113.075     48.999  2518.075
    8      1.000  2400.000     99.902     43.291  2499.902
    9      1.500  2370.000    121.582     52.686  2491.582
Pipe Data:
-------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                          Coef.
-------------------------------------------------------------------
   1    0    1     1500.0     1.000  0.000167     9.712    51.400
   2    1    2     1000.0     0.667  0.000167     2.886    25.800
   3    2    3     2000.0     0.667  0.000167     1.689    18.700
   4    5    3     1000.0     0.667  0.000167     0.206     0.200
   5    3    4     2000.0     0.500  0.000167     0.880    23.000
   6    6    4     1000.0     0.500  0.000167     0.519     4.300
   7    1    5     2000.0     0.667  0.000167     2.664    44.300
   8    5    6     2000.0     0.500  0.000167     0.793    18.900
   9    1    7     1200.0     0.667  0.000167     2.863    30.500
  10    7    8     2000.0     0.667  0.000167     2.032    26.500
  11    5    8     2000.0     0.667  0.000167     1.374    12.700
  12    8    9     1200.0     0.667  0.000167     0.000     0.000
  13    9    6     1200.0     0.667  0.000167     1.231     6.200
  14    0    9     1500.0     0.667  0.000167     1.285     8.478
Node Data:
--------------------------------------------------------
 Node     Demand Elevation       Head   Pressure HGL-elev.
--------------------------------------------------------


    1      1.298  2410.000    138.600     60.060  2548.600
    2      1.197  2405.000    117.800     51.047  2522.800
    3      1.015  2400.000    104.100     45.110  2504.100
    4      1.399  2340.000    141.100     61.143  2481.100
    5      0.291  2405.000     99.300     43.030  2504.300
```

```
      6     1.505  2350.000   135.400     58.673  2485.400
      7     0.831  2405.000   113.100     49.010  2518.100
      8     3.406  2400.000    91.600     39.693  2491.600
      9     0.055  2370.000   121.600     52.693  2491.600
Devices caused the following changes in heads
Device   1 in pipe   14 Change in head =   53.417
Pipe Data:
-----------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                       Coef.
-----------------------------------------------------------------
   1    0    1    1500.0     1.000  0.000167     9.637    50.640
   2    1    2    1000.0     0.667  0.000167     2.865    25.435
   3    2    3    2000.0     0.667  0.000167     1.665    18.188
   4    5    3    1000.0     0.667  0.000167     0.206     0.200
   5    3    4    2000.0     0.500  0.000167     0.871    22.540
   6    6    4    1000.0     0.500  0.000167     0.529     4.457
   7    1    5    2000.0     0.667  0.000167     2.636    43.423
   8    5    6    2000.0     0.500  0.000167     0.779    18.283
   9    1    7    1200.0     0.667  0.000167     2.836    29.952
  10    7    8    2000.0     0.667  0.000167     1.636    17.614
  11    5    8    2000.0     0.667  0.000167     0.751     4.142
  12    8    9    1200.0     0.667  0.000167     1.387     7.756
  13    9    6    1200.0     0.667  0.000167     1.250     6.386
  14    0    9    1500.0     0.667  0.000167     1.363     9.465
Node Data:
-------------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
-------------------------------------------------------

      1     1.300  2410.000   139.360     60.389  2549.360
      2     1.200  2405.000   118.925     51.534  2523.925
      3     1.000  2400.000   105.737     45.819  2505.737
      4     1.400  2340.000   143.197     62.052  2483.197
      5     0.900  2405.000   100.937     43.739  2505.937
      6     1.500  2350.000   137.654     59.650  2487.654
      7     1.200  2405.000   114.409     49.577  2519.409
      8     1.000  2400.000   101.795     44.111  2501.795
      9     1.500  2370.000   124.040     53.751  2494.040
Devices caused the following changes in heads
Device   1 in pipe   14 Change in head =   52.769
Pipe Data:
-----------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                       Coef.
-----------------------------------------------------------------
   1    0    1    1500.0     1.000  0.000167     9.712    51.400
   2    1    2    1000.0     0.667  0.000167     2.886    25.800
   3    2    3    2000.0     0.667  0.000167     1.689    18.700
   4    5    3    1000.0     0.667  0.000167     0.206     0.200
   5    3    4    2000.0     0.500  0.000167     0.880    23.000
   6    6    4    1000.0     0.500  0.000167     0.519     4.300
   7    1    5    2000.0     0.667  0.000167     2.664    44.300
   8    5    6    2000.0     0.500  0.000167     0.793    18.900
   9    1    7    1200.0     0.667  0.000167     2.863    30.500
  10    7    8    2000.0     0.667  0.000167     2.032    26.500
  11    5    8    2000.0     0.667  0.000167     1.374    12.700
  12    8    9    1200.0     0.667  0.000167     0.000     0.000
  13    9    6    1200.0     0.667  0.000167     1.231     6.200
  14    0    9    1500.0     0.667  0.000167     1.496    11.273
Node Data:
-------------------------------------------------------
```

```
     Node    Demand Elevation      Head  Pressure HGL-elev.
    -------------------------------------------------------

        1     1.298  2410.000   138.600    60.060  2548.600
        2     1.197  2405.000   117.800    51.047  2522.800
        3     1.015  2400.000   104.100    45.110  2504.100
        4     1.399  2340.000   141.100    61.143  2481.100
        5     0.291  2405.000    99.300    43.030  2504.300
        6     1.505  2350.000   135.400    58.673  2485.400
        7     0.831  2405.000   113.100    49.010  2518.100
        8     3.406  2400.000    91.600    39.693  2491.600
        9     0.266  2370.000   121.600    52.693  2491.600
```

```
14 9 2 0
1 2550 14 2500
0 1 1500 1 .000166667 9.7
1 2 1000 .66667 .000166667 2.9
2 3 2000 .66667 .000166667 1.7
5 3 1000 .66667 .000166667 0.2
3 4 2000 .5 .000166667 .9
6 4 1000 .5 .000166667 .5
1 5 2000 .66667 .000166667 2.7
5 6 2000 .5 .000166667 .8
1 7 1200 .66667 .000166667 2.9
7 8 2000 .66667 .000166667 1.7
5 8 2000 .66667 .000166667 0.8
8 9 1200 .66667 .000166667 1.4
9 6 1200 .66667 .000166667 1.2
0 9 1500 .66667 .000166667 1.3

1.3 2548.6 2410
1.2 2522.8 2405
1.0 2504.1 2400
1.4 2481.1 2340
0.9 2504.3 2405
1.5 2485.4 2350
1.2 2518.1 2405
1.0 2491.6 2400
1.5 2491.6 2370
```

```
Pipe Data:
------------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
 No.   #1   #2                       Coef.
------------------------------------------------------------------
   1    0    1    1500.0     1.000  0.000167    10.891    64.144
   2    1    2    1000.0     0.667  0.000167     3.230    32.002
   3    2    3    2000.0     0.667  0.000167     2.030    26.448
   4    5    3    1000.0     0.667  0.000167     0.027     0.006
   5    3    4    2000.0     0.500  0.000167     1.057    32.440
   6    6    4    1000.0     0.500  0.000167     0.343     2.010
   7    1    5    2000.0     0.667  0.000167     3.080    58.444
   8    5    6    2000.0     0.500  0.000167     1.022    30.435
   9    1    7    1200.0     0.667  0.000167     3.281    39.567
  10    7    8    2000.0     0.667  0.000167     2.081    27.713
  11    5    8    2000.0     0.667  0.000167     1.131     8.836
  12    8    9    1200.0     0.667  0.000167     2.212    18.672
  13    9    6    1200.0     0.667  0.000167     0.821     2.927
  14    0    9    1500.0     0.667  0.000167     0.109     0.097
Node Data:
--------------------------------------------------------
 Node    Demand Elevation      Head  Pressure HGL-elev.
--------------------------------------------------------

    1     1.300  2410.000   175.856    76.204  2585.856
    2     1.200  2405.000   148.854    64.503  2553.854
    3     1.000  2400.000   127.406    55.209  2527.406
    4     1.400  2340.000   154.966    67.152  2494.966
    5     0.900  2405.000   122.412    53.045  2527.412
    6     1.500  2350.000   146.976    63.690  2496.976
    7     1.200  2405.000   141.289    61.225  2546.289
    8     1.000  2400.000   118.576    51.383  2518.576
    9     1.500  2370.000   129.904    56.292  2499.904
Pipe Data:
------------------------------------------------------------------
Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss
 No.   #1   #2                       Coef.
------------------------------------------------------------------
   1    0    1    1500.0     1.000  0.000167    13.791   101.400
   2    1    2    1000.0     0.667  0.000167     2.886    25.800
   3    2    3    2000.0     0.667  0.000167     1.689    18.700
   4    5    3    1000.0     0.667  0.000167     0.206     0.200
   5    3    4    2000.0     0.500  0.000167     0.880    23.000
   6    6    4    1000.0     0.500  0.000167     0.519     4.300
   7    1    5    2000.0     0.667  0.000167     2.664    44.300
   8    5    6    2000.0     0.500  0.000167     0.793    18.900
   9    1    7    1200.0     0.667  0.000167     2.863    30.500
  10    7    8    2000.0     0.667  0.000167     2.032    26.500
  11    5    8    2000.0     0.667  0.000167     1.374    12.700
  12    8    9    1200.0     0.667  0.000167     0.000     0.000
  13    9    6    1200.0     0.667  0.000167     1.231     6.200
  14    0    9    1500.0     0.667  0.000167     1.285     8.478
Node Data:
--------------------------------------------------------
 Node    Demand Elevation      Head  Pressure HGL-elev.
--------------------------------------------------------

    1     5.378  2410.000   138.600    60.060  2548.600
    2     1.197  2405.000   117.800    51.047  2522.800
    3     1.015  2400.000   104.100    45.110  2504.100
    4     1.399  2340.000   141.100    61.143  2481.100
    5     0.291  2405.000    99.300    43.030  2504.300
```

```
        6      1.505  2350.000   135.400     58.673  2485.400
        7      0.831  2405.000   113.100     49.010  2518.100
        8      3.406  2400.000    91.600     39.693  2491.600
        9      0.055  2370.000   121.600     52.693  2491.600
Pipe Data:
--------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                        Coef.
--------------------------------------------------------------------
   1    0    1     1500.0     1.000  0.000167     9.714    51.418
   2    1    2     1000.0     0.667  0.000167     2.887    25.810
   3    2    3     2000.0     0.667  0.000167     1.687    18.644
   4    5    3     1000.0     0.667  0.000167     0.194     0.179
   5    3    4     2000.0     0.500  0.000167     0.881    23.016
   6    6    4     1000.0     0.500  0.000167     0.519     4.307
   7    1    5     2000.0     0.667  0.000167     2.663    44.275
   8    5    6     2000.0     0.500  0.000167     0.792    18.889
   9    1    7     1200.0     0.667  0.000167     2.864    30.507
  10    7    8     2000.0     0.667  0.000167     1.664    18.173
  11    5    8     2000.0     0.667  0.000167     0.777     4.405
  12    8    9     1200.0     0.667  0.000167     1.440     8.320
  13    9    6     1200.0     0.667  0.000167     1.227     6.164
  14    0    9     1500.0     0.667  0.000167     1.286     8.496
Node Data:
----------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
----------------------------------------------------


        1      1.300  2410.000   138.582     60.052  2548.582
        2      1.200  2405.000   117.772     51.034  2522.772
        3      1.000  2400.000   104.128     45.122  2504.128
        4      1.400  2340.000   141.112     61.148  2481.112
        5      0.900  2405.000    99.307     43.033  2504.307
        6      1.500  2350.000   135.418     58.681  2485.418
        7      1.200  2405.000   113.075     48.999  2518.075
        8      1.000  2400.000    99.902     43.291  2499.902
        9      1.500  2370.000   121.582     52.686  2491.582
Pipe Data:
--------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                        Coef.
--------------------------------------------------------------------
   1    0    1     1500.0     1.000  0.000167     9.712    51.400
   2    1    2     1000.0     0.667  0.000167     2.886    25.800
   3    2    3     2000.0     0.667  0.000167     1.689    18.700
   4    5    3     1000.0     0.667  0.000167     0.206     0.200
   5    3    4     2000.0     0.500  0.000167     0.880    23.000
   6    6    4     1000.0     0.500  0.000167     0.519     4.300
   7    1    5     2000.0     0.667  0.000167     2.664    44.300
   8    5    6     2000.0     0.500  0.000167     0.793    18.900
   9    1    7     1200.0     0.667  0.000167     2.863    30.500
  10    7    8     2000.0     0.667  0.000167     2.032    26.500
  11    5    8     2000.0     0.667  0.000167     1.374    12.700
  12    8    9     1200.0     0.667  0.000167     0.000     0.000
  13    9    6     1200.0     0.667  0.000167     1.231     6.200
  14    0    9     1500.0     0.667  0.000167     1.285     8.478
Node Data:
----------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
----------------------------------------------------


        1      1.298  2410.000   138.600     60.060  2548.600
```

```
      2     1.197  2405.000   117.800     51.047  2522.800
      3     1.015  2400.000   104.100     45.110  2504.100
      4     1.399  2340.000   141.100     61.143  2481.100
      5     0.291  2405.000    99.300     43.030  2504.300
      6     1.505  2350.000   135.400     58.673  2485.400
      7     0.831  2405.000   113.100     49.010  2518.100
      8     3.406  2400.000    91.600     39.693  2491.600
      9     0.055  2370.000   121.600     52.693  2491.600
Pipe Data:
------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                        Coef.
------------------------------------------------------------------
   1    0    1     1500.0     1.000  0.000167     9.437    48.634
   2    1    2     1000.0     0.667  0.000167     2.807    24.471
   3    2    3     2000.0     0.667  0.000167     1.607    17.027
   4    5    3     1000.0     0.667  0.000167     0.239     0.261
   5    3    4     2000.0     0.500  0.000167     0.847    21.371
   6    6    4     1000.0     0.500  0.000167     0.553     4.845
   7    1    5     2000.0     0.667  0.000167     2.566    41.238
   8    5    6     2000.0     0.500  0.000167     0.744    16.787
   9    1    7     1200.0     0.667  0.000167     2.764    28.521
  10    7    8     2000.0     0.667  0.000167     1.564    16.189
  11    5    8     2000.0     0.667  0.000167     0.682     3.472
  12    8    9     1200.0     0.667  0.000167     1.247     6.353
  13    9    6     1200.0     0.667  0.000167     1.309     6.962
  14    0    9     1500.0     0.667  0.000167     1.563    12.230
Node Data:
-------------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
-------------------------------------------------------

      1     1.300  2410.000   128.946     55.877  2538.946
      2     1.200  2405.000   109.475     47.439  2514.475
      3     1.000  2400.000    97.448     42.227  2497.448
      4     1.400  2340.000   136.077     58.967  2476.077
      5     0.900  2405.000    92.709     40.174  2497.709
      6     1.500  2350.000   130.922     56.733  2480.922
      7     1.200  2405.000   105.426     45.684  2510.426
      8     1.000  2400.000    94.237     40.836  2494.237
      9     1.500  2370.000   117.884     51.083  2487.884
Pipe Data:
------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                        Coef.
------------------------------------------------------------------
   1    0    1     1500.0     1.000  0.000167     8.413    38.980
   2    1    2     1000.0     0.667  0.000167     2.886    25.800
   3    2    3     2000.0     0.667  0.000167     1.689    18.700
   4    5    3     1000.0     0.667  0.000167     0.206     0.200
   5    3    4     2000.0     0.500  0.000167     0.880    23.000
   6    6    4     1000.0     0.500  0.000167     0.519     4.300
   7    1    5     2000.0     0.667  0.000167     2.664    44.300
   8    5    6     2000.0     0.500  0.000167     0.793    18.900
   9    1    7     1200.0     0.667  0.000167     2.863    30.500
  10    7    8     2000.0     0.667  0.000167     2.032    26.500
  11    5    8     2000.0     0.667  0.000167     1.374    12.700
  12    8    9     1200.0     0.667  0.000167     0.000     0.000
  13    9    6     1200.0     0.667  0.000167     1.231     6.200
  14    0    9     1500.0     0.667  0.000167     1.285     8.478
Node Data:
-------------------------------------------------------
```

```
 Node     Demand Elevation      Head  Pressure HGL-elev.
-----------------------------------------------------------


    1      0.000  2410.000   138.600    60.060  2548.600
    2      1.197  2405.000   117.800    51.047  2522.800
    3      1.015  2400.000   104.100    45.110  2504.100
    4      1.399  2340.000   141.100    61.143  2481.100
    5      0.291  2405.000    99.300    43.030  2504.300
    6      1.505  2350.000   135.400    58.673  2485.400
    7      0.831  2405.000   113.100    49.010  2518.100
    8      3.406  2400.000    91.600    39.693  2491.600
    9      0.055  2370.000   121.600    52.693  2491.600
Pipe Data:
-------------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                         Coef.
-------------------------------------------------------------------------
   1    0    1     1500.0     1.000  0.000167     8.624    40.886
   2    1    2     1000.0     0.667  0.000167     2.576    20.777
   3    2    3     2000.0     0.667  0.000167     1.376    12.730
   4    5    3     1000.0     0.667  0.000167     0.389     0.624
   5    3    4     2000.0     0.500  0.000167     0.765    17.687
   6    6    4     1000.0     0.500  0.000167     0.635     6.249
   7    1    5     2000.0     0.667  0.000167     2.278    32.883
   8    5    6     2000.0     0.500  0.000167     0.623    12.061
   9    1    7     1200.0     0.667  0.000167     2.471    23.029
  10    7    8     2000.0     0.667  0.000167     1.271    10.970
  11    5    8     2000.0     0.667  0.000167     0.366     1.116
  12    8    9     1200.0     0.667  0.000167     0.636     1.834
  13    9    6     1200.0     0.667  0.000167     1.512     9.112
  14    0    9     1500.0     0.667  0.000167     2.376    26.972
Node Data:
-----------------------------------------------------------
 Node     Demand Elevation      Head  Pressure HGL-elev.
-----------------------------------------------------------


    1      1.300  2410.000    99.115    42.950  2509.115
    2      1.200  2405.000    83.337    36.113  2488.337
    3      1.000  2400.000    75.607    32.763  2475.607
    4      1.400  2340.000   117.921    51.099  2457.921
    5      0.900  2405.000    71.231    30.867  2476.231
    6      1.500  2350.000   114.170    49.474  2464.170
    7      1.200  2405.000    81.086    35.137  2486.086
    8      1.000  2400.000    75.115    32.550  2475.115
    9      1.500  2370.000   103.282    44.756  2473.282
Pipe Data:
-------------------------------------------------------------------------
Pipe Node Node     Length  Diameter Roughness  Flowrate  Headloss
  No.  #1   #2                         Coef.
-------------------------------------------------------------------------
   1    0    1     1500.0     1.000  0.000167     1.433     1.400
   2    1    2     1000.0     0.667  0.000167     2.886    25.800
   3    2    3     2000.0     0.667  0.000167     1.689    18.700
   4    5    3     1000.0     0.667  0.000167     0.206     0.200
   5    3    4     2000.0     0.500  0.000167     0.880    23.000
   6    6    4     1000.0     0.500  0.000167     0.519     4.300
   7    1    5     2000.0     0.667  0.000167     2.664    44.300
   8    5    6     2000.0     0.500  0.000167     0.793    18.900
   9    1    7     1200.0     0.667  0.000167     2.863    30.500
  10    7    8     2000.0     0.667  0.000167     2.032    26.500
  11    5    8     2000.0     0.667  0.000167     1.374    12.700
  12    8    9     1200.0     0.667  0.000167     0.000     0.000
```

```
      13     9     6     1200.0       0.667  0.000167       1.231      6.200
      14     0     9     1500.0       0.667  0.000167       1.285      8.478
    Node Data:
    ------------------------------------------------------
     Node     Demand Elevation       Head  Pressure HGL-elev.
    ------------------------------------------------------

        1     -6.981  2410.000    138.600    60.060  2548.600
        2      1.197  2405.000    117.800    51.047  2522.800
        3      1.015  2400.000    104.100    45.110  2504.100
        4      1.399  2340.000    141.100    61.143  2481.100
        5      0.291  2405.000     99.300    43.030  2504.300
        6      1.505  2350.000    135.400    58.673  2485.400
        7      0.831  2405.000    113.100    49.010  2518.100
        8      3.406  2400.000     91.600    39.693  2491.600
        9      0.055  2370.000    121.600    52.693  2491.600
```

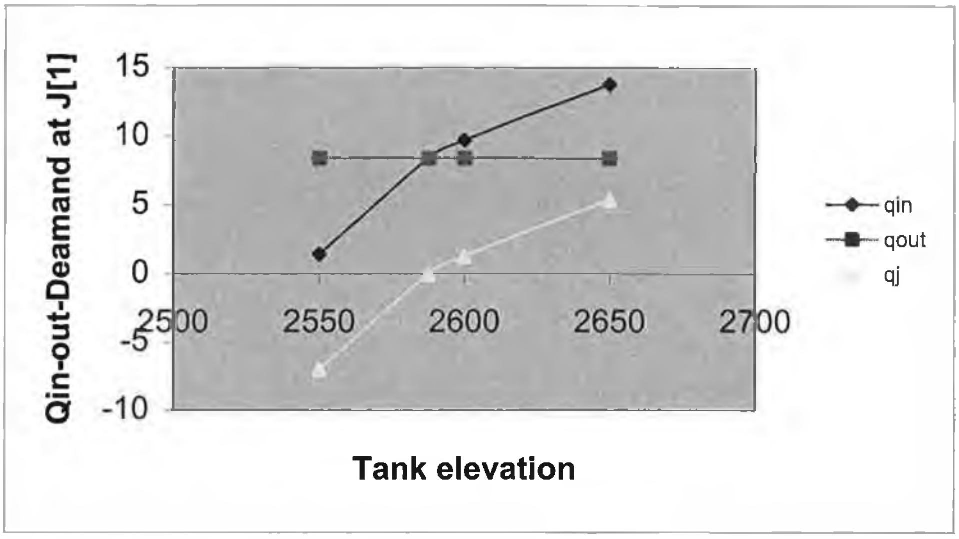


Figure 4.A21

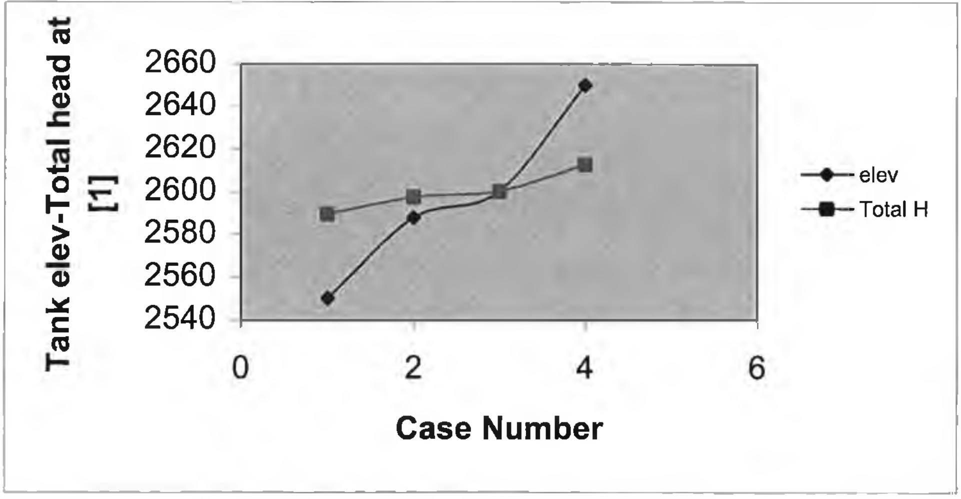


Figure 4.A22

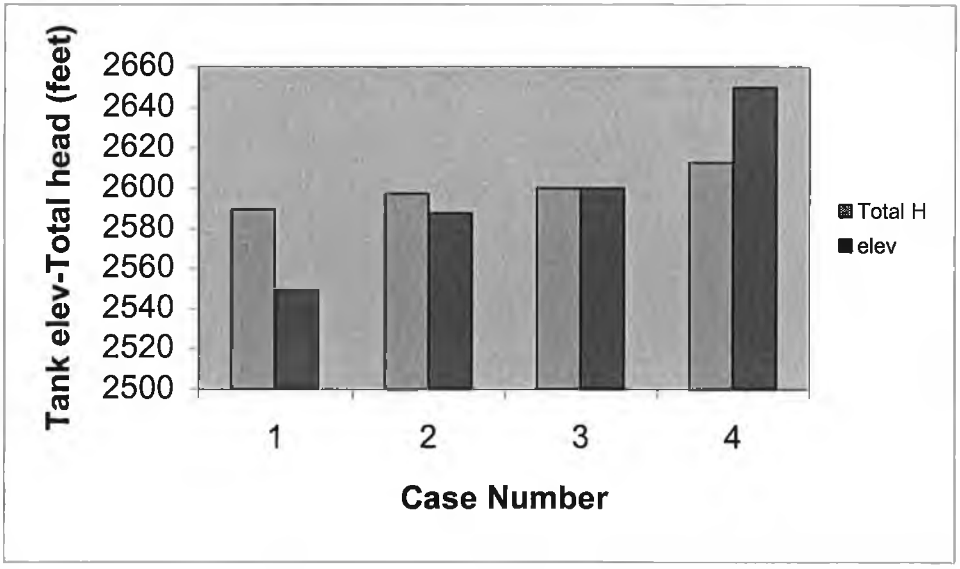


Figure 4.A23

# Chapter five

## Conclusions

The goal of this thesis was to develop a computer programme for analysing steady sate flow problem in pipe networks. The programme has been developed perform the following task.

1. Read the input data that defines the networks
2. Develop from this information the system of equations and the junction continuity equations and the energy equations of the networks. This task defies the equations and also forms each element of the matrix.
3. Solve the system of equations. Using standard linear algebra solver. However program for large network problems should have a special linear algebra solver that takes advantage of the special properties of a sparse Jacobian matrix.
4. Obtain the head at each node after the pipe discharges have been found.
5. Write the solution results in tables that can be readily understood. (Pipe data and node data).

The use of programs for network analysis can also allow designers to obtain answer for many questions that naturally occur during the design process. For example, what head and capacity should a pump produce to maintain a prescribed pressure and or discharge at the far end of the network? What will be the discharge from the junction if the pressure is known from measurement there. The program developed has been successfully tested by solving a number of examples on network analysis for steady flow liquid

## Suggestions for Further Work

The program developed for solving steady state problem can be used as initial condition for the unsteady state analysis problem. Steady state analysis for a network must be available before a transient (water hammer) analysis of the networks can be conducted. Herein steady sate solution can be obtained form this work, since it can be produce a file with information that is needed by the transient network analysis.

Extended time simulation is important to the design of looped networks. The systems do not operate under steady state condition, extended time simulation is needed to simulate the performance as they respond to demands which vary with time, and which have pumps switched on or off, depending in those demands. The tank characteristics must be specified. These include the tank diameter, maximum surface elevation and minimum surface. If the tank if full no additional flow from the pipe system can enter the tank, and if it is empty no additional flow can leave the tank. For further study a message from a program can be written to indicate the limit of the thank surface. Extended time simulation consists a series of steady-state solution based on changing demands and reservoir water level, the number of operating pumps etc. This type of time dependent solution is obtained by solving a system of simultaneous non-linear algebraic equations. Taken full advantage of the Jacobean matrix in computing, the networks can be solved in efficient manner. The size of the matrix can be reduced by taking out all the zero elements of the matrix so saving computer time and reducing the amount of memory to solve system of equations on a personal computer.

# REFERNCE

**1-Roach, P.J.**, Computational of fluid dynamics, Hermosa Publisher, Albuquerque, 1976.

**2-Roland W. Jepson**, "Analysis of flow in pipe networks, Ann Arbor Science 161pp 1976"

**3-Philip M Gerhart. Richard Gross. John I Hochstein**. "Fundamental of fluid mechanics second edition"ISBN 0-201-18358-7

**4-Cross, H.,** "analysis of flow in networks of conduits or Conductors" Bulletin *No*.286, university of Illinois Eng. Expr station, urban, ill, 1936

**5-Chenoweth, H., and Crawford C,** "pipe network analysis," journal, American waterworks association, jan.1974.

**6-Dillinaham, J, H.,** "computer analysis of water distribution systems,"parts1-5, water and sewage works, Jan-may, 1967.

**7-Fietz, T.R** "steady flow in small pipe networks by the simple loop method "waters research laboratory report no

**8-Jeppson R W** "analysis of flow in pipe networks," Ann Arbor Science Ann Arbor Mich 1977.

**9-Robinson, P.M and Rossum J R** "program documentation for hydraulic networks analysis computer programs," California water service company, Jun 1975.

**10-Martin, D W and Peters G** "the application of Newton's Method to netwprk analysis by Digital Computer,"Journal of the Institute of Water Engineers,, vol.17, 1963,pp115-129.

**11-Epp, R., and Fowler, A.G** ''efficient code for steady-state flows in networks, ''journal of the hydraulics division ASCE, vol.96, noHY1 Pro paper 7002.jan.1970, pp.43-56.

**12-Heafley A.H. And Lawson, J.D.**,'' analysis of water distribution networks, ''proceeding of the fifth Manitoba conference on numerical mathematics, university of Manitoba, Oct., 1975.

**13-Wood, D and C Charles**,'' Hydraulic Network Analysis Using Linear Theory,'' Journal Hydraulic Division, ASCE, Vol. 98,No HY7, PP 1157-1170,July 1972.

**14-Fietz, T.R**. ''Steady flow in small pipe networks using linear theory,'' water research laboratory report no.130, university of new south Wales, Sydney Australia, Oct. 1972.

**15-Wood, D.J,** ''user manual-a computer program for the analysis of pressure and flow in pipe distribution system,'' office of engineering continuing education, university of Kentucky, Lexington, K. y, 1974

**16-Shamir, U., and Howard,C.D.D,** ''water distribution system analysis,'' journal of the hydraulics division,ASCE,vol.94,no.HYI,proc.paper 5758,jan.,1968,pp.219-234.

**17-Barlow, J, F., and Markland,e**. ,''computer analysis of pipe netwroks,2 proceedings i.c.e./vol.43.june,1969,pp.249-259.

**18-Lemieux, P.F**., ''efficient algorithms for distribution networks'' journal of the hydraulics division, ASCE, vol. 98,no. HY11, proc.paper 9336,nov.,1972,pp.1911-1920

**19-Collins M.A.,Cooper L.,Helgason R. ,Kennington J.and Le Blanc L** :(1978),''solving the pipe networks analysis problem optimisation techniques'', Management Science,24,7,pp. 747-760.

**20-Contro R and Franzetti S**. (1982), a new objective function for analysing hydraulic pipe netwroks in the pressure of different sets of flow, dipartimento di Ingegneria strutturale a istituto di Idraulica del politecnico di Milan o." computer application in water supply V1.IEBN 0-86380-062-9 P-19.

**21-E TODINI and S PILATI (87),** "A gradient algorithm for the solution of looped pipe networks"computer application in water supply v1 ISBN 0-86380-062-9 pp 1-20.

**22-Hoga, L.N., and W., G.,"** pipeline networks analysis by electronic Digital Computer" journal of the American water association Vol. 49 pp, 571-524.

**23-Coolins,M. A.,and Kennington,J.L.**, discussion of "extended period siumlation of water system- part B,"by H.S. Rao,L.C.Markel,and D.W.Brue.Journal of hydraluics Division ,ASCE,Vol 103,No HY12,proc paper 13378.Dec.,1977,pp.1496-1500.

**24-Wood.D.J. (1981a)**,"Algorithms for pipe networks analysis and their reliability", research Report no.127, water resource research institute, University of Kentucky.

**25-Wood D J and Rayes A G (1981)**,"reliability of algorithms for pipe network analysis", journal hydraulics division, ASCE, Vol. 107 no. HY10, pp.1145-1161.

**26-Wood.D.J,(1981),** Discussion of Isaacs and Mills(1980): 2 linear theory method for pipe networks analysis" Journal of Hydraulic division ,ASCE, March pp,384-385

**27-R. Salgado, E Todini and P E O'Connell-1987-PP38-62."**Comparison of the gradient method with some traditional methods for the analysis of water supply distribution networks.,Int.Conf. on Computer Application for Water Supply and Distribution , Leicester, U.K.

**28-E. TODINI and S Pilati (1987)** a gradient algorithm for the analysis of pipe networks " Proceeding of International/Conference on computer application in water supply and distribution, Leicester ,UK

**29-Templeman, B.**," Discussion of 'Optimisation of looped water Distribution systems' by Qunidry et al.," Journal of the Environmental Engineering Division, ASCE, Vol. 108,No 3,pp.599-602, 1982.

**30- Kevin E Lansey and Larry W Mays**, "Reliability analysis of water network distribution network" simulation models 1989 Chapter 2 P-11-36

**31-Wood D.J and Rays,A.G.(1981)-"**Reliability of algorithm for pipe network analysis."J.Hydr.Div.,ASCE,107(10),1145-1161.

**32-Paul F. Boulos and Don J. Wood**," Explicit Calculation of Pipe Network Parameters, Journal of Hydraulic Engineering Vol.116, No 11,Nov 1990,pp1329-1344).

**33-Giles, R.V., Evett, J.B., Liu, C., 1994**. Theory and Properties of Fluid Mechanics and Hydraulics, 3rd ed. McGraw-Hill, New York, pp. 362.

**34-Dake, J.M., 1972.** Essentials of Engineering Hydraulics. Wiley, New York, pp. 418.

**35- Jepson, R** " Analysis of Flow in pipe networks ". Ann Arbor Science, Ann Arbor, Michigan, 1972

**36-Granet, I., 1996.** Fluid Mechanics, 4th ed. Prentice Hall, Englewood Cliffs, NJ, pp. 460.

**37-Streeter, V.L., Wylie, E.B., 1985.** Fluid Mechanics, 8th Ed. McGraw Hill, New York, pp. 586.

**38-Murdock, J.W., 1976**. Fluid Mechanics and Its Applications. Houghton Mifflin, Boston, pp. 422.

**39-Miller, D.S.1978.** Internal Flow System. Bedford England: The British Hydrodynamic Reasearch Association, Fluid Engineering Granfiled.

**40-Mohtar, R.H., Bralts, V.F. and Shayya, W.H., 1990**. A finite element model for the analysis and optimisation of pipe networks.Trans. Am. Soc. Agric. Eng. 34 2, pp. 393-401

**41-Gerrish, P.J., Shayya, W.H. and Bralts, V.F., 1996.** Improved methods for incorporating pipe components into the analysis of hydraulic networks. Trans. Am. Soc. Agric. Eng. 39 4, pp. 1337¯1343

**42-Bralts, V.F., Kelly, S.F., Shayya, W.H. and Segerlind, L.J., 1993**. Finite element analysis of microirrigation hydraulics uses a virtual emitter system. Trans. Am. Soc. Agric. Eng. 36 3, pp. 717-725

**43-Gerrish, P.J., Bralts, V.F. and Shayya, W.H., 1996.** An improved analysis of microirrigation hydraulics uses a virtual emitter system. Trans. Am. Soc. Agric. Eng. 39 4, pp. 1403-1410

**44-Brater, E.F., King, H.W., Lindell, J.E., Wei, C.Y., 1996**. Handbook of Hydraulics, 8th ed. McGraw-Hill, New York, pp 640.

**45-Bralts, V.F. and Segerlind, L.J., 1985**. Finite element analysis of drip irrigation sub-main units. Trans. Am. Soc. Agric. Eng. 28 3, pp. 809-814

**46-Haghighi, K., Bralts, V.F. and Segerlind, L.J., 1988.** Finite element formulation of tee and bend components in hydraulic pipe network analysis. Trans. Am. Soc. Agric. Eng. 31 6, pp. 1750-1758

**47-Kang, Y. and Nishiyama, S., 1995**. Hydraulic analysis of micro-irrigation sub-main units. Trans. Am. Soc. Agric. Eng. 38 5, pp.1377¯1384 Abstract

**48- Haghighi,K.,V.F.Bralts,R.Mohtar and L.J.Segerlind.(1989)**-Modelling expansion/contraction, valve and booster pump in hydraulic pipe network analysis :A finite element approach .Transactions of the ASAE 32(6):1945-1953.

**49-kevin E. Lansey and Larry W.Mays** ”network simulation model” chapter-2 pp24-25

**50- Tavallaee, A.**, ”Inclusion of pressure reducing valve and reservoir in pipe networks solved by linear theory method, ”MS Thesis, Department of Civil and Environmental Engineering, Utah State University, 1974.

**51-J. Krop and D. Goricanec** “ Analysis of networks including pumps energy and building Journal of Hydraulic Engineering Vol.112 Sep1991, pp 141-145

**52-Wood, D, J,** ”User Manual – Computer Analysis of Flow in pipe networks including Extended Period Simulation,” College of Engineering University of Kentucky, Lexington, KY 1980.

# APPENDIX A

Table A.1 Physical properties of ordinary water and common liquids (SI units)

| Liquid | Temperature $T(^oC)$ | Density $\rho(\frac{kg}{m^3})$ | Specific gravity S | Absolute viscosity $\mu(\frac{N.s}{m^2})$ | Kinematics viscosity $\nu(\frac{m^2}{s})$ |
|---|---|---|---|---|---|
| Water | 0 | 1000 | 1.000 | 1.97E-6 | 1.79E-6 |
| | 10 | 1000 | 1.000 | 1.31E-3 | 1.31E-6 |
| | 20 | 998 | 0.998 | 1.00E-3 | 1.00E-6 |
| | 30 | 996 | 0.996 | 7.98E-4 | 8.01E-7 |
| | 40 | 992 | 0.992 | 6.53E-4 | 6.58E-7 |
| | 50 | 988 | 0.988 | 5.47E-4 | 5.48E-7 |
| | 60 | 983 | 0.983 | 4.67E-4 | 4.75E-7 |
| | 70 | 978 | 0.978 | 4.04E-4 | 4.13E-7 |
| | 80 | 972 | 0.972 | 3.55E-4 | 3.65E-7 |
| | 90 | 965 | 0.965 | 3.15E-4 | 3.26E-7 |
| | 100 | 958 | 0.958 | 2.82E-4 | 2.94E-7 |
| Mercury | 0 | 13600 | 13.60 | 1.68E-3 | 1.24E-7 |
| | 4 | 13590 | 13.59 | ---------- | ---------- |
| | 20 | 13550 | 13.55 | 1.55E-3 | 1.14E-7 |
| | 40 | 13500 | 13.50 | 1.45E-3 | 1.07E-3 |
| | 60 | 13450 | 13.45 | 1.37E-3 | 1.02E-7 |
| | 80 | 13400 | 13.40 | 1.30E-3 | 9.70E-8 |
| | 100 | 13350 | 13.35 | 1.24E-3 | 9.29E-8 |
| Ethylene Glycol | 0 | --------- | --------- | 5.70E-2 | -------- |
| | 20 | 1110 | 1.11 | 1.99E-2 | 1.79E-5 |
| | 40 | 1110 | 1.10 | 9.13E-3 | 8.30E-6 |
| | 60 | 1090 | 1.09 | 4.95E-3 | 4.54E-6 |
| | 80 | 1070 | 1.07 | 3.02E-3 | 2.82E-6 |
| | 100 | 1060 | 1.06 | 1.99E-3 | 1.88E-6 |
| Methyl Alcohol (methanol) | 0 | 810 | 0.810 | 8.17E-4 | 1.01E-6 |
| | 10 | 801 | 0.801 | --------- | -------- |
| | 20 | 792 | 0.792 | 5.84E-4 | 7.37E-7 |
| | 30 | 783 | 0.783 | 5.10E-4 | 6.51E-7 |
| | 40 | 774 | 0.774 | 4.51E-4 | 5.81E-7 |
| | 50 | 765 | 0.765 | 3.96E-4 | 5.18E-7 |
| Ethyl alcohol (ethanol) | 0 | 806 | 0.806 | 1.77E-3 | 2.20E-6 |
| | 20 | 789 | 0.789 | 1.20E-3 | 1.52E-6 |
| | 40 | 772 | 0.772 | 8.34E-4 | 1.08E-6 |
| | 60 | 754 | 0.754 | 5.92E-4 | 7.85E-7 |
| Normal octane | 0 | 718 | 0.718 | 7.06E-4 | 9.83E-7 |
| | 16 | -------- | ------ | 5.74E-4 | -------- |
| | 20 | 702 | 0.702 | 5.42E-4 | 7.72E-7 |
| | 25 | ----- | ------- | ------- | --------- |
| | 40 | 686 | 0.686 | 4.33E-4 | 6.31E-7 |
| Benzene | 0 | 900 | 0.900 | 9.12E-4 | 1.01E-6 |
| | 20 | 879 | 0.879 | 6.52E-4 | 7.42E-7 |
| | 40 | 858 | 0.857 | 5.03E-4 | 5.86E-7 |
| | 60 | 836 | 0.836 | 3.92E-4 | 4.69E-7 |
| | 80 | 815 | 0.815 | 3.29E-4 | 4.04E-6 |
| Kerosene | -18 | 841 | 0.841 | 7.06E-3 | 8.40E-6 |
| | 20 | 814 | 0.814 | 1.90E-3 | 2.37E-6 |
| Lubricating oil | 20 | 871 | 0.871 | 1.31E-2 | 1.50E-5 |
| | 40 | 858 | 0.858 | 6.81E-3 | 7.94E-6 |
| | 60 | 845 | 0.845 | 4.18E-3 | 4.95E-6 |
| | 80 | 832 | 0.832 | 2.83E-3 | 3.40E-6 |
| | 100 | 820 | 0.820 | 2.00E-3 | 2.44E-6 |
| | 120 | 809 | 0.809 | 1.54E-3 | 1.90E-6 |

Table A.2 Physical properties of ordinary water and common liquids (EE & BG Units)

| Liquid | Temp $T(^oF)$ | Density $\rho(\frac{Ib}{ft^3})$ | Density $\rho(\frac{Slug}{ft^3})$ | Specific gravity S | Absolute viscosity $\mu(\frac{Ib-\sec}{ft^2})$ | Kinematics viscosity $\nu(\frac{ft^2}{\sec})$ |
|---|---|---|---|---|---|---|
| Water | 32 | 62.4 | 1.940 | 1.00 | 3.75E-5 | 1.93E-5 |
| | 40 | 62.4 | 1.940 | 1.00 | 3.23E-5 | 1.66E-5 |
| | 60 | 62.4 | 1.938 | 0.999 | 2.36E-5 | 1.22E-5 |
| | 80 | 62.2 | 1.934 | 0.997 | 1.80E-5 | 9.30E-6 |
| | 100 | 62.0 | 1.927 | 0.993 | 1.42E-5 | 7.39E-6 |
| | 120 | 61.7 | 1.918 | 0.988 | 1.17E-5 | 6.09E-6 |
| | 140 | 61.4 | 1.908 | 0.983 | 9.81E-6 | 5.14E-6 |
| | 160 | 61.0 | 1.896 | 0.977 | 8.38E-6 | 4.42E-6 |
| | 180 | 60.6 | 1.883 | 0.970 | 7.26E-6 | 3.85E-6 |
| | 200 | 60.1 | 1.868 | 0.963 | 6.37E-6 | 3.41E-6 |
| | 212 | 59.8 | 1.860 | 0.958 | 5.93E-6 | 3.19E-6 |
| Mercury | 50 | 847 | 26.3 | 13.6 | 3.2E-5 | 1.2E-6 |
| | 200 | 834 | 25.9 | 13.4 | 2.6E-5 | 1.0E-6 |
| | 300 | 826 | 25.7 | 13.2 | 2.3E-5 | 9.0E-7 |
| | 400 | 817 | 25.4 | 13.1 | 2.0E-5 | 8.0E-7 |
| | 600 | 802 | 24.9 | 12.8 | 1.7E-5 | 7.0E-7 |
| Ethylene glycol | 68 | 69.3 | 2.15 | 1.11 | 4.16E-4 | 1.93E-4 |
| | 104 | 68.7 | 2.14 | 1.10 | 1.91E-4 | 8.93E-5 |
| | 140 | 68.0 | 2.11 | 1.09 | 1.03E-4 | 4.89E-5 |
| | 176 | 66.8 | 2.08 | 1.07 | 6.31E-5 | 3.04E-5 |
| | 212 | 66.2 | 2.06 | 1.06 | 4.12E-5 | 2.02E-5 |
| Ethyl alcohol(methanol) | 32 | 50.6 | 1.57 | 0.810 | 1.71E-5 | 1.09E-5 |
| | 68 | 50.0 | 1.55 | 0.801 | ------- | -------- |
| | 104 | 49.4 | 1.54 | 0.792 | 1.22E-5 | 7.93E-6 |
| | 140 | 48.9 | 1.52 | 0.783 | 1.07E-5 | 7.01E-6 |
| | 176 | 48.3 | 1.50 | 0.774 | 9.40E-6 | 6.25E-6 |
| | 212 | 47.8 | 1.49 | 0.765 | 8.27E-6 | 5.58E-6 |
| Ethyl alcohol(ethanol) | 32 | 50.3 | 1.56 | 0.806 | 3.70E-5 | 2.37E-5 |
| | 68 | 49.8 | 1.55 | 0.798 | 3.03E-5 | 1.96E-5 |
| | 104 | 49.3 | 1.53 | 0.789 | 2.51E-5 | 1.64E-5 |
| | 140 | 48.2 | 1.50 | 0.772 | 1.74E-5 | 1.16E-5 |
| | 176 | 47.7 | 1.48 | 0.754 | 1.24E-5 | 8.45E-6 |
| | 212 | 47.1 | 1.46 | 0.745 | ------- | ----- |
| Normal octane | 32 | 44.8 | 1.39 | 0.718 | 1.47E-5 | 1.06E-5 |
| | 68 | 43.8 | 1.36 | 0.702 | 1.13E-5 | 8.31E-6 |
| | 104 | 42.8 | 1.33 | 0.686 | 9.04E-6 | 6.79E-6 |
| Benzene | 32 | 56.2 | 1.75 | 0.900 | 1.90E-5 | 1.09E-5 |
| | 68 | 54.9 | 1.71 | 0.879 | 1.36E-5 | 7.99E-6 |
| | 104 | 53.6 | 1.67 | 0.858 | 1.05E-5 | 6.31E-6 |
| | 140 | 52.2 | 1.62 | 0.836 | 8.19E-6 | 5.05E-6 |
| | 176 | 50.9 | 1.58 | 0.815 | 6.87E-6 | 4.35E-6 |
| Kerosene | 0 | 52.5 | 1.63 | 0.841 | 1.48E-4 | 9.05E-5 |
| | 77 | 50.8 | 1.58 | 0.814 | 3.97E-5 | 2.55E-5 |
| Lubricating oil | 68 | 54.4 | 1.69 | 0.871 | 2.74E-4 | 1.61E-4 |
| | 104 | 53.6 | 1.67 | 0.858 | 1.42E-4 | 8.55E-4 |
| | 140 | 62.6 | 1.63 | 0.845 | 8.73E-5 | 5.33E-5 |
| | 176 | 51.9 | 1.61 | 0.832 | 5.91E-5 | 3.66E-5 |
| | 212 | 51.2 | 1.59 | 0.820 | 4.18E-5 | 2.63E-5 |
| | 248 | 50.5 | 1.57 | 0.809 | 3.22E-5 | 2.05E-5 |

# Appendix A

## Linear Algebra Routines

C function solveq:

Before describing the C (or CPP) linear algebra solver(s) it should be noted that arrays in C commonly begin with element 0, i.e., for a one-dimensional array b of length n the elements are stored in b[0], b[1],.....b[n-1], and two dimensional arrays are stored starting with [0][0], etc. In other words, in C (or C++) array dimensioned to N have a range of 0...N-1; i.e., somewhat of a inconvenience when thinking of strings of values that are N long. Thus, the first row of a matrix goes into the 0th subscript of the first subscript, and the first column uses a 0 for the second subscript, etc. C programmers should be accustomed to handling arrays of pointers for two dimensional arrays. CPP-programs will be more familiar with the use of arrays starting with 0 instead of 1. C was originally designed more for systems programmer, who rarely deals with two-dimensional arrays whose size is variable and known only at execution time, as is common in engineering computing. Thus, a function in C does not allow sizes of a two dimensional array to be passed as arguments, i.e. void linearsol(a,m,n) { float a[n][m];is not permissible. (CPP does allow a variation of this.). In C, two-dimensional arrays are handled as pointers to pointer, i.e.,**a. Thus for example with the declaration float **a; then the memory addressed by a[3][4], for example is to: (1) add 3 to the address of a and (2) take the memory thus addressed as a new address and add 4 to it, (3) return the value thus addressed.. If **a is the declaration then memory must be allocated with the library calloc, malloc, or (new in cpp). It is necessary that the correct prototypes be included in the code before the solver function(s) is called, either by including these as separate declarations, or by including a header file containing these declarations.

File solveq2.c, uses C pointers, i.e., the matrix is a pointer to a pointer **a, and the vector is a pointer *b. The second in file solveq.cpp utilizes the CPP capabilities of passing arrays as arguments to functions. This version requires that a global constant const N=...be declared that determines the dimensions of the matrix a and the known vector b in both the main (calling function) and the solver, i.e., the arrays are declared by a[N][N] and b[N]. Both of these solvers require that the beginning of the arrays be at 0; that is the first element of the matrix is at a[0][0] and the first element of the known vector is at b[0].

These functions of the linear algebra solver are described below.

Solveq2.c

The prototype for the function solveq is:

void solveq(int n,float **a,float *b,int itype,float *dd,int *indx);

in which n must contain the size of the arrays, i.e., n = the number of equations that are being solved a must contain the elements of the coefficient matrix. The first subscript of a corresponds to the rows minus 1 of the matrix, and the second subscript corresponds its columns minus 1, i.e., both the beginning row and column start with 0, and both end their range at n-1 b must contain the elements of the known vector, unless only the inverse of the coefficient matrix is asked for by the value given to itype. The first element is b[0] and its last value is in b[n-1].

itype must equal one of the following depending upon what is to be accomplished by solveq:

- =1 solves linear system of equations
- =2 produces inverse matrix (in a)
- =3 evaluate determinant, and gives value in dd
- =4 solve eqs. & Produces inverse matrix
- =5 evaluate determinant & produces inverse matrix
- =6 evaluate determinant, produce inverse matrix and solve eqs.

dd is a pointer that returns the determinant, if ask for by itype, and in the calling function must be declared by float *dd.

indx is pointer to work space for solveq that must have at least n allocated integer positions.

The calling function should contain declarations such as:

```
float **a,*b;  int *indx;
```

and statements such as the following to allocate the needed memory.

```
indx=(int*)calloc(nj,sizeof(int));

a=(float**)malloc(np*sizeof(float*)); //allocates pointers to rows
for(i=0;i<np;i++) a[i]=(float*)malloc(np*sizeof(float));// allocates rows & pointers to them
```

function solveq calls to functions: void dcompos(float **a,int n,int *indx,float d) and

void finsol(float **a,int n,int *indx,float *b).

The main program that calls on solveq can "include" the header file solveq2.h (which is listed below), or needs to contain similar declarations.

solveq2.h

```
void dcompos(float **a,int n,int *indx, float *d);
void finsol(float **a,int n,int *indx, float *b);
void solveq(int n, float **a, float *b, int itype, float *dd, int *indx);
```

solveq.cpp

The function solveq.cpp assumes that the standard is used of starting all arrays at 0, i.e., for example 4 x 4 coefficient matrix is to be solved that it is defined with float a[4][4] declared and that the calling program will have given values to

```
a[0][0] a[0][1].........a[0][N-1]

a[1][0] a[1][1]..........a[1][N-1]

a[N-1][0]...........a[N-1][N-1]
```

For example the beginning lines in the .cpp program might consist of the following:

```
#include <stdio.h>

#include <stdlib.h>

#include <math.h>
```

```
const int N=4;
```

```
void main(void){ int indx[N]; float b[N],a[N][N];
```

If solveq.cpp is compiled separately, and then linked to the main program the const int N= must have the same value in all elements. The prototype for this linear algebra solver is:

```
void solveq(int n, float a[N][N], float b[N], int itype, float &dd, int indx[N]);
```

The statement that executes the function solveq (which in turn executes the functions dcompos and finsol) could consist of the following:

```
solveq(N,a,b,itype,*dd,indx);
```

in which the arguments have the following meanings:

N is an int (which denotes integer in C) that equals the number of equations that are to be solved, or the size of the matrix that is to be solve. In other words, the main program will have defined the square matrix a with N rows and N columns and the known vector b with N rows. Both a and b begin with subscript 0 and end with subscript N-1.

a is the coefficient matrix, and must be a square matrix. It consists of the following elements that must be given values in the calling program.

a[0][0] a[0][1] . . . a[0][N-1]
a[1][0] a[1][1] . . . a[1][N-1]
a[N-1][0] . . . a[N-1][N-1]

and

b[0] b[1] . . . b[N-1]

The other arguments are as defined previously with the exception that dd that can receive the determinant, if asked for, uses the indirection & capability of CPP, rather than a pointer.

Notice that the first uses the function solveq (that requires that arrays begin with 0, the second uses the function solveq.cpp (these arrays also begin with 0), and the third uses solveone that has arrays begins with 1 and it allocates the memory with arrays beginning with 1 by calling on the allocation function that are included in the first of the listing, and that are prototyped in solveone.h. Notice that all of these programs open the file MAINSOL.DA1 that must contain the input data for this problem.

mainsol.c (Calls on solveq that uses pointers, and requires that all arrays begin with 0)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "solveq2.h"
void main(void){ int n, *indx,i,j,itype; float **a,*b,*dd; FILE *fil;
printf("Give N & ITYPE"); scanf("%d %d",&n,&itype);
b=(float*)calloc(n,sizeof(float));indx=(int*)calloc(n,sizeof(int));
a=(float**)malloc(n*sizeof(float*)); //allocates pointers to rows
for(i=0;i<n;i++) a[i]=(float*)malloc(n*sizeof(float));// allocates rows & pointers to them
fil=fopen("mainsol.da1","r");
for(i=0;i<n;i++){for(j=0;j<n;j++) fscanf(fil,"%f",&a[i][j]); fscanf(fil,"%f",&b[i]);}
solveq(n,a,b,itype,dd,indx);
if((itype==1) || (itype==3)) goto L1;
for(i=0;i<n;i++){for(j=0;j<n;j++) printf("%f ",a[i][j]); printf("%f\n",b[i]);}
L1: if((itype>2) && (itype!=4)) printf("Det = %f",*dd);
if((itype==1) || (itype==6)){printf("\nSolution:\n"); for(i=0;i<n;i++)printf("%f ",b[i]);}
printf("\n"); free(b);free(indx);for(i=n-1;i>=0;i--) free(a[i]); free(a);
}
```

mainsol.cpp (using array declarations rather than pointers, and &; some C++ features)

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
const N=4;
void solveq(int n,float a[N][N],float b[N],int itype,float &dd,int indx[N]);
void main(void){ int n,indx[N],i,j,itype; float a[N][N],b[N],dd; FILE *fil;
printf("Give N & ITYPE"); scanf("%d %d",&n,&itype);
fil=fopen("mainsol.da1","r");
for(i=0;i<n;i++){for(j=0;j<n;j++) fscanf(fil,"%f",&a[i][j]); fscanf(fil,"%f",&b[i]);}
solveq(n,a,b,itype,dd,indx);
if((itype==1) || (itype==3)) goto L1;
for(i=0;i<n;i++){for(j=0;j<n;j++) printf("%f ",a[i][j]); printf("%f\n",b[i]);}
L1: if((itype>2) && (itype!=4)) printf("Det = %f",dd);
if((itype==1) || (itype==6)){printf("\nSolution:\n"); for(i=0;i<n;i++)printf("%f ",b[i]);}
printf("\n");
}
```

Below there are two listing of Gaussian Elimination functions in C to solve a linear system of equations). The first, GAUSEL.C, uses the standard C arrays that begin with 0 and terminate with n-1.

Gausel.c (Arrays start with 0 & main must supply coef. matrix starting with a[0][0], etc.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define  eps 1.e-6
#define one 1.
#define onn -1.

int gausel(int n,float **a,float *b,float *det,float *errnor){

int n1,i,j,k,i1,*l;
float **ao,*bo,**xo,fac;
*det=one; n1=n-1;
l=(int *)malloc((unsigned)n*sizeof(int));
bo=(float *)malloc((unsigned)n*sizeof(float));
ao=(float **)malloc((unsigned)n*sizeof(float *));
        for(i=0;i<n;i++) ao[i]=(float *)malloc((unsigned)n*sizeof(float));
        xo=(float **)malloc((unsigned)n*sizeof(float*));
        for(i=0;i<n;i++) xo[i]=(float *)malloc((unsigned)n*sizeof(float));
/* Copies original a & b into ao & bo for later use in determining error. */
        for(i=0;i<n;i++){bo[i]=b[i];for(j=0;j<n;j++) ao[i][j]=a[i][j];}
/* pivots on largest row & keeps track of row no in l[k]; */
        for(k=0;k<n1;k++){fac=fabs(a[k][k]); l[k]=k;
        for(i=k+1;i<n;i++) if(fabs(a[i][k])>fac){fac=fabs(a[i][k]);l[k]=i;}
        if(fac<eps){printf("Matrix is singular"); return 1;
        }
        if(l[k] != k){
        *det *= onn;for(j=k;j<n;j++){
        fac=a[k][j];a[k][j]=a[l[k]][j];
        a[l[k]][j]=fac;
}
}

/* Gaussian elimination (only a because b must be done also for iterative cor.) */
        for(i=k+1;i<n;i++){
        fac=a[i][k]/a[k][k]; a[i][k]=fac;
        for(j=k+1;j<n;j++)a[i][j]-=fac*a[k][j];
        }
        }
/* Computes determinant det */
        for(i=0;i<n;i++) *det *= a[i][i];
        for(i1=0;i1<2;i1++){ /* now adjust b for orig. elim & iter. correction */
        for(k=0;k<n1;k++){if(l[k] != k){fac=b[k];b[k]=b[l[k]];b[l[k]]=fac;}
        for(i=k+1;i<n;i++) b[i]-=a[i][k]*b[k];}
/* Back substitution */
        xo[n1][i1]=b[n1]/a[n1][n1];
        for(i=n1-1;i>=0;i--){ fac=b[i];
        for(j=i+1;j<n;j++) fac-=a[i][j]*xo[j][i1]; xo[i][i1]=fac/a[i][i];}
        /* Computes residual vector  {r}={b}-[a]{x} */
        for(i=0;i<n;i++){ fac=bo[i];
        for(j=0;j<n;j++)fac-=ao[i][j]*xo[j][i1]; b[i]=fac;bo[i]=fac; } }
        for (i=0;i<n;i++){b[i]=xo[i][0]+xo[i][1];
        errnor[i]=fabs(xo[i][1])/(fabs(b[i])+eps); }
        free(l);free(bo);for(i=n-1;i>=0;i--) free(xo[i]);free(xo);
        for(i=n-1;i>=0;i--)free(ao[i]);free(ao);return 0;
        }
```

To solve linear system equations the following C main programs could be used. The first C program calls on gausel and using arrays beginning with 0, and the second calls on gausel1, and uses arrays beginning with 1.

MAINGAU.C

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define  eps 1.e-6
#define one 1.
#define onn -1.
int gausel(int n,float **a,float *b, float *dd,float *error);
void main(void){ int n,i,j; float **a,*b,*dd,*error; FILE *fil;
printf("Give N "); scanf("%d",&n);
 b=(float*)calloc(n,sizeof(float));
 error=(float*)calloc(n,sizeof(float));
 a=(float**)malloc(n*sizeof(float*)); //allocates pointers to rows
 for(i=0;i<n;i++) a[i]=(float*)malloc(n*sizeof(float));// allocates rows & pointers to them
fil=fopen("mainsol.da1","r");
        for(i=0;i<n;i++){for(j=0;j<n;j++) fscanf(fil,"%f",&a[i][j]); fscanf(fil,"%f",&b[i]);}
         j=gausel(n,a,b,dd,error);
        printf("\nSolution:\n"); for(i=0;i<n;i++)printf("%f ",b[i]);
        printf("\nDeterminant=%f\n",*dd);
        // free(b);free(error);for(i=n-1;i>=0;i--) free(a[i]); free(a);
        }
```

# APPENDIX B

# COMPUTER PROGRAM

```
/*===============================================================================*/
/*             ANALYSIS OF FLOW AND PRESSURE IN PIPE NETWORKS                      */
/*                                                                                 */
/*function one:                                                                    */
/*                                                                                 */
/*              Read an option data                                                */
/*              Read pipe data                                                     */
/*              Read PRV data                                                      */
/*              Read pump data                                                     */
/*                                File name:                                       */
/*                          a:tesetin,a:tesetout                                   */
/*===============================================================================*/

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#include<conio.h>
//#define MAX_SIZ   10
//#define MAX_PIP   40
//#define MAX_PUM   40
//#define MAX_FIX    40
//#define MAX_PRV    40
#define PI    3.141592654

//void  main(void);
FILE *fili,*filo,*fil1;
const k1=10;
void pip_data(int np,int JC[],int JA[],int JB[],double l[],double D[],double C[],
              double ml[],double fgn[],int AAA[]);
void prv_input(int nprv,int LY[],int LZ[],double EMIN[],int MPL[]);
void pumpdata(int AAA[],double AA[],double BB[],double CC[],double EE[],double FF[]
,
              double DD[],double GG[],double A3,double CQ,int NPUMP,int NIPE);

void main(void)
{

int  nd,nj,np,fu,nprv,uu,ck;
double  sw;//DDQ
int    LY[k1],LZ[k1],JC[k1],KCLO[k1];
double EMIN[k1];
int    JD[k1];
int    MPL[k1],KC[k1];
int    KIP[k1],KPI[k1],JX[k1];
int j;
//int K99=20;int J99=20;
double Q[k1],S[k1],V[k1];

int NIPE,J1,J2,JPIN,NXX,a,q;int NPUMP=0;
int JA[k1],JB[k1];
int JJUN[k1];
double AA[k1],BB[k1],CC[k1],EE[k1],FF[k1];
double A3,A4,A1,A2,CQ,P;
int JPIP[k1],AAA[k1];
double l[k1],D[k1],C[k1],ml[k1],fgn[k1];
double DD[k1],GG[k1];
int NTEP=0;//int NPUMP=0;
NXX=0;ck=0;
char fnam2[20],fnam1[20];

/*-------------------------------------------------------------------------------*/
```

```
printf("Give: input file name\r\n");
   scanf("%s",fnam1);
if((fili=fopen(fnam1,"w+"))==NULL){
   printf("File %s does not exist",fnam1);
   exit(0);
   }

printf("Give file name for output\r\n");
   scanf("%s",fnam2);
if((filo=fopen(fnam2,"w+"))==NULL) {
   printf("Cannot open output f. %s",fnam2);
   exit(0);
  }

/*============================================================================*/
/*variable option:                                                            */
/*                                                                            */
/*                ck:Check pipe connection junction                           */
/*                fu: flow unite(0=CFS;1=GPM;2=MGD;3=SI(LITER/Sec)            */
/*                np:number of pipes                                          */
/*                nj: number of junction                                      */
/*                nprv: number of PRV                                         */
/*                sw: specfic Gravity (0) water with SG=1.0                   */
/*                uu:kinematic viscosity(!=0 Darcy-Weisbach                   */
/*                                                                            */
/*============================================================================*/

fscanf(fili,"%d %d %d %d %d %d %lf %d",&ck,&fu,&np,&nj,&nprv,&nd,&sw,&uu);

for(j=1;j<=20;++j){
    KPI[j]=0;
    KC[j]=0;
    KCLO[j]=0;
    JD[j]=0;
    }
for(j=1;j<=20;++j){
    MPL[j]=0;
    JX[j]=0;
    KIP[j]=0;
   }

 /*get pressure grulator valve*/

  if(nprv!=0){
  prv_input(nprv,LY,LZ,EMIN,MPL);
  }

/*get pipe data*/

 pip_data(np,JC,JA,JB,l,D,C,ml,fgn,AAA);

    if(sw==0){
      sw=1.0;
    }
   if(fu==0){//-----------------------------------------------------1
   fprintf(filo,"\nFlow Rate Is Expressed In CfS And Pressure In Psig");
   A1=4.73;A2=0.02517;A3=8.18/sw;A4=12.0;CQ=1.0;
   }else if(fu==1){
   fprintf(filo,"\nFlow Rate Is Expessed In gpm And Pressure In psig");
   A1=4.73;A2=0.02517;A3=8.18/sw;A4=12.0;CQ=448.86;
   }else if(fu==2){
   fprintf(filo,"\nFlow Rate Is Expressed In mgd And Pressure In psig");
   A1=4.73;A2=0.02517;A3=8.18/sw;CQ=0.646358;
   }else{
   fprintf(filo,"\nFlow Rate Is Expressed In Liter/Sec And Prssure IN (KN/M*M)");
   A1=10.69;A2=0.08265;A3=0.10197/sw;A4=100.0;CQ=1000.0;
   }
   if(uu==0){P=1.852;
      }else{P=2.0;
      //printf("\n THE DARCY-WEISH HEAD LOSS EQUATION IS USED UU=");
      }

 for(j=1;j<=np;++j){
     NIPE=j;
    fprintf(filo,"\n NIPE=%d",j);
    KIP[NIPE]=j;
    fprintf(filo,"\nKIP[%d]=%d",NIPE,j);
    KPI[j]=NIPE;
    fprintf(filo,"\nKPI[%d]=%d",j,NIPE);
    J1=JA[j];
    fprintf(filo,"\nJ1=%d",J1);
    J2=JB[j];
    fprintf(filo,"\nJ2=%d",J1);
 if(MPL[NIPE]==101){
    goto Ntep;
    }
 if((J1+J2)<= abs(J1-J2)){
    Ntep:
    ;
    NTEP=NTEP+1;
    fprintf(filo,"\n NTEP=%d",NTEP);
    JD[j]=2;
 if(NTEP==1){
    JPIN=j;
    }
 if(MPL[NTEP]==101){
    JJUN[NTEP]=JB[j];
    }else{
     JJUN[NTEP]=JA[j]+JB[j];
    }
    JPIP[NTEP]=j;
 }

if(AAA[j]!=0){
NPUMP=NPUMP+1;
pumpdata(AAA,AA,BB,CC,EE,FF,DD,GG,A3,CQ,NPUMP,NIPE);
}
/*----------------------------------------------------------------------------*/
fprintf(filo,"\n number of pump in the system NPUMP=%d",NPUMP);
 if(JA[j]!=0){
    a=JA[j];
    JX[a]=JA[j];
  }
 if(JB[j]!=0){
   q=JB[j];
   JX[q]=JB[j];
  }
/*----------------------------------------------------------------------------*/
 if(JC[j]==1){
     if(nd==0 && KC[j]==0){
     fprintf(filo,"\nThere Is a Chek Valve in Line Number (%d)",NIPE);
  }
 }
 if(JC[j]==2 ){
   KCLO[j]=1;
   if(nd==0){
   printf ("\nThe line Number(%d) Is Closed ",NIPE);
   }
}
  D[j]=D[j]/A4;
  Q[j]=PI * pow(D[j],2.0);
  S[j]=A1 * l[j] / pow(C[j],P)* pow(D[j],4.87);/*P=1.852*/
  V[j]=A2 * ml[j]/(pow(D[j],4.0));
  fprintf(filo,"\n %d %d %lf %lf %lf %lf",NXX,JPIN,D[j],Q[j],S[j],V[j]);

  }/*-----------------<<<<<<<<<land of the j loop*/
 }

/*============================================================================*/
/*                              PRV DATA                                      */
/*                                                                            */
/*function data                                                               */
/*                                                                            */
/* read LY[],LZ[],EMIN[]                                                      */
/* LY[] : JUNCTION NODE REPRESENT FIRST PRV,                                  */
/* LZ[] : PIPE NUMBER DOWNSTREAM FROM FIRST PRV,                              */
/* LY[] : GRADE WHICH THE PRV  SET TO MAINTAIN,                               */
/* MPL[]=101 idicated that this pipe containe PRV                             */
/*                                                                            */
/*                                                                            */
/*============================================================================*/

void prv_input(int nprv,int LY[],int LZ[],double EMIN[],int MPL[])

  {

  int j,J1;

  for(j=1;j<=nprv;++j){

           printf("\nJUNCTION NODE REPRESNET FIRST PRV LY[%d]=%d",j,LY[j]);
           scanf("%d",&LY[j]);
           fprintf(fili,"\nPIPE NUMBER DOWENSTREAM FROM FIRST PRV LZ[%d]=",j);
           scanf("%d",&LZ[j]);
           printf("\nGRADE WHICH THE (PRV)SET TO MAINTAIN EMIN[%d]=",j);
           scanf("%lf",&EMIN[j]);
           J1=LZ[j];
           MPL[J1]=101; //MPL[]=101 there is PRV IN the pipe

           }
  }

/*============================================================================*/
/*                            Pipe data                                       */
/*function data:                                                              */
/*             np: number of pipes                                            */
/*             JA-JB:Code number connecting the pipe                          */
/*             1 :length                                                      */
/*             D :diameter                                                    */
/*             C:HAZEN-WILLIAMS roughhness                                    */
/*             ml: minore losses                                              */
/*             fgn: fixed grade node                                          */
/*            AAA:PUMP Char                                                   */
/*                                                                            */
/*============================================================================*/
```

```
void pip_data(int np,int JC[],int JA[],int JB[],double l[],double D[],double C[],
              double ml[],double fgn[],int AAA[])
     {

    int j;
     //printf("\n number of pipe KK=",KK);
     //scanf("%d",&KK);
     for(j=1;j<=np;++j){
    scanf("%d",&JC[j]);
    fprintf(fili,"\ncode 2:indicate this pipe is closed code 1:check valve JC[j]=%d
",j,JC[j]);
    scanf("%d",&JA[j]);    /*N1*/
    fprintf(fili,"\n code number conncting this pipe JA[%d]=%d",j,JA[j]);
    scanf("%d",&JB[j]);   /*N2*/
    fprintf(fili,"\ncode number  conncting this pipe JB[%d]=%d",j,JB[j]);
    scanf("%lf",&l[j]);
    fprintf(fili,"\n line  length  in feet R[%d]=%lf",j,l[j]);
    scanf("%lf",&D[j]);
    fprintf(fili,"\n inside  diameter D[%d]=%lf",j,D[j]);
    scanf("%lf",&C[j]);
    fprintf(fili,"\n HAZEN-WILLIAMES roughness C[%d]=%lf",j,C[j]);
    scanf("%lf",&ml[j]);
    fprintf(fili,"\n sum of the mainor loss for this pipe VVVV[%d]=%lf",j,ml[j]);
    scanf("%d",&AAA[j]);
    fprintf(fili,"\n pump char(+,-)(uesfule power,opertaing data)AAA[%d]=%d",j,AAA[
j]);
    scanf("%lf",&fgn[j]);
    fprintf(fili,"\n fixed grade node ENGY[%d]=%lf",j,fgn[j]);

  }
 }
/*============================================================================*/
/*                               PUMP DATA                                    */
/*                                                                            */
/*AAA:  input for  every pipe in the networks                                 */
/*     (0) there is no pmup in the pipe                                       */
/*     (number(-,+)there is  a pmup in the pipe                               */
/*     npump=npump+1;                                                         */
/*     if(jc[j] !=2)  jc[j]=0                                                 */
/*     kc[j]=npump isto indiacte if there is pump in the line or not =0Nopmup */
/*     AA(NPUMP)=AAA    UESFUL POWER                                          */
/*     BB(NPUMP)=0                                                            */
/*     I=NPUMP                                                                */
/*----------------------------------------------------------------------------*/
/*    (+)UESFULE POWER                                                        */
/*    (-)OPERATING DATA(-1,-2)                                                */
/*    (-1)PAIRS OF HEAD-DISCHARGE DATA                                        */
/*    (-2)USEFUL POWER BE COMPUTED FROM THE OPERTAING DATA                    */
/*          AA[I=NPMUP]=X1*Y1+X2*Y2+X3*Y3/A3*3*CQ                             */
/*         x,y: (head,flow rate)                                              */
/*============================================================================*/

void pumpdata(int AAA[],double AA[],double BB[],double CC[],double EE[],double FF[]
,
              double DD[],double GG[],double A3,double CQ,int NPUMP,int NIPE)
{

double X1,Y1,X2,Y2,X3,Y3,S,D,T1;
int  j,I;
int NXX=0;
int KC[k1];
for (j=1;j<2;++j){
      KC[j]=NPUMP;
      AA[NPUMP]=double(AAA[j]);
      BB[NPUMP]=0;
      I=NPUMP;

if(AAA[j]<0){//------
  printf("\n x1=");// h
  scanf("%lf",&X1);
  printf("\n y1=");//Q
  scanf("%lf",&Y1);
  printf("\nx2=");
  scanf("%lf",&X2);
  printf("\ny2=");
  scanf("%lf",&Y2);
  printf("\nX3=");
  scanf("%lf",&X3);
  printf("\ny3=");
  scanf("%lf",&Y3);

 if(X1<X2  || X2<X3){

   printf("\n PUMP DATA FOR LINE (%d)IS NOT SIUTABLE ",NIPE);
   NXX=1;
   }
   AA[I]=(X1 * Y1 + X2 * Y2 + X3 * Y3)/(A3 * 3.0 * CQ);
 if(AAA[j]>-2){

    AA[I]=Y1/CQ;
    S=log(X1-X3)/(X1-X2);
    D=log(Y3/Y2);
    BB[I]=S/D;
    T1=pow((Y2/CQ),BB[I]);
    CC[I]=(X1-X2)/T1;
    EE[I]=-BB[I]* CC[I]* pow((Y3/CQ),(BB[I]-1.));
    FF[I]=X3-EE[I]* Y3/CQ;
    DD[I]=X1;
    GG[I]=Y3/CQ;

fprintf(filo,"\n%lf %lf %lf %lf %lf %lf %lf",AA[I],BB[I],CC[I],EE[I],FF[I],DD[I],GG
[I]);

   }
  }//----AAA
 }
printf("\n NXX=%d",NXX);

}
```

```
/*==========================================================================*/
/*   computer analysis of flow and pressure in pipe networks                */
/*                                                                          */
/*function two:                                                             */
/*                                                                          */
/*NTEP -NUMBER OF FIXED GRADE NODE IN TE SYSTEM                             */
/*JJ- COUNTER TO CHECK THE NUMBER OF JUNCTION                               */
/*KN: N OF Junction In The System                                           */
/*JA-JB:JUNCTION CONNECTED THE PIPe                                         */
/*IP[][]-MATRIX FOR EACH JUNCTION IN THE SYSTEM,                            */
/*EMIN[]- GRADE WHICH THE PRV SET                                           */
/*LZ[]- PIPE NUMBER DOWEN STREAM FROM TEH FIRST PRV                         */
/*NOP-NUMBER OF PIPE AROUND EACH JUNCTION                                   */
/*NEL- NUMBER OF PIPE AROUND ALL THE JUNCTION IN THE SYSTEM                 */
/*KJ- KN-1..NUMBER OF JUNCTION -1                                           */
/*KP- KK-KN...NUMBER OF PIPE - NUMBER OF JUNCTION                           */
/*LNUM- NUMBER OF JUNCTION                                                  */
/*KK- N number of pipes                                                     */
/*                                                                          */
/*                           file Nmae     monain.monao                     */
/* =========================================================================*/

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#define MAX_SIZ   50
#define MAX_PIP   40
#define MAX_PUM   40
#define MAX_FIX    40
#define MAX_PRV    40
#define MAX_CPIP   3

FILE *fili,*filo,*fil1;

void main(void)
{


int KN,KK,NR,NTEP;
int J99=30;
int LZ[MAX_PRV],NUMJ[10];
int IP[MAX_PIP][13],M[MAX_PIP],JA[MAX_PIP],JB[MAX_PIP];
int MPL[MAX_SIZ],JX[MAX_SIZ],JIJ[MAX_SIZ],KIP[MAX_PIP],KPI[MAX_PIP],JJI[MAX_PIP];
int JJ,NEL,NOP,J1,J2,MMM,JMAX;
int KTEP ;
int ERR=0;
int j,m,h,v,t,g,k,i;
double ENGY[MAX_PIP],EMIN[MAX_SIZ];
char fnam1[20],fnam2[20];


/*==========================================================================*/

printf("\Give file name for input\r\n");
      scanf("%s",fnam1);
      if((fili=fopen(fnam1,"r+"))==NULL){
      printf("Cannot open input f.%s",fnam1);
      exit(0);
   }
printf("Give file name for output\r\n");
      scanf("%s",fnam2);
      if((filo=fopen(fnam2,"a+"))==NULL) {
```

```
      printf("Cannot open output f. %s",fnam2);
      exit(0);
  }

/*==============================================================================*/

 fscanf(fili,"%d %d %d %d",KN,KK,NR,NTEP);

 for(j=1;j<=KK;++j){
 fscanf(fili,"%d %d %d %d %d",JA[j],JB[j],KPI[j],KIP[j],MPL[j]);
 }
 for(i=1;i<=KN;++i){
     fscanf(fili,"%d %d",NUMJ[i],JX[i]);
     }
 for(j=1;j<=50;++j){
     M[j]=0;
     }
    KTEP=NTEP-1;
    JJ=0;
    NEL=0;


for(j=1;j<=J99;++j){//-------------------------->>j
     if(JX[j]!=0){ // if JX[j]=0 means fiexd grade node
       JMAX=j;
        JJ=JJ+1;
        fprintf(filo,"\n JJ=%d",JJ);
        JJI[JJ]=j;
        fprintf(filo,"\nJJI[%d]=%d",JJ,j);
        JIJ[j]=JJ;
        fprintf(filo,"\nJIJ[%d]=%d",j,JJ);
        printf("\n------marix--------- =");
        scanf("%d",&ERR);
    }
 }//-------------------------------<<j

if(JJ!=KN){//JJ is COUNTER to CHECK the number of junction
   KN=JJ;
 }

for(j=1;j<=KK;++j){
     JJ=KPI[j];
     fprintf(filo,"\nJJ=%d",KPI[j]);
     J1=0;
     J2=0;
if(JA[j]!= 0){
    t=JA[j];
    J1=J1+JIJ[t];
    fprintf(filo,"\nJ1=%d",JIJ[t]);
 }
if(JB[j]!=0){
    g=JB[j];
    J2=J2+JIJ[g];
    fprintf(filo,"\nJ2=%d",JIJ[g]);
  }
if(J1!= 0){
   M[J1]=M[J1]+1;
   fprintf(filo,"\nM[%d]=%d",J1,M[J1]);
   MMM=M[J1];
   fprintf(filo,"\nMMM=%d",MMM);
   IP[J1][MMM]=j;
   fprintf(filo,"\nIP[%d][%d]=%d",J1,MMM,j);
   printf("\n------ MATRIX[]--------- =%d",ERR);
   scanf("%d",&ERR);
 }
if(J2!=0){
   M[J2]=M[J2]+1;
   fprintf(filo,"\nM[%d]=%d",J2,M[J2]);
   MMM=M[J2];
   fprintf(filo,"\nMMM=%d",MMM);
   IP[J2][MMM]=-j;
   fprintf(filo,"\nIP[%d][%d]=-%d",J2,MMM,j);
   printf("\n--------- MATRIX[] ----------=",ERR);
   scanf("%d",&ERR);
 }
if(MPL[JJ]==101){
   JA[j]=0;
 }
}

/*------------------------------------------------------------------------------*/

if(NR!=0){
for(j=1;j<=NR;++j){
    v=LZ[j];//LZ[j]= pipe number dowenstream from first prv
    JJ=KIP[v];
    ENGY[JJ]=EMIN[j];//the enrgy for this pipe is equal to grade which prv set
  }
}

for(j=1;j<=KN;++j){ //--------------------->>>>>>>>>>>>M
       NOP=M[j]; //sum of the pipe around each junction
       fprintf(filo,"\n NOP=%d",NOP);
      if(NOP !=0){
      for(k=1;k<=NOP;++k){
        NEL=NEL+1;  //the sum of all the pipe in all junction in the system
        fprintf(filo,"\n NEL=%d",NEL);
        MPL[NEL]=IP[j][k]; //setting the matrix
        fprintf(filo,"\n ----MPL[%d]=IP[%d][%d]",NEL,j,k);
       }
      M[j]=NEL-NOP+1;
      fprintf(filo,"\n M[%d]=%d-%d+1",j,NEL,NOP);
      printf("\n--------matrix ----------=%d",ERR);
      scanf("%d",&ERR);
   }
 }

/*------------------------------------------------------------------------------*/

//KJ=KN+1;
//KP=KK-KN;
//LNUM=KN;
//NZX=0;
M[KN+1]=NEL+1;
for(j=1;j<=KK;++j){
    if(JA[j]!=0){
      m=JA[j];
      JA[j]=JIJ[m];
      }
    if(JB[j]!=0){
       h=JB[j];
       JB[j]=JIJ[h];
      }
 }
```

```
  fprintf(filo,"\n KTEP=%d      JMAX=%d ",KTEP,JMAX);
 }
```

```
/*============================================================================*/
/*              computer analysis of flow and pressure in pipe networks       */
/*  FUNCTION Three:                                                           */
/*              GGGG[]-DEMAND OF THE JUNCTION                                 */
/*              EEEE[]-ELEVATION OF THE JUCNTION                              */
/*              NUMJ[]- NUMBER OF JUNCTION                                    */
/*              JIJ[]-ARRAY CONTAINE THE NUMBER OF JUNCTION                   */
/*              MPL[]-ARRAY CONTINE THE NUMBER OF PIEP CONNECTED TO THE JUNCTI*/
/*              NCK-NONZERO INTRY WILL INDICATED TO CHECKED PIPE CONNECTED J  */
/*              NSD NONZERO INTRY WOULDN'T SHOW DATA OF THE JUNCTION          */
/*                                                                            */
/*              file name; mohamedin-mohamedout                               */
/*                                                                            */
/*============================================================================*/

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#define MAX_SIZ   50
#define MAX_PIP   40
#define MAX_PUM   40
#define MAX_FIX    40
#define MAX_PRV    40
#define MAX_CPIP   3

FILE *fili,*filo,*fil1;
void main()
{

int NCK,NSD,KN,KK,NTEP;
int NPO[10],NUMJ[10];
int JG[MAX_PIP],M[MAX_PIP],JA[MAX_PIP],JB[MAX_PIP];
int MPL[MAX_SIZ],JX[MAX_SIZ],JIJ[MAX_SIZ],KIP[MAX_PIP],KPI[MAX_PIP],JJI[MAX_PIP];
int NEL,NOP,L,J1,J2,MAXJ;
int MBEG,KJ,LNUM,KP,NAB,NZX,NXX,NUJ,KTEP;
int j,i,w1,l,k,tem,II;
double E[MAX_PIP],BI[MAX_PIP],B[MAX_PIP];
double GGGG[10],EEEE[10];
double CQ;
char fnam1[20],fnam2[20];

/*============================================================================*/

printf("Give file name for input\r\n");
      scanf("%s",fnam1);
      if((fili=fopen(fnam1,"r+"))==NULL){
      printf("Cannot open input f. %s",fnam1);
      exit(0);
      }
printf("Give file name for output\r\n");
   scanf("%s",fnam2);
if((filo=fopen(fnam2,"a+"))==NULL) {
   printf("Cannot open output f. %s",fnam2);
   exit(0);
  }

/*============================================================================*/

fscanf(fili,"%d %d %d %d %d %d %lf",NCK,NSD,KN,KK,NTEP,NEL,CQ);

for(j=1;j<=KK;++j){
   fscanf(fili,"%d %d %d %d",JA[j],JB[j],KPI[j],KIP[j]);
   }
  for(j=1;j<=KN+1;++j){
     fscanf(fili,"%d",M[j]);
     }
     for(i=1;i<=KN;++i){
        fscanf(fili,"%d %d %d",NUMJ[i],JG[i],JX[i]);
        }
     for(j=1;j<=NEL;++j){
     fscanf(fili,"%d",MPL[j]);
     }

KTEP=NTEP-1;
NEL=0;
KJ=KN+1;
KP=KK-KN;
LNUM=KN;
NZX=0;
//M[KN+1]=NEL+1;

for(k=1;k<KJ;++k){//------------------------------->>S

   printf("\nDemand of the junction GGGG[%d]=",k);
   scanf("%lf",&GGGG[k]);
   printf("\nElevation of the junction EEEE[%d]=",k);
   scanf("%lf",&EEEE[k]);
   printf("\nNumber of the junction NUMJ[%d]=",k);
   scanf("%d",&NUMJ[k]);

  for(j=1;j<=MAX_CPIP;++j){   // printf("\nJG[%d]=",j);
   scanf("%d",&JG[j]);
   }

 if(k==KJ){break;}

      tem=NUMJ[k];
      L=JIJ[tem];//NUMBER OF JUNCTION
  if(L==0){ //MEAN THERE IS NO JUNCTION NUMBER WRONG DATA HAS BEEN GIVEN
  fprintf(filo,"\nData is input for Jun nod NUMJ(%d)WHICHISNOT used INTHE PIPEDATA"
,NUMJ,k);
  NXX=1;
 }
   B[L]=-GGGG[k]/CQ;
   BI[L]=B[L];
   E[L]=EEEE[k];

 if(NCK!=0){//not equaleto zero the mean the data for teh system must be chekcd
   NUJ=M[L+1]-M[L];//NUJ THE NUMBER OF PIPE AROUND EACH JUNCTION
   fprintf(filo,"\n+++++++NUJ=%d",NUJ);
  for(i=1;i<=MAX_CPIP;++i){//------->>i
     if(i<=NUJ){
        w1=M[L];//L NUMBER OF JUNCTION
        NAB=MPL[w1+i-1];//MPL ARRAY CONTAINE THE NUMBER OF PIPE CONN TH JUNCTI
        NAB=abs(NAB);
        NAB=KPI[NAB];
        fprintf(filo,"\n+++++++NAB=%d",NAB);
        }else{
        NAB=0;
        }
     if(abs(NAB)!=JG[i]){//
       fprintf(filo,"\nDATA IN PUT FOR PIPE CONNECTION JUNC(%d)DOES NOT CHECK",NUMJ
);
       NZX=1;
       }
    }//---------------<<<<<i
  }//------------------------------------<<<<<<
}//-----------------------------------<<k loop--S

if(NCK !=0 && NZX==0){//PIPE CONEECTED JUNCTION HAS BEEN CHECKED

   fprintf(filo,"\nSuccessfuly Geometric Verification Hase Been Complted");
   }
if(NZX==1){//ERROR IN THE INPUT DATA
   NXX=1;
   fprintf(filo,"\n Error in the input data");
   exit(0);
   }
if(NSD==0){//WRITE THE DATA FOR THE JUNCTION NODE
   fprintf(filo,"\n Junction No#  Demand  Elevation  Conccection Pipe#");
   }
for(l=1;l<=KN;++l){//------------------------------>>>>>>p
    NUMJ[l]=JJI[l];
    MAXJ=M[l+1]-1;
    MBEG=M[l];
    NOP=M[l+1]-M[l];
    fprintf(filo,"\n NOP=%d",NOP);
   for(J1=1;J1<=NOP;++J1){
      J2=MPL[MBEG+J1-1];
      NPO[J1]=abs(J2);
      }
  GGGG[l]=-B[l] * CQ;//IF B[L]==0>>>>>>GGGG=0;

 if(NSD<=0){//------------------------->>
 //printf("\n  NUM[%d]=%d   GGGG[%d]=%lf ",l,NUMJ[l],GGGG[l]); //E[l];
 for(j=1;j<=NOP;++j){//----------->>
   J2=MPL[MBEG+j-1];
   NPO[j]=abs(J2);
   II=NPO[j];
   fprintf(filo,"\nJ2---------->>=%d",J2);
   printf("\r\n");
   printf("\nKPI[%d]=%d",II,KPI[II]);

   }//-------------------------->>>j
  } //--------------------------------->>>if
 }
 fprintf(filo,"\nJ2=%d KTEP=%d NTEP=%d NEL=%d KJ=%d ",J2,KTEP,NTEP,NEL,KJ);
 fprintf(filo,"\nKP=%d LNUM=%d MAXJ=%d ",KP,LNUM,MAXJ);
 fprintf(filo,"\n NXX=%d",NXX);

}
```

```
/*============================================================================*/
/*         computer analysis of flow and pressure in pipe netwoks            */
/*        function Four:                                                      */
/*                                                                            */
/*  JA,JB: NODE CONNECTED PIPE  ja[] or jb[]==0 if it is FGN                  */
/*  IP:FOR EACH JUNCTION IN THE NET AND A PIPE CONECCTED THE JUNCTION         */
/*  M:NUMBER OF PIPE CONEECTED EACH JUNCTION                                  */
/*  JD:==2 FOR EACH FIXED GRADE NODE IN THE NET OTEHR ==0                     */
/*  PIN:N OF PIPE conect to THE FIRST FIXED GRADE NODE IN THE NET>>>NTEP==1   */
/*  KN:NUMBER OF JUNCTION IN THE NET                                          */
/*  KK: NUMBER OF PIP IN THE NET                                              */
/*  KJ:=KN+1                                                                  */
/*  KP:=KK-KN                                                                 */
/*  MPL:FOR EACH JUNCTION HOW MANY PIPE CONNEDTED EACH JUNCTION               */
/*                                                                            */
/*     the function do spefied the path and the loops equation for the system */
/*============================================================================*/

#include<stdio.h>
#include <math.h>
#include<stdlib.h>

#define  MAX_SIZ   80

// void combine(int JF[],int NZ);
// void combine2(int KN,int NP,int IP[20][13],int JA[],int JB[],int JD[],int M[],
               //int M6,int M7,int M8,int M9);
 //void combine3(int KN,int IX[],int JE[]);
 FILE *fili,*filo,*fill;
 void combine4(int MPL[],int M[],int NA[],int NB[]);
 void main(void)
{

int KN,KP, NEL,KJ,KK,NTEP,LNUM;
int M[MAX_SIZ],JD[MAX_SIZ],JG[MAX_SIZ],IX[20],MPL[MAX_SIZ],JE[MAX_SIZ];
int NA[20],NB[20],JA[20],JB[20],IP[20][13],JF[MAX_SIZ];
int KL,J8,M1,NP,L8,L9,NZ,N8,j,k,I,jj;
int JPIN,NZX,NXX;
int M6=0;int M7=0;int M8=0;int M9=0;
int J7=1;int J6=0;
int JXNIN,JXNJJN,JEM;
int i,ii,JJJJ,JJJ1,jjj;
int ERR=0;
char fnam1[20],fnam2[20];

/*============================================================================*/

printf("Give file name for input\r\n");
scanf("%s",fnam1);
if((fili=fopen(fnam1,"r+"))==NULL){
printf("Cannot open input f. %s",fnam1);
exit(0);
}
printf("Give file name for output\r\n");
   scanf("%s",fnam2);
if((filo=fopen(fnam2,"a+"))==NULL) {
   printf("Cannot open output f. %s",fnam2);
   exit(0);
  }

/*============================================================================*/
```

```
   fscanf(fili,"%d %d %d %d %d %d %d",KN,KP,NEL,KJ,KK,NTEP,LNUM);
   fscanf(fili,"%d %d %d",JPIN,NZX,NXX);

  for(j=1;j<=KK;++j){
       fscanf(fili,"%d %d %d",JA[j],JB[j],JD[j]);
      }
    for(j=1;j<=KN+1;++j){
         fscanf(fili,"%d",M[j]);
       }
   for(j=1;j<=KN;++j){
      for(i=1;i<=3;++i){
          fscanf(fili,"%d",IP[j][i]);
       }
    }

  KL=KP-NTEP+1;

/* call function */
combine4(MPL,M,NA,NB);

JD[JPIN]=1;
M1=KN;
J8=JA[JPIN]+JB[JPIN];
if(JA[JPIN]>JB[JPIN]){
   JG[J8]=-JPIN;
  }else{
   JG[J8]=JPIN;
  }
  Age:
 ;
 JE[J8]=1;
 L9=1;
 IX[1]=J8;
/*----------------------------------------------------------------------------*/
for(k=1;k<=KN;++k){
    NZ=0;
    L8=IX[k];
  if(J7!=2 && L8==0){
     fprintf(filo,"\n this system is disconnected");
     NXX=1;
   }
  if(J7==2 && L8==0){
     break;
  }

  NP=M[L8+1]-M[L8];
 for(j=1;j<=NP;++j){
     N8=IP[L8][j];
     N8=abs(N8);
     if(JA[N8]!=L8 && JB[N8]!=L8){
        continue;
        }else{
        J8=JA[N8]+JB[N8]-L8;
        }
    if(JD[N8]==1){
       continue;
       }
   if(abs(J8-J6)>0){goto Rod;}
      NZ=NZ+1;
      M1=M1+1;
   if((M1-KN)<=NTEP){
      NB[M1-KN]=N8;
  }
   JF[NZ]=N8;
 if(J7!=2){
    JD[N8]=1;
   }
 if(JA[N8]!=L8){
    JF[NZ]=-N8;
   }
   Fog:
   ;
  J8=JA[N8]+JB[N8]-J8;
 if(J8!=J6){
 goto Gof;
 }
    if((M1-KN)<=NTEP){
       NA[M1-KN]=N8;
       fprintf(filo,"\n NA[%d-%d]=%d",M1,KN,N8);
   }

  for(ii=2;ii<=NZ;++ii){//-----------------------------------ii
      JJJJ=ii-1;
    for(jj=1;jj<=JJJJ;++jj){
        JXNIN=JF[ii];
        JXNJJN=JF[jj];
      if(abs(JXNIN)>=abs(JXNJJN)){
         continue;
        }else{
        JJJ1=jj+1;
        JEM=JF[ii];
     for(jjj=JJJ1;jjj<=ii;++jjj){
        JF[ii+JJJ1-jjj]=JF[ii+JJJ1-jjj-1];
      }
      JF[jj]=JEM;
   }
   break;
   } //--------jj
  }//---------------------ii
   LNUM=LNUM+1;
   M[LNUM]=NEL+1;
   fprintf(filo,"\n M[%d]=%d",LNUM,NEL+1);
for(jj=1;jj<=NZ;++jj){
    NEL=NEL+1;
    fprintf(filo,"\n NEL=%d",NEL);
    MPL[NEL]=JF[jj];
    fprintf(filo,"\nMPL[%d]=%d",NEL,MPL[NEL]);
    printf("\n---------flag ----------=%d",ERR);
    scanf("%d",&ERR);
 }//-------------------->>jj
NZ=0;
if(J7==2){goto Again;}
  if(J7==1){
   continue;
   }
   Gof:
   ;
 N8=abs(JG[J8]);
 NZ=NZ+1;
 JF[NZ]=JG[J8];
 goto Fog;
  Rod:
  ;
 if(JE[J8]!=1){
      L9=L9+1;
      IX[L9]=J8;
      JE[J8]=1;
      JG[J8]=N8;
   if(abs(J8-JB[N8])!=0){
      JG[J8]=-N8;
     }
    }
  }//---------j

 }//------->>k
Again:
;
M9=0;
for(j=1;j<=KN;++j){ //------------------------------->J
    M7=0;
    NP=M[j+1]-M[j];
    printf("\n NP=%d",NP);
     for(k=1;k<=NP;++k){//---------------------------->K
         M8=IP[j][k];
         M8=abs(M8);
         if(JA[M8]!=j && JB[M8]!=j){
            continue;
            }
            if(JD[M8]<1){
              M7=M7+1;
              M6=M8;
             }
       } //---------------------->K
if(M7==1){
   M9=1;
   JD[M6]=1;
 }
}//-----------------------------------------------------------2--->J
 if(M9==1){goto Again;}
   if(M7==2){goto Num;}
I=1;
Sum:
;
if(JD[I]<0){
for(k=1;k<=KN;++k){//---------------->d
    IX[k]=0;
    JE[k]=0;
 }//------------------->d
 J6=JA[I];
 J8=JB[I];
 J7=2;
 JG[J8]=I;
 JD[I]=1;
goto Age;
}
 Num:
 ;
 I=I+1;
if(I<=KK){goto Sum;}

  M[KK+1]=NEL+1;
  KL=LNUM-KN-NTEP+1;
  M[KK+1]=NEL+1;
  printf("\n--------- stop ----------=%d",ERR);
  scanf("%d",&ERR);
 fprintf(filo,"\n KK=%d  KN=%d   KL=%d    NTEP=%d  KJ=%d",KK,KN,KL,NTEP,KJ);
 fprintf(filo,"\n NXX=%d    NZX=%d",NXX,NZX);

  M[KK+1]=NEL+1;
if(LNUM !=KK){
   fprintf(filo,"\nXXXXXXXXX[The Relation[P=J+L+T-1]IS NOT SATESFIED]XXXXXXXXX");
   NXX=1;
   }
   if(NXX!=0){
     fprintf(filo,"\nXXXXXXXX[Error in the data]XXXXXXXXXXXXXX");
     }
 }
/*===============================================================================*/
/*                                                                               */
/*===============================================================================*/
void combine4(int MPL[],int M[],int NA[],int NB[])
   {

  int j,i;
for(j=1;j<80;++j){
    MPL[j]=0;
    }
    for(j=1;j<70;++j){
    M[j+5]=0;
    }
for(i=1;i<20;++i){
    NA[i]=0;
    NB[i]=0;
    }
  }
```

```
/*==============================================================================*/
/*              computer analysis of flow and peressure in pipe networks        */
/* function Five:                                                               */
/*            KTEP= THE NUMBER OF FGN-1                                         */
/*            KN- NUMBER OF JUNCTION IN THE SYSTEM                              */
/*            B[]-THE DEFFERENT OF ENERGY BETWEEN EACH THE FGN (J+KN)           */
/*            MAXT- MAXMUM NUMBER OF ITERATION                                  */
/*            KCLO-FLAGE TO INDIACTE THE PIPE IS CLOSED OR NOT                  */
/*                  IF THE PIPE IS CLOSED SO WE TAKE q from the equation        */
/*                                                                              */
/*                                                                              */
/*==============================================================================*/

#include<stdio.h>
#include<math.h>
#include<stdlib.h>
#define MAXT   20
#define MAX_SIZ  80
//void main(void);

FILE *fili,*filo;

int  gauss (double a[][MAX_SIZ],double  b[],int N,double *ptr_det);

void main(void)

{

double AA[MAX_SIZ],BB[MAX_SIZ],CC[MAX_SIZ],EE[MAX_SIZ],FF[MAX_SIZ];
double DD[MAX_SIZ],GG[MAX_SIZ];
int NCODE=1;//int NRST=0;
int JC[MAX_SIZ],MPL[MAX_SIZ];
int i,j,k,NP,MBEG,jj,s,d,l,J8,J9,L,NXX,n,moh;
double C[MAX_SIZ],R[MAX_SIZ];
int LN,NN,NNS,II,N1,N2,MJJ,J,NJUNC,NEXT;
int JP[MAX_SIZ],NEX[MAX_SIZ],NIX[MAX_SIZ],KC[MAX_SIZ],JX[MAX_SIZ],IX[MAX_SIZ];
int IP[MAX_SIZ][13],JF[MAX_SIZ],JIJ[MAX_SIZ],JA[MAX_SIZ],JB[MAX_SIZ];
int JPIP[MAX_SIZ],JJUN[MAX_SIZ],M[MAX_SIZ],KCLO[MAX_SIZ],NA[10],NB[10];
double Q[MAX_SIZ],D[MAX_SIZ],S[MAX_SIZ],V[MAX_SIZ];
double W9[MAX_SIZ],AM[MAX_SIZ],BM[MAX_SIZ],B[MAX_SIZ],YY[MAX_SIZ],ENGY[MAX_SIZ];
double H1,G1,EP,T1,T2,QI,PO3,PO4,PO5,PO6,PO7;
double a[MAX_SIZ][MAX_SIZ],b[MAX_SIZ],det;
int return_val,N;
int KTEP,KN,NTEP,KK,KJ,NEL;
double A3,P;//double UU=0;double A2=.02517;
//double HDIFF,REY,FFAC,PW;
char fnam1[20],fnam2[20];

/*==============================================================================*/

printf("Give: input file name\r\n");
scanf("%s",fnam1);
if((fili=fopen(fnam1,"r+"))==NULL){
printf("Cannot open input f.%s",fnam1);
exit(0);
}
printf("Give output fule name\r\n");
scanf("%s",fnam2);
if((filo=fopen(fnam2,"w+"))==NULL){
   printf("Cannot open output f.%s",fnam2);
   exit(0);
   }
```

```
/*============================================================================*/

fscanf(fili,"%d %d %d %d %d %d %lf %lf",KN,KK,KTEP,NTEP,KJ,NEL,A3,P);
 for(j=1;j<=KK;++j){
     fscanf(fili,"%d %d %lf %lf %lf %lf %lf %lf",JA[j],JB[j],Q[j],D[j],S[j],V[j],R[
j],C[j],ENGY[j]);
    }
 for(j=1;j<=KTEP;++j){
     fscanf(fili,"%d %d",NA[j],NB[j]);
    }
 for(j=1;j<=KN;++j){
     fscanf(fili,"%d %d %d",IX[j],JIJ[j],B[j]);
    }
 for(i=1;i<=KK+1;++i){
     fscanf(fili,"%d",M[i]);
    }
 for(j=1;j<=NEL;++j){
     fscanf(fili,"%d",MPL[j]);
    }
  for(i=1;i<=NTEP;++i){
        fscanf(fili,"%d %d",JPIP[i],JJUN[i]);
      }
      N=KK;

for(i=1;i<=20;++i){
    KC[i]=0;
    KCLO[i]=0;
    }
for(j=1;j<=KTEP;++j){
    s=NA[j];
    d=NB[j];
    B[j+KN]=ENGY[s]-ENGY[d];
   }

/*----------------------------------------------------------------------------*/

for(i=1;i<=MAXT;++i){
    NNS=0;
  for(k=1;k<=KN;++k){
       W9[k]=0;
       JP[k]=NNS+1;
       NP=M[k+1]-1;
       MBEG=M[k];
   for(jj=MBEG;jj<=NP;++jj){//-------------->>jj
          LN=MPL[jj];
          NN=abs(LN);
          if(KCLO[NN]!=1){
          NNS=NNS+1;
          JX[NNS]=NN;
          AM[NNS]=1;
          if(LN < 0){
             AM[NNS]=-AM[NNS];
             }
           }
        a[k][NN]=AM[NNS];
        }//------------------<<jj
      }
/*----------------------------------------------------------------------------*/
      if(KN==KK){
         goto Came;
         }
```

```
for(k=KJ;k<=KK;++k){//--------------k
    W9[k]=0;
    L=k-KJ+1;
    JP[k]=NNS+1;
    NP=M[k+1]-1;
    MBEG=M[k];
for(jj=MBEG;jj<=NP;++jj){//-----------E
   LN=MPL[jj];
   NN=abs(LN);
   NNS=NNS+1;
   JX[NNS]=NN;
 if(KCLO[NN]==1){
   goto Ali;
  }
 if(Q[NN]<0 && KC[NN]>0){
    Q[NN]=3.14 * D[NN] * D[NN];
  }
   QI=fabs(Q[NN]);
   T1=0;
   T2=0;
 if(QI==0){
   goto Ali;
  }
 if(KC[NN]==0){
   goto Rama;
  }
   II=KC[NN];
 if(BB[II]==0){
   goto Omar;
  }
   JC[NN]=1;
 if(QI < GG[II]){
   goto Rajb;
  }
   T1=FF[II]+EE[II] * QI;
   T2=EE[II];
   goto Rama;
   Rajb:
   ;
   PO3=pow(QI,BB[II]);
   T1=DD[II] - CC[II] * PO3;
   PO4=pow(QI,(BB[II]-1.));
   T2=BB[II] * CC[II] * PO4;
   goto Rama;
   Omar:
   ;
   T1=AA[II] * A3 / QI;
   T2=AA[II] * A3/(QI * QI);
   Rama:
   ;
   PO5=pow(QI,P);
   PO6=pow(QI,(2.0-T1));
   H1=(S[NN] * PO5)+(V[NN] * PO6);
   PO7=pow(QI,(P-1.0));
   G1=(P*S[NN]* PO7)+( 2 * V[NN]* QI)+T2;
   AM[NNS]=G1*LN/NN;
   EP=((G1 * QI) - H1)* Q[NN]* LN/(QI * NN);
   Ali:
   ;
 if(KCLO[NN]!=0 && QI==0){
    AM[NNS]=1 * LN/NN;
    EP=0;
   }
    a[k][NN]=AM[NNS];
    W9[k]=W9[k]+EP;
   }
 }
/*------------------------------------------------------------------------------*/
 Came:
 ;
 JP[KK+1]=NNS+1;
   if(NCODE==0){
       goto Salam;
      }
for(j=1;j<=NNS;++j){
    IX[j]=JX[j];
    //printf("\nIX[%d]=%d",j,IX[j]);
    BM[j]=AM[j];
    printf("\nBM[%d]=%2.5lf",j,BM[j]);
    printf("\n moha= ");
    scanf("%d",&moh);
  }
for(j=1;j<=KK;++j){
    if(JP[j+1]==JP[j]){
       printf("\nXXX[all pipe connection the junctionJJI[%d]=%d are closed]XXXX",j)
;
      }
  IP[j][1]=JP[j];
  printf("\n IP[%d][1]=%d",j,JP[j]);
 }
  IP[KK+1][1]=NNS+1;

for(j=1;j<=KJ;++j){
   JF[j]=0;
   }

NJUNC=0;
NEXT=NTEP;
/*------------------------------------------------------------------------------*/

for(l=1;l<=NTEP;++l){
    J9=JPIP[l];
    if(KCLO[J9]!=0){
      NEXT=NEXT-1;
      goto Oalid;      //NEXT L
     }
    n=JJUN[l];
    J8=JIJ[n];
    if(JF[J8]==1){
      NEXT=NEXT-1;
      goto Oalid;     //NEXT L
     }else{
     JF[J8]=1;
     }
     NJUNC=NJUNC+1;
     NEX[NJUNC]=J8;

  Oalid:
  ;
 }
/*------------------------------------------------------------------------------*/
 Salim:
 ;
 NXX=0;
 if(NEXT==0){
    goto Ahmed;
   }
for(l=1;l<=NEXT;++l){
    J=NEX[l];
    MBEG=M[J];
    MJJ=M[J+1]-1;
  for(k=MBEG;k<=MJJ;++k){//-------K
     N1=MPL[k];
     N1=abs(N1);
     if(KCLO[N1]!=1){// FULL OUT PUT FOR THE JUNCTION NODE
        N2=JA[N1]+JB[N1]-J;
        if(JA[N1]!=0){
           if(JB[N1]!=0){
           if(JF[N2]!=1){
           JF[N2]=1;
           NJUNC=NJUNC+1;
           NXX=NXX+1;
           NIX[NXX]=N2;
           }
         }
        }
       }
   }//---------------K
}//-------------------L

NEXT=NXX;
if(NEXT!=0){
   for(l=1;l<=NEXT;++l){
       NEX[l]=NIX[l];
       }
    goto Salim;
 }
 Ahmed:
 ;
 if(NJUNC!=KN){
    printf("\nXXXXXXX[no open connection to the system]XXXXXXXXX");
    }

  Salam:
  ;
for(j=1;j<=KK;++j){

    YY[j]=B[j]+W9[j];
    b[j]=YY[j];
    printf("\nYY[%d]=%lf",j,YY[j]);

    }
   printf("\n L=%d",L);

/* call gaussl function*/

return_val=gauss(a,b,KK,&det);

/*pritn rreslut*/

if (return_val == 0){

printf("\n\n \t the sluatio of the  smulation of linear equation  is");
printf("\n the sluation is ");
for(i=1;i<=N;++i){
 printf("\n \t x(%d)=%lf",i,b[i]);
 }
 printf("\n determinat=%lf",&det);
  }else
printf("\n\n \t the matrix is singular " );

for(j=1;j<=N;++j){
      Q[j]=b[j];
      printf("\nQ[%d]=%lf",j,Q[j]);

    }//-------------------------------j
   }
}  /*main*/

/*============================================================================*/
/*                      function to slove linear equation                     */
/*                                                                            */
/*                                                                            */
/*============================================================================*/

int  gauss (double a[][MAX_SIZ],double  b[],int N,double *ptr_det)

{

double temp,mult,trt,tol;
int npivot,i,j,k,l,error_flag;

*ptr_det=1.0;
tol=1e-30;
npivot=1;

for(k=1;k<=N-1;++k){
 for(i=k;i<=N;++i){
    printf("aik[%d][%d]=%lf",i,k,a[i][k] );
    printf("akk[%d][%d]=%lf",k,k,a[k][k] );

    if(fabs(a[i][k])>fabs(a[k][k])){
    ++npivot;
      for(l=1;l<=N;++l){
       temp=a[i][l];
       a[i][l]=a[k][l];
       a[k][l]=temp;

      }
      temp=b[i];
      b[i]=b[k];
      b[k]=temp;
     }                                    /*  if  */
  printf("\n\t  b[%d]=%lf",i,b[i]);
      }                                     /*   i  */

   trt=*ptr_det;
   printf("\n\t atrt=%lf",trt );
      printf("akk[%d][%d]=%lf",k,k,a[k][k] );
     trt =*ptr_det*a[k][k];
if (fabs(trt)<tol){
    error_flag=1;
    return(error_flag);
  }

  for(i=2;i<=N;++i){
    if(i!=k){
      mult=a[i][k]/a[k][k];
      b[i]=b[i]-b[k]*mult;
      printf("\n\t mult=%lf",mult );
   for(j=1;j<=N;++j)
    {
    a[i][j]=a[i][j]-a[k][j]*mult;
   printf("aii[%d][%d]=%lf",i,j,a[i][j] );

    }
   }
  }
}  /*  k  */

for(i=1;i<=N;++i)
  for(j=1;j<=N;++j)
printf("\n\t  af[%d][%d]=%lf",i,j,a[i][j]);
for(i=1;i<=N;++i)
{
printf("\n\t  b[%d]=%lf",i,b[i]);
}
if (npivot %2==1)
*ptr_det=*ptr_det*(-1.0);

b[N]=b[N]/a[N][N];
 printf("\n\t  b[%d]=%lf",N,b[N]);
 printf("\n\t  af[%d][%d]=%lf",N,N,a[N][N]);
for(i=N-1;i>=1;i--){
 for(j=i+1;j<=N;++j)
     b[i]=b[i]-a[i][j]*b[j];
     b[i]=b[i]/a[i][i];
   }
  error_flag=0;
 return(error_flag);
}
```

```
/*==============================================================================*/
/* computer analysis of flow and pressure in pipe networks                      */
/*  function :                                                                  */
/*            Gaussin Elimintation                                              */
/*                                                                              */
/*==============================================================================*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define  eps 1.e-6
#define one 1.
#define onn -1.

FILE *fili,*filo;

int gausel(int n,float **a,float *b,float *dd,float *error);

void main(void){

 int n,i,j;
 float **a,*b,*dd,*error;
 char fnam1[20],fnam2[20];

printf("Give N "); scanf("%d",&n);

 b=(float*)calloc(n,sizeof(float));
 dd=(float*)calloc(n,sizeof(float));
 error=(float*)calloc(n,sizeof(float));

 a=(float**)malloc(n*sizeof(float*)); //allocates pointers to rows

 for(i=0;i<n;i++) a[i]=(float*)malloc(n*sizeof(float));// allocates rows & pointers
 to them

/*==============================================================================*/

 printf("Give input file name\r\n");
 scanf("%s",fnam1);
 if((fili=fopen(fnam1,"r"))==NULL){
 printf("File %s does not exist",fnam1);
 exit(0);
 }
 printf("Give output file name\r\n");
 scanf("%s",fnam2);
 if((fili=fopen(fnam2,"w"))==NULL){
 printf("File %s does not exist",fnam2);
 exit(0);
 }

/*==============================================================================*/

 for(i=0;i<n;i++){
     for(j=0;j<n;j++)
        fscanf(fili,"%f",&a[i][j]);
         fscanf(fili,"%f",&b[i]);
         }

 j=gausel(n,a,b,dd,error);

 printf("\nSolution:\n");for(i=0;i<n;i++)printf("%f ",b[i]);
 printf("\nDeterminant=%f\n",*dd);

// free(b);free(error);for(i=n-1;i>=0;i--) free(a[i]); free(a);
}

/*--------------------------------------------------------------------------*/

int gausel(int n,float **a,float *b,float *det,float *errnor){

int n1,i,j,k,i1,*l;

float **ao,*bo,**xo,fac;

*det=one; n1=n-1;

l=(int *)malloc((unsigned)n*sizeof(int));

bo=(float *)malloc((unsigned)n*sizeof(float));

ao=(float **)malloc((unsigned)n*sizeof(float *));

 for(i=0;i<n;i++) ao[i]=(float *)malloc((unsigned)n*sizeof(float));

xo=(float **)malloc((unsigned)n*sizeof(float*));

 for(i=0;i<n;i++) xo[i]=(float *)malloc((unsigned)n*sizeof(float));

/* Copies original a & b into ao & bo for later use in determining error. */

 for(i=0;i<n;i++){
      bo[i]=b[i];
     for(j=0;j<n;j++)
       ao[i][j]=a[i][j];
      }

/* pivots on largest row & keeps track of row no in l[k]; */

 for(k=0;k<n1;k++){
     fac=fabs(a[k][k]);
      l[k]=k;

  for(i=k+1;i<n;i++)
      if(fabs(a[i][k])>fac){
         fac=fabs(a[i][k]);
         l[k]=i;
       }

  if(fac<eps){printf("Matrix is singular"); return 1;}

  if(l[k] != k){
     *det *= onn;
      for(j=k;j<n;j++){
      fac=a[k][j];
      a[k][j]=a[l[k]][j];
      a[l[k]][j]=fac;
      }
    }

/* Gaussian elimination (only a because b must be done also for iterative cor.) */

 for(i=k+1;i<n;i++){
     fac=a[i][k]/a[k][k];
     a[i][k]=fac;
     for(j=k+1;j<n;j++)
     a[i][j]-=fac*a[k][j];
     }
    }

/* Computes determinant det */

 for(i=0;i<n;i++) *det *= a[i][i];

 for(i1=0;i1<2;i1++){ /* now adjust b for orig. elim & iter. correction */

  for(k=0;k<n1;k++){
     if(l[k] != k){
        fac=b[k];
        b[k]=b[l[k]];
        b[l[k]]=fac;
      }
     for(i=k+1;i<n;i++)
        b[i]-=a[i][k]*b[k];
      }

/* Back substitution */

 xo[n1][i1]=b[n1]/a[n1][n1];

 for(i=n1-1;i>=0;i--){
     fac=b[i];
    for(j=i+1;j<n;j++)
     fac-=a[i][j]*xo[j][i1];
      xo[i][i1]=fac/a[i][i];
      }

/* Computes residual vector  {r}={b}-[a]{x} */

   for(i=0;i<n;i++){//------ii
       fac=bo[i];
     for(j=0;j<n;j++)
          fac-=ao[i][j]*xo[j][i1];
          b[i]=fac;
          bo[i]=fac;
       } //-------ii
    }

 for (i=0;i<n;i++){b[i]=xo[i][0]+xo[i][1];

     errnor[i]=fabs(xo[i][1])/(fabs(b[i])+eps); }

 free(l);free(bo);for(i=n-1;i>=0;i--) free(xo[i]);free(xo);

  for(i=n-1;i>=0;i--)free(ao[i]);free(ao);
  return 0;

}
```

```
/*============================================================================*/
/*      COMPUTER ANALYSIS OF FLOW AND PRESSURE IN PIPE NETWORKS               */
/* FUNCTION 6:                                                                */
/*                                                                            */
/*         FLOW: FLOW RATE IN EACH PIPE                                       */
/*         HL: HEAD LOSSES IN THE PIPE                                        */
/*         PUMP: PUMP HEAD                                                    */
/*         ZINOR: MINOR LOSSES IN THE PIPE                                    */
/*         VEL: VELOCITY                                                      */
/*                                                                            */
/*============================================================================*/

#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define PE   1.852
#define MAX_SIZ  20

FILE *fili,*filo;

void main(void)
{

double AA[MAX_SIZ],BB[MAX_SIZ],CC[MAX_SIZ],EE[MAX_SIZ],FF[MAX_SIZ];
double DD[MAX_SIZ],GG[MAX_SIZ];
int KPI[MAX_SIZ],JA[MAX_SIZ],JB[MAX_SIZ],JJI[MAX_SIZ],KCLO[MAX_SIZ],KC[MAX_SIZ];
int j,NN,II,LL,IPRINT,JAAAA,JBBBB,M;
int NPO[MAX_SIZ],JC[MAX_SIZ];
double Q[MAX_SIZ],V[MAX_SIZ],D[MAX_SIZ],R[MAX_SIZ],S[MAX_SIZ],YY[MAX_SIZ];
double FLOW,HL,PUMP,ZINOR,VEL,XHL,A3,CQ,P,QI;
int IOUT,NPOUT,KK,KN;
char fnam1[20],fnam2[20];

/*============================================================================*/
printf("Give file name for input\r\n");
        scanf("%s",fnam1);
      if((fili=fopen(fnam1,"r+"))==NULL){
          printf("Cannot open input f. %s",fnam1);
          exit(0);
      }
printf("Give file name for output\r\n");
      scanf("%s",fnam2);
  if((filo=fopen(fnam2,"a+"))==NULL) {
      printf("Cannot open output f. %s",fnam2);
      exit(0);
  }

/*============================================================================*/

/*----------------------------------------------------------------------------*/
/* IOUT==1 THAT IS MEAN NOT FULL OUT PUT THERE ARE SOME PIPE SLEECTED         */
/* NPOUT =2 HOW MANY PIPE HASE BEEN SLEECTED                                  */
/* IF IOUT=0 FULL OUT PUT                                                     */
/*----------------------------------------------------------------------------*/

fscanf(fili,"%d %d %d %lf %lf %lf",KK,KN,IOUT,NPOUT,A3,CQ,P);
for(j=1;j<=KK;++j){
    fscanf(fili,"%d %d %d %d %d %d",JA[j],JB[j],KC[j],KCLO[j],KPI[j],JC[j]);
    }
    for(j=1;j<=KK;++j){
       fscanf(fili,"%lf %lf %lf %lf %lf",Q[j],S[j],V[j],R[j],D[j]);
       }
```

```
            for(j=1;j<=KN;++j){
               fscanf(fili,"%d %d",JJI[j],NPO[j]);
               }
        for(NN=1;NN<=KK;++NN){
            LL=KPI[NN];
            IPRINT=1;
          if(IOUT!=0){//-------------->>9520
            IPRINT=0;
          if(NPOUT!=0){//>>>>>>>>>>>1
         for(j=1;j<=NPOUT;++j){
            if(NPO[j]==LL){
              IPRINT=1;
             }
            }
           }//<<<<<<<<<<<<<<<<<<1
         }//---------------------<<<<9520
         if(JA[NN]!=0){
            M=JA[NN];
            JAAAA=JJI[M];
            }else{
            JAAAA=0;
         }
         if(JB[NN]!=0){
           M=JB[NN];
           JBBBB=JJI[M];
            }else{
            JBBBB=0;
         }
         QI=fabs(Q[NN]);
         HL=(S[NN] * pow(QI,P-1.0)) * Q[NN];
         ZINOR=V[NN] * Q[NN] * fabs(Q[NN]);
         PUMP=0;
         if(KCLO[NN]==1){//-->>.20
            goto MO;
            }
         if(KC[NN]==0){//--->>20
            goto MO;
            }
          IPRINT=1;
          II=KC[NN];
         if(BB[II]!=0){//-->>21
            goto SA;
            }
         if(Q[NN]==0){//-->>20
            goto MO;
            }
          PUMP=AA[II] * A3/Q[NN];
          goto MO;
          //printf("\n the number of the pip LL=%d",LL);//-->>20
          SA:
          ;
         if(Q[NN] >= GG[II]){///21<<--->>23
          PUMP=FF[II]+EE[II] * Q[NN];
          printf("\n the number of the pipe LL=%d",LL);//-->>20
          goto MO;
          }
          PUMP=DD[II]-CC[II] * pow(Q[NN],BB[II]);//----23
         if(Q[NN] <= AA[II]){//------>>>20
          printf("\n the number of the pipe LL=%d",LL);
          }
          MO:
          ;
```

```
  YY[NN]=HL - PUMP +ZINOR;//---<<20
 if(KCLO[NN]==1){
  YY[NN]=HL;
  }
  FLOW=Q[NN]*CQ;
  VEL= (Q[NN]* 4)/(3.14159 * pow(D[NN],2.0));
  XHL=HL * 1000 /R[NN];
 if(KCLO[NN] ==1 && JC[NN]!=1){
 printf("\n line (%d) is closed ",LL);
 }
 if(KCLO[NN]==1 && JC[NN]==1){
     printf("\n the check valve in line %d  is closed",LL);
   }
 if(KCLO[NN]==1){
    Q[NN]=0;
    continue;
    }
 if(IPRINT ==0 ){continue;}
 printf("\n LL=%d  JAAAA=%d  JBBBB=%d ",LL,JAAAA,JBBBB);
 printf("\n FLOW=%lf HL=%lf PUMP=%lf",FLOW,HL,PUMP);
 printf("\nZINOR=%lf VEL=%lf XHL=%lf",ZINOR,VEL,XHL);
  }
}
```

```
/*=========================================================================*/
/*computer analysis of flow and pressure in pipe networks                  */
/*                                                                         */
/*function :                                                               */
/*          Netown method to solve DARCY Equation                          */
/*=========================================================================*/
#include <math.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#define sqr(x) x*x

void main(void){

 float x[6],cu[2]={.1586599,.2874489},cv[2]={1.1651147,2.110871},visc,g,g2;
 float uc1,uc2,uc3,dx,hdl,adl,f,f1,dif,xx;
 int ii,i,iv,i6,iu,m;
 char *v[]={"hf","D","e","L","Q","V"};

 L1:
 printf("Give 1=ES or 2=SI (or 0 0=STOP) & Visc ");
 scanf("%d %f",&ii,&visc);
 if(ii<1) exit(0);
 if(ii==1) g=32.2; else g=9.81; g2=2.*g;
 printf("\r\nGive No. of Unknown\r\n");
 for(i=0;i<6;i++)
  printf("%d %s\r\n",i,v[i]);
  scanf("%d",&iu);
 if(iu<4) {
    printf("\r\nGive 1 if Q will be given or 2 if V is known ");
    scanf("%d",&iv);
    } else iv=iu-3; ii--;
   uc1=cu[ii]; uc2=cv[ii]*visc;  uc3=.78539816*uc1;
   printf("\r\nGive value to knowns\r\n");
   i=0;
   if(iu>3)i6=4;
   else i6=5;
 do {if(i==iu)i++;
 if(i==6-iv)i++;
 printf("%s = ",v[i]);
 scanf("%f",&x[i]);
 i++;
 } while (i<i6);
// Initializes unknown
 switch(iu){
  case 0: if(iv==1) x[5]=x[4]/(.7853982*sqr(x[1]));
    x[0]=.02*x[3]*sqr(x[5])/g2;  break;
  case 1: if(iv==1) x[1]=pow(.02*x[3]*sqr(x[4])/(.6185*g2*x[0]),.2);
    else x[1]=.02*x[3]*sqr(x[5])/(g2*x[0]);  break;
  case 2: x[2]=.0006; break;
  case 3: if(iv==1) x[3]=.6185*g2*x[0]*pow(x[1],5.)/(.02*sqr(x[4]));
    else x[3]=g2*x[0]*x[1]/(.02*sqr(x[5])); break;
  case 4: x[4]=sqrt(.6185*g2*x[0]*pow(x[1],5.)/(.02*x[3])); break;
  case 5: x[5]=sqrt(g2*x[0]*x[1]/(.02*x[3]));    }
// Newton Method
 m=0;
 do {
    xx=x[iu];
    x[iu]*=1.005;
    dx=x[iu]-xx;
     L2:
    hdl=sqrt(x[3]/x[0]);
    adl=2.*log10(x[2]/x[1]+uc2*hdl/pow(x[1],1.5))-1.14;
  if(iv==1) f=uc1*hdl*x[4]/pow(x[1],2.5)+adl;
    else f=uc3*hdl*x[5]/sqrt(x[1])+adl;
     m++;
    if(m%2 !=0){
    x[iu]=xx;f1=f;
     goto L2;
     }
    dif=dx*f/(f1-f);
    if(fabs(dif)>.8*x[iu])
      dif*=.5;
      x[iu]=xx-dif;
 } while ((fabs(dif)>.00005) && (m<30));

 if(m>29) printf("Failed to converge %f\r\n",dif);
 if(iv==1) x[5]=x[4]/(.78539816*sqr(x[1]));
  else x[4]=.78539816*sqr(x[1])*x[5];
 for(i=0;i<6;i++)printf("%s = %f\r\n",v[i],x[i]);

 printf("f = %f\r\n",x[0]*x[1]*g2/(x[3]*sqr(x[5])) );
 goto L1;
}
```

```
/*============================================================================*/
/*                COMPUTER ANALYSIS OF FLOW IN PIPE NETWORKS                   */
/*  FUNCTION 7:                                                                */
/*                                                                             */
/*                       NQ=1  FLOW UNIT IN GPM                                */
/*NSD=0  SUPPRESS IN PUT DATA SUMMARY,IF IT NOT EQUAL TO ZERO                  */
/*                                                                             */
/*============================================================================*/

#include<stdio.h>
#include<math.h>
#include<stdlib.h>

#define MAX_SIZ   20

FILE *fili,*filo;

void main(void)
{

int JA[MAX_SIZ],JB[MAX_SIZ],JIJ[MAX_SIZ],JJUN[MAX_SIZ],MPL[30],JD[MAX_SIZ];
int M[MAX_SIZ],NEX[MAX_SIZ],NIX[MAX_SIZ],JPIP[MAX_SIZ],JJI[MAX_SIZ],NJO[MAX_SIZ];
int KCLO[MAX_SIZ],KPI[MAX_SIZ];
double E[MAX_SIZ],B[MAX_SIZ],W9[MAX_SIZ],Y[MAX_SIZ],YY[MAX_SIZ],ENGY[MAX_SIZ];
double Q[MAX_SIZ];
int KK,KN,KJ,NEL,IPRINT,NXX,NEXT,NTEP,N1,N2,N3,MBEG,MJJ,m,JMAX,JMIN,II;
double CQ,QEXTT,QEXT,PRES,SW,PMAX,PMIN,QEX,QIN,QOUT;
int i,j,k,l,J8,J9,J,I,IOUT,NJOUT,NQ,NMOM,JDIFF,L,TT1;

char fnam1[21],fnam2[21];

/*============================================================================*/
printf("Give file name for input\r\n");
scanf("%s",fnam1);
if((fili=fopen(fnam1,"r+"))==NULL){
printf("Cannot open input f. %s",fnam1);
exit(0);
}
printf("Give file name for output\r\n");
   scanf("%s",fnam2);
if((filo=fopen(fnam2,"a+"))==NULL){
   printf("Cannot open output f. %s",fnam2);
   exit(0);
  }

/*============================================================================*/
fscanf(fili,"%d %d %d %d %d %d %d %d %d",KK,KN,KJ,NEL,NTEP,IOUT,NJOUT,NQ,NMOM);
fscanf(fili,"%lf %lf",CQ,SW);

for(j=1;j<=KK+1;++j){
    fscanf(fili,"%d",M[j]);
    }
    for(j=1;j<=NEL;++j){
       fscanf(fili,"%d",MPL[j]);
       }
for(j=1;j<=KK;++j){
    fscanf(fili,"%lf %lf %lf",Q[j],ENGY[j],YY[j]);
    }
for(j=1;j<=KK;++j){
    fscanf(fili,"%d %d %d %d",JA[j],JB[j],KPI[j],KCLO[j]);
    }
    for(i=1;i<=NTEP;++i){
       fscanf(fili,"%d %d",JJUN[j],JPIP[j]);
       }
       for(j=1;j<=KN;++j){
       fscanf(fili,"%d %d %d %d",JIJ[j],JJI[j],E[j],B[j]);
       }


/*===========================================================================*/
/* IOUT== ZERO FOR FULL OUT PUT                                              */
/* IOUT!=0 FOR SECLEECTED JUNCTION FOR THE RESLUT                            */
/* NJOUT!=0 THERE ARE JUNCTION SLECTEED TO SHOW AS RESLUT                    */
/* NJO[]==1 HER THE JUNCTION ONE IS SLECTEED IN THE RESLUT                   */
/*YY[]= THE HEAD LOSS IN THE PIPE =HL-PUMP HEAD+MIONR LOSSES                 */
/* THIS PROGRAMMS TO COMPTE THE QRADE LINE AND THE PRESSURE AT EACH JUNCTION */
/* AND TO SHOW THE MAXMMUMPRESSURE AND THE MINMMUM PRESSURE IN WHERE         */
/* IN THE NET WORKS.                                                         */
/*SUMMARY OF INFLOW AND OUT FLOW FROM FIXED GRADE NODE.                      */
/* THE NET FLOW INTO THE SYSTEM AND THE NET OUT OF THE SYSTEM                */
/*===========================================================================*/

for(i=1;i<=KJ;++i){
   Y[i]=0.0;
   }
   NEXT=NTEP;
for(j=1;j<=NEXT;++j){
    m=JJUN[j];
    J8=JIJ[m];
    NEX[j]=J8;
    J9=JPIP[j];
    Y[J8]=ENGY[J9]+YY[J9];
   if(JA[J9]==0){
     Y[J8]=ENGY[J9]-YY[J9];
     }
   }
     MO:
     ;
     NXX=0;
 for(i=1;i<=NEXT;++i){
     J=NEX[i];
     MBEG=M[J];
     MJJ=M[J+1]-1;
 for(k=MBEG;k<=MJJ;++k){
     N1=MPL[k];
     N1=abs(N1);
     N2=JA[N1];
     N3=JB[N1];
  if(N2==J){
    if(N3==0){
    continue;
   }
    if(Y[N3]!=0){
    continue;
   }
   Y[N3]=Y[N2]-YY[N1];
   NXX=NXX+1;
   NIX[NXX]=N3;
  }
  if(N2==0){
    continue;
    }
  if(Y[N2]!=0){
     continue;
     }
    Y[N2]=Y[N3]+YY[N1];
    NXX=NXX+1;
    NIX[NXX]=N2;
   }
 }
 NEXT=NXX;
  if(NEXT!=0){
    for(i=1;i<=NEXT;++i){
    NEX[i]=NIX[i];
    }
   goto MO;
 }
 for(j=1;j<=KN;++j){
    I=JJI[j];
    IPRINT=1;
    if(IOUT!=0){
      IPRINT=0;
     if(NJOUT!=0){
      for(l=1;l<=NJOUT;++l){//-----L
         if(NJO[l]==I){
           IPRINT=1;
        }
     }//----------L
   }
  }
   PRES= (Y[j] - E[j]) * SW * 62.4/144.0;
   if(NQ==3){
   PRES=(Y[j] - E[j]) * SW * 9.807;
   }
   W9[j]=PRES;
   if(E[j]==0){
   W9[j]=0;
   }
   QEXT=- B[j] * CQ;
  if(B[j]==0){
    QEXT=0;
    }
   QEXT=QEXTT+QEXT;//the net system demand
  if(IPRINT==0){//------->next j
     continue;
     }
  if(E[j]==0){
     printf("\n I=%d  QEXT=%lf  %lf  %lf",I,QEXT,Y[j],W9[j]);
     }
  if(E[j]!=0){
    printf("\n I=%d  QEXT=%lf  %lf  PRES=%lf",I,QEXT,Y[j],PRES);
    }
  }
if(NMOM!=0){
  printf("\n Maxmmum pressure at the junction");
  for(i=1;i<=NMOM;++i){
      PMAX=-100000.0;
      PMIN=1000.0;
   for(j=1;j<=KN;++j){//-------KN
      if(W9[j]!=0){
       if(W9[j]>PMAX){
          JMAX=j;
          PMAX=W9[j];
          }
       if(W9[j]<PMIN){
          JMIN=j;
          PMIN=W9[j];
          }
         }
      }//<<<<<<<<<<<<<<<<KN
   II=JJI[JMAX];
   JD[i]=JMIN;
   W9[KN+i]=PMIN;
   QEXT=-B[JMAX] * CQ;
   printf("\n %d   %lf   %lf    %lf    %lf",II,QEXT,Y[JMAX],E[JMAX],PMAX);
   W9[JMAX]=0.0;
   W9[JMIN]=0.0;
   }
   printf("\n Minummum pressure for the junction");
   for(i=1;i<=NMOM;++i){
     JMIN=JD[i];
     PMIN=W9[KN+i];
     QEXT=-B[JMIN] * CQ;
     II=JJI[JMIN];
     printf("\n %d  %lf  %lf  %lf  %lf",II,QEXT,Y[JMIN],E[JMIN],PMIN);
     printf("\n The net system demand QEXTT=%lf",QEXTT);
     }
   }
  printf("\n Summary of the inflow(+)and out flow(-)from fixed grad node");
  printf("\n  pipe number    flow rate");

/*--------------------------------------------------------------------------------*/

for(j=1;j<=KK;++j){//-------------------->>>>>KK
    JDIFF=JA[j]-JB[j];
    if((JA[j]+JB[j])<=abs(JDIFF)){
       if(KCLO[j]!=1){
         if(W9[j]!=9999.0){
           TT1=(JA[j]+JB[j])/(JA[j]-JB[j]);
           QEX=-Q[j] * TT1 * CQ;
           if(QEX>0.0){
              QIN=QIN+QEX;
              }
              if(QEX<0.0){
                QOUT=QOUT+QEX;
                }
             }
          }
       }
  L=KPI[j];
  printf("\n   L=%d    QEX=%lf ",L,QEX);
}//-------------------------------------->>>>>KK
 printf("\n The net flow into the system from fixed grad node QIN=%lf",QIN);
 printf("\n The net flow out of the system into fixed grad node QOUT=%lf",QOUT);
}
```

```
/*=========================================================================*/
/*           computer analysis of flow and pressure in piep networks         */
/*                                                                           */
/*                        file-name (oalidin-oalido)                         */
/*=========================================================================*/
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <conio.h>

FILE *fili,*filo,*fil1;
const k1=40,k2=100,k3=5,k4=8;
int nunk,np,nj,npump,nres,nj2,njp2,njp,nvar,jn[k2],nn[k1+1],ires[k4],ipump[k3],l1[k
1],l2[k1];
float l[k1],x[k2],hl[k1],ap[k3],bp[k3],cp[k3],d[k2][k2],sf1=8.,vis7,rg2,hwc;

void solveq(int n,float a[][k2],float b[],int itype,float &dd,int indx[]);
/*=========================================================================*/
int rline(int *list,int in55){

int i,jj,ij,j,ii,n1,nobl;
n1=0;//input from the user
char line[80],rln[3],ch;
 if(in55) gets(line);
    else fgets(line,80,fil1);
for(i=0,ii=0,jj=0,ij=0,nobl=0;i<80;i++){
   ch=line[i];
  switch(ch) {
             case',': case' ': case '/':case NULL:{
 if(nobl && ((ch==' ')||(ch==NULL))){
    nobl++;
    if((nobl>10)||(ch==NULL)) return ii;
    ij=i+1;
  }
  else {
       for(j=0;j<(i-ij);j++)rln[j]=line[ij+j];
      if(jj){
          for(j=n1;j<=atoi(rln);j++) list[ii++]=j; jj=0;
          } else list[ii++]=atoi(rln);
      if((ch=='/')||(ch==NULL)) return ii;ij=i+1;
      if(ch==' ') nobl=1;
   }
   break;
  }
 case'-':{
          for(j=0;j<(i-ij);j++) rln[j]=line[ij+j];
          n1=atoi(rln);
          jj=1;
          ij=i+1;
   break;
  }
 default:nobl=0;
  }
 }
 return ii;
} // End of function rline

/*=========================================================================*/

void fun(float *f) {

 int i,j,ii,id,iq;
```

```c
 float qa,re,fr,sf;
 sf=1;

 for(i=0;i<nj;i++){
 f[i]=x[i+nj];
   for(j=nn[i];j<nn[i+1];j++){
   ii=abs(jn[j]);
   f[i]+=(float)(jn[j]/ii)*x[ii-1+nj2];
   }
 }
 for(i=0;i<np;i++){///np
  id=i+njp;
  iq=i+nj2;
  qa=fabs(x[iq]);
  if(x[i+njp2]>20.) hl[i]=hwc*l[i]*x[iq]*pow(qa,.85185185)/(pow(x[i+njp2],1.8518581
5)*pow(x[id],4.87037));
   else {
        re=qa/(vis7*x[id]);
    if(re<21.8) fr=.4;
       else {if(re<286.)fr=8.715223/re;
        else do{
         sf1=sf;
         sf=1.14-2.*log10(x[i+njp2]/x[id]+sf1/re);
     } while(fabs(sf-sf1)>5.e-6);
     }
  fr=1./sf/sf;
   hl[i]=rg2*fr*l[i]*x[iq]*qa/pow(x[id],5.);
    }///else
 if(l1[i]<0.) f[i+nj]=x[abs(l1[i])-1+nvar]-hl[i]-x[l2[i]-1];
  else f[i+nj]=x[l1[i]-1]-hl[i]-x[l2[i]-1];
   }/////np
 for(i=0;i<npump;i++){
 ii=ipump[i]-1;f[ii+nj]+=(ap[i]*x[ii+nj2]+bp[i])*x[ii+nj2]+cp[i];
 }
} // End of function fun

/*=============================================================================*/

void main(void) {

int njpp,m,mm,ii,nct,in2,in5,in4,i,j,nnj,nnj1,muk[6],list[k1],indx[k2],ipuk[k2];
char *cuk[]={"HGLs at nodes ","Nodal demands ","Pipe flowrates","Pipe diameters",
             "Pipe roughnesses","Reser. ws-elev"};
char fnam1[20],fnam2[20],fnam3[20];
float g=32.2,visc=1.317e-5,conv=.43333333;
float q1,q2,q3,h1,h2,h3,xx,sum,dd,dx,elev[k1],f[k2],f1[k2];
printf("Give accel. of gravity & viscosity\r\n");
scanf("%f %f",&g,&visc);
vis7=7.34347283*visc;
printf("If network input from kybd give(1)(file=0);unknw input kybd 1;& output to m
onitor 1 (file=0)\r\n");
scanf("%d %d %d",&in2,&in5,&in4);

if(!in2) {
   printf("Give: input file name\r\n");
   scanf("%s",fnam1);
if((fili=fopen(fnam1,"r"))==NULL){
   printf("File %s does not exist",fnam1);
   exit(0);
 }
}
if(!in5) {
   printf("Give input of unkn file\r\n");
   scanf("%s",fnam3);
if((fil1=fopen(fnam3,"w"))==NULL) {
   printf("Cannot open output f. %s",fnam3);
   exit(0);
  }
}
if(!in4) {
   printf("Give file name for output\r\n");
   scanf("%s",fnam2);
if((filo=fopen(fnam2,"a+"))==NULL) {
   printf("Cannot open output f. %s",fnam2);
   exit(0);
  }
}
if(g>20.) hwc=4.727328;
   else {
         conv=9.806;hwc=10.67417;
     }

/*============================================================================*/
/*        Reads:                                                              */
/*              No-pipe-junction-reservoir-pump                               */
/*                                                                            */
/*============================================================================*/

if(in2) scanf("%d %d %d %d",&np,&nj,&nres,&npump);
 else fscanf(fili,"%d %d %d %d",&np,&nj,&nres,&npump);
nj2=2*nj;
 njp=nj2+np;
  njp2=njp+np;
   nvar=njp2+np;
    njpp=nj+1;
     rg2=.81056947/g;
      sf1=8.;
/*============================================================================*/
/* Reads reservoir data:                                                      */
/*                      No-line-elevation                                     */
/*                                                                            */
/*============================================================================*/

for(i=0;i<nres;i++){
if(in2)scanf("%d %f",&ires[i],&x[nvar+i]);
else fscanf(fili,"%d %f",&ires[i],&x[nvar+i]);
}
/*============================================================================*/
/* Reads pump data:                                                           */
/*                 No-line-(heaf-flow) data                                   */
/*                                                                            */
/*============================================================================*/

for(i=0;i<npump;i++){
if(in2) scanf("%d %f %f %f %f %f %f",&ipump[i],&q1,&h1,&q2,&h2,&q3,&h3);
 else fscanf(fili,"%d %f %f %f %f %f %f",&ipump[i],&q1,&h1,&q2,&h2,&q3,&h3);
 h1/=((q1-q2)*(q1-q3));
 h2/=((q2-q1)*(q2-q3));
 h3/=((q3-q1)*(q3-q2));
 ap[i]=h1+h2+h3;
 bp[i]=-(q2+q3)*h1-(q1+q3)*h2-(q1+q2)*h3;
 cp[i]=q2*q3*h1+q1*q3*h2+q1*q2*h3;
 }
```

```
for(i=0;i<njpp;i++) nn[i]=0;

/*=========================================================================*/
/* function to read pipe data:                                             */
/* READS: Node1,Node2,Length,Diameter,roughness, & guess flowrate          */
/*                                                                         */
/*=========================================================================*/

printf("----------------");
for(i=0;i<np;i++){
if(in2) scanf("%d %d %f %f %f %f",&l1[i],&l2[i],&l[i],&x[njp+i],&x[njp2+i],&x[nj2+i
]);
else fscanf(fili,"%d %d %f %f %f %f",&l1[i],&l2[i],&l[i],&x[njp+i],&x[njp2+i],&x[nj
2+i]);
if(l2[i]==0){
   printf(" 0 for reservoir must be 1st given node");
   exit(0);
 }
}
nnj=0;
for(i=0;i<np;i++){
  if(l1[i]){
   nnj1=nn[l1[i]]+1;
 for(j=nnj-1;j>=(nnj1-1);j--) jn[j+1]=jn[j];
    jn[nnj1-1]=i+1;
  for(j=l1[i];j<njpp;j++) nn[j]+=1;
    nnj=nn[njpp-1];
    }
   nnj1=nn[l2[i]]+1;
 for(j=nnj-1;j>=(nnj1-1);j--) jn[j+1]=jn[j];
    jn[nnj1-1]=-(i+1);
 for(j=l2[i];j<njpp;j++) nn[j]+=1;
  nnj=nn[njpp-1];
}

/*=========================================================================*/
/*Reads junction data:                                                     */
/*                    Demands,HGL-ELEV-Elev                                */
/*                                                                         */
/*=========================================================================*/

for(i=0;i<nj;i++) if(in2) scanf("%f %f %f",&x[nj+i],&x[i],&elev[i]);
else fscanf(fili,"%f %f %f",&x[nj+i],&x[i],&elev[i]);
for(i=0;i<nres;i++) l1[ires[i]-1]=-(i+1);  nunk=nj+np; m=0;
do{
if(in5) printf("%4d Unknowns must be given.Give no. of each of following\r\n",nunk)
;
for(i=0;i<6;i++){
    if(in5){
    printf(" %1d . %s ",i+1,cuk[i]);
    scanf("%d",&muk[i]);
 }
 else fscanf(fili,"%d",&muk[i]);
 }
 }while ((muk[0]+muk[1]+muk[2]+muk[3]+muk[4]+muk[5]!=nunk)&&(m++<5));
 if(m>=5){
 printf("Incorrect no. of unknowns\r\n");
  exit(0);
  }
m=0; mm=0;

for(i=0;i<6;i++){
```

```
if(muk[i]){
do {
   if(in5)printf("\r\n %2d %s Numbers= ",muk[i],cuk[i]);
} while (rline(list,in5)!=muk[i]);
for(j=0;j<muk[i];j++) ipuk[m++]=mm+list[j];
}
if(i<2) mm+=nj;
 else mm+=np;
  }
for(i=0;i<np;i++){
for(j=0;j<nunk;j++) if(ipuk[j]==i+1+nj2) goto L28;
 for(j=0;j<nj;j++) if((ipuk[j]==l1[i]) || (ipuk[j]==l2[i])) goto L28;
if(l1[i]<0){xx=x[abs(l1[i])-1+nvar];
 ii=0;
 } else {xx=x[l1[i]-1];ii=l1[i];
 }
if((xx-x[l2[i]-1])*x[i-1+nj2]> -1.e-5) goto L28;

printf("Specified flowrate not consistent with H's in pipe %3d %3d %3d\r\n",i+1,ii,
l2[i]);
printf("H1= %8.2f H2= %8.2f Q= %8.2f must have H1 > H2\r\n",xx,x[l2[i]],x[i+nj2]);
printf("Should I reverse direction of this flow? (1=yes, 0=no\r\n)");
scanf("%d",&ii);

 if(ii) x[i+nj2]=-x[i+nj2];
  else exit(0);
L28: continue;
}
 nct=0;
do {
 sum=0.;
 fun(f1);
 for(j=0;j<nunk;j++){
  ii=ipuk[j]-1;
  if(fabs(x[ii])<1.e-3) dx=.001;
   else dx=.005*x[ii];
 x[ii]+=dx;
   fun(f);
    for(i=0;i<nunk;i++) d[i][j]=(f[i]-f1[i])/dx;
     x[ii]-=dx;
     }

 solveq(nunk,d,f1,1,dd,indx);

 for(i=0;i<nunk;i++){
 sum+=fabs(f1[i]);
 x[ipuk[i]-1]-=f1[i];
 }
  nct++;
  printf("NCT= %d  SUM= %f\r\n",nct,sum);
}while((nct<20) && (sum>.001));
if(npump){
if(in4)printf("Devices caused the following changes in heads\r\n");
 else fprintf(filo,"Devices caused the following changes in heads\n");
 for(i=0;i<npump;i++){
 ii=ipump[i];
 dx=x[ii-1+nj2];
 if(in4)printf("Device %3d in pipe %4d Change in head = %8.3f\r\n",i+1,ii,(ap[i]*dx
+bp[i])*dx+cp[i]);
 else fprintf(filo,"Device %3d in pipe %4d Change in head = %8.3f\n",i+1,ii,(ap[i]*
dx+bp[i])*dx+cp[i]);}}
if(in4) {
```

```
printf("Pipe Data:\r\n");
for(i=0;i<65;i++)printf("-");
printf("\r\n");
  printf("Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss\r\n");
  printf("  No.  #1   #2                        Coef.\r\n");
  for(i=0;i<65;i++) printf("-");printf("\r\n");
  } else {
  fprintf(filo,"Pipe Data:\n");for(i=0;i<65;i++)fprintf(filo,"-");fprintf(filo,"\n"
);
  fprintf(filo,"Pipe Node Node    Length  Diameter Roughness  Flowrate  Headloss\n"
);
  fprintf(filo,"  No.  #1   #2                        Coef.\n");
  for(i=0;i<65;i++) fprintf(filo,"-"); fprintf(filo,"\n");
  }
 for(i=0;i<np;i++){
 ii=l1[i];if(ii<0) ii=0;
  if(in4)printf("%4d %4d %4d %9.1f %9.3f %9.6f %9.3f %9.3f\r\n",i+1,ii,l2[i],l[i],x
[njp+i],x[njp2+i],x[nj2+i],hl[i]); else
 fprintf(filo,"%4d %4d %4d %9.1f %9.3f %9.6f %9.3f %9.3f\n",i+1,ii,l2[i],l[i],x[njp
+i],x[njp2+i],x[nj2+i],hl[i]);}
 if(in4){
 printf("Node Data:\r\n");
 for(i=0;i<54;i++)printf("-");
 printf("\r\n");
  printf(" Node    Demand Elevation      Head  Pressure HGL-elev.\r\n");
  for(i=0;i<54;i++)printf("-");
  printf("\r\n");
  } else {
 fprintf(filo,"Node Data:\n");for(i=0;i<54;i++)fprintf(filo,"-");fprintf(filo,"\n")
;
  fprintf(filo," Node    Demand Elevation      Head  Pressure HGL-elev.\n");
  for(i=0;i<54;i++)fprintf(filo,"-");
  fprintf(filo,"\r\n");
  }
 for(i=0;i<nj;i++){
 xx=x[i]-elev[i];
  if(in4)printf("%5d %9.3f %9.3f %9.3f %9.3f %9.3f\r\n",i+1,x[nj+i],elev[i],xx,conv
*xx,x[i]);
  else fprintf(filo,"%5d %9.3f %9.3f %9.3f %9.3f %9.3f\n",i+1,x[nj+i],elev[i],xx,co
nv*xx,x[i]);
  }
}
#define TINY 1.0e-20;
void dcompos(float a[][k2],int n,int indx[],float d) {
int i,imax,j,k;
float aamax,dum,sum,temp;
float *vv;
vv=(float*)calloc(n,sizeof(float)); d=1.0;
 for(i=0;i<n;i++) {
 aamax=0.0;
  for(j=0;j<n;j++) if ((temp=fabs(a[i][j])) > aamax) aamax=temp;
  if (aamax == 0.0) printf("Singular matrix in routine DCOMPOS\n");vv[i]=1.0/aamax;
  }
 for(j=0;j<n;j++) {
 for(i=0;i<j;i++) {
 sum=a[i][j];
  for(k=0;k<i;k++) sum -= a[i][k]*a[k][j];
  a[i][j]=sum;
  }
 aamax=0.0;
 for(i=j;i<n;i++) {
 sum=a[i][j];
 for(k=0;k<j;k++)sum -= a[i][k]*a[k][j];
  a[i][j]=sum;
  if ( (dum=vv[i]*fabs(sum)) >= aamax) {
  aamax=dum;imax=i;
  }
  }
  if (j != imax) {for(k=0;k<n;k++) {
    dum=a[imax][k];a[imax][k]=a[j][k];
    a[j][k]=dum;
    }
   d = -(d);vv[imax]=vv[j];
   }
   indx[j]=imax;
    if (a[j][j] == 0.0) a[j][j]=TINY;
  if (j != n) {
  dum=1.0/(a[j][j]);
  for(i=j+1;i<n;i++) a[i][j] *= dum;
  }
 }
   free(vv);
}

#undef TINY
void finsol(float a[][k2],int n,int indx[],float b[])
 {int i,ii,ip,j;  float sum;  ii=-1;
for(i=0;i<n;i++) {
ip=indx[i];
sum=b[ip];
b[ip]=b[i];
if (ii != -1){
for(j=ii;j<i;j++) sum -= a[i][j]*b[j];
} else if (sum!=0.) ii=i;
 b[i]=sum;}
for(i=n-1;i>=0;i--) {
sum=b[i];
if(i<(n-1)){
for(j=i+1;j<n;j++) sum -= a[i][j]*b[j];
  }
  b[i]=sum/a[i][i];
 }
}
void solveq(int n,float a[][k2],float b[],int itype,float &dd,int indx[]){
int detrm=0,eqsol=0,invsol=0,i,j; FILE *fil;
if((itype==3)||(itype>4)) detrm=1;
if((itype==1)||(itype==4)||(itype==6)) eqsol=1;
if((itype==2)||(itype>3)) invsol=1;

dcompos(a,n,indx,dd);

if(detrm) for(i=0;i<n;i++) dd*=a[i][i];
if(eqsol) finsol(a,n,indx,b);
if(invsol){
 if(eqsol){
 printf("\nSolution Vector\n");
 for(i=0;i<n;i++){
 printf("%10.3f",b[i]);
 if (!(i%8)) printf("\n");
 }
 }
if((fil=fopen("OMARIN","wt"))==NULL)
   printf("Data file OMARIN cannot be opened\n");
for(j=0;j<n;j++){
 for(i=0;i<n;i++) b[i]=0;
  b[j]=1;
   finsol(a,n,indx,b);
 for(i=0;i<n;i++) fprintf(fil,"%15.7f",b[i]);
  }
  rewind(fil);
for(j=0;j<n;j++) for(i=0;i<n;i++) fscanf(fil,"%15.7f",a[i][j]);
fclose(fil); remove("TEMP.DAT");
}
  return;
}
```