CITY UNIVERSITY

DEPARTMENT OF MECHANICAL ENGINEERING & AERONAUTICS

PRESSURE TRANSIENTS IN A RUPTURED GAS PIPELINE WITH FRICTION AND THERMAL EFFECTS INCLUDED

by

C.H. TILEY

A Dissertation submitted to
City University
for the Degree of
Doctor of Philosophy
in Mechanical Engineering

SEPTEMBER 1989

TEXT CUT OFF IN THE ORIGINAL

LIST OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

This project was conducted with the sponsorship of the Science & Engineering Research Council.

Special thanks go to my supervisor, Professor A.R.D. Thorley, for all his help and advice during the course of this project and for the encouragement he has given me throughout the work.

I would also like to thank other members of staff in the Department of Mechanical Engineering & Aeronautics at City University for their support.

Thanks are also due to Professor V. Price and staff in the Computing Department at City University for their help with the numerical analysis and programming involved in this project.

I would also like to thank David Jones from British Gas (Engineering Research Station) and Brian Rothwell from Foothills Pipelines (Yukon) Ltd., for their help and for supplying me with some experimental data without which I could not have successfully tested my theoretical model.

Finally, a very special thank you to Simon and my parents for all the love and support they have given me throughout the course of this project.

ABSTRACT

A theoretical model has been developed which can simulate a linebreak occurring in a gas pipeline. By assuming one-dimensional homogeneous gas flow and neglecting minor losses and changes in cross-sectional area of the pipe, three simultaneous non-linear partial differential equations were derived from first principles which mathematically model pressure transients in a non-perfect gas. A constant value steady-flow friction factor was used to calculate the frictional losses which was considered to be a reasonable approach since it would not be possible to account for all the variations in friction. The heat transfer into the pipe was accounted for using a constant value Stanton Number approach which again was an acceptable approximation considering the comparatively small effect that heat transfer has on the pressure transients.

The equations were converted to ordinary differential equations using the Method of Characteristics and these were then solved numerically using a Taylor expansion. A novel feature of this project was the incorporation of a reduced grid size in the vicinity of the break allowing closer monitoring of the expansion waves in this area. Also included was a means of modelling flow reversal in the pipe which enabled situations with a non-zero initial flow rate to be simulated.

A computer code solving the mathematical model was written in Fortran 77 for use on a Gould PN9005 mainframe computer. Both tabular and graphical output were produced which could then be compared with available experimental data.

The experimental data that was selected for validation of the theoretical model included shock tube test results and some full size tests. Reasonable agreement was obtained between the theoretical and experimental results and any possible error sources were investigated.

NOMENCLATURE

The symbols used in this text have the following meanings, except where they have been otherwise specifically defined:

Symbol		Units
A	Cross-sectional area of pipe	m^2
a_s	Isentropic wavespeed	m/s
C_p	Specific heat at constant pressure	J/kg K
C_v	Specific heat at constant volume	J/kg K
d	Diameter of pipe	m
e	Specific internal energy	J/kg
f	Darcy friction factor	-
g	Acceleration due to gravity	m/s^2
h	Specific enthalpy	J/kg
i, j	Rectangular coordinates used in explicit finite difference methods	-, -
P	Pressure	Pa
Pr	Prandtl number	-
Q	Heat transfer rate per unit volume	J/m^3s
R	Specific gas constant	J/kg
Re	Reynolds number	-
s	Specific entropy	J/kg K
St	Stanton number	-
T	Temperature of the gas	K
t	Time	s
u	Velocity of the gas	m/s
W	Frictional force per unit length of pipe	N/m
x	Distance along the pipe	m
x'	Thermodynamic quality or dryness fraction	-
z	Gas Compressibility factor	-

Greek Symbols		Units
θ	Angle of inclination of pipe to the horizontal	Rad
ρ	Mean density of the gas	kg/m^3
Ω	Heat flow into the pipe per unit length of pipe and per unit time	J/ms
ω	actual wave propagation speed selective to the pipe	m/s

CHAPTER 1

INTRODUCTION

Ever since the first gas pipelines of the Western World were laid in Philadelphia (1796), Genova (1802), and Fredonia (1821), the demand for gas as an energy source has been growing worldwide. Gas is now considered to be one of the most valuable raw materials due to its high calorific value and so safe and efficient transportation is of prime importance.

Up until the mid 1960's, this country was using town gas which was manufactured in gas-making plants in the towns and then distributed locally at relatively low pressures. Originally, in the 1920's and 1930's, the pipes for this gas distribution were made from cast iron but these were irregular in shape and thickness and by the mid twentieth century steel pipes were being used.

With the advent of natural gas as an energy resource in this country, longer distance pipelines became necessary, and in 1964 the first natural gas steel pipeline in the U.K. was installed, connecting the liquefied natural gas import terminal at Canvey Island with the Midlands. Following the discovery of natural gas in the U.K. sector of the North Sea, long distance deep-water lines were built, for example, the 354 km long, 350 mm diameter pipeline installed in 1973/4 between the northern North Sea (Ekofisk platform) and Teeside.

On these early long distance gas transmission systems, compression of the gas (necessary in order to overcome the

expansion of the gas due to friction) was by reciprocating compressors driven by gas or diesel engines. Although machines of this type could compress gas over a wide range of pressures and flows, there has been an almost complete switch to centrifugal machines driven by gas turbines. These are more suitable for handling large volumes of gas although they will only deliver over a restricted combination of pressure and flow.

Today, gas supplies 20% of the primary energy demand in Britain, most of this coming from the North Sea. The offshore gas is landed and treated by North Sea operating companies at coastal terminals and is then fed into the national grid. The national grid consists of three main sections:-

i) National transmission system - approximately 5000 km of pipes with diameters of up to 1050 mm, operating at pressures up to 70 bar.

ii) Regional distribution systems - approximately 12000 km of smaller diameter pipes (minimum diameter = 100 mm), operating at pressures of 7 bar upwards.

iii) Local service systems - small diameter pipe operating at low pressures.

There have also been large pipeline networks developed in several other countries worldwide. The U.S. natural gas pipeline network is the largest in the world having an overall length of more than 1.5 million km, and following large natural gas discoveries in Siberia and Central Asia, the U.S.S.R. now has the second largest

network. In total, the overall length of natural gas pipelines makes up 70% of all the world's pipelines.

However, there are some problems involved with natural gas transportation by pipeline. Although containing a high proportion of methane, it is also rich in heavier gas components such as butane and pentane. Since these heavier components are liquid at normal temperatures and pressures (between 0 and 20°C and up to approximately 100 bar) the natural gas exists as a two-phase mixture under those conditions as shown in Figure 1.1.

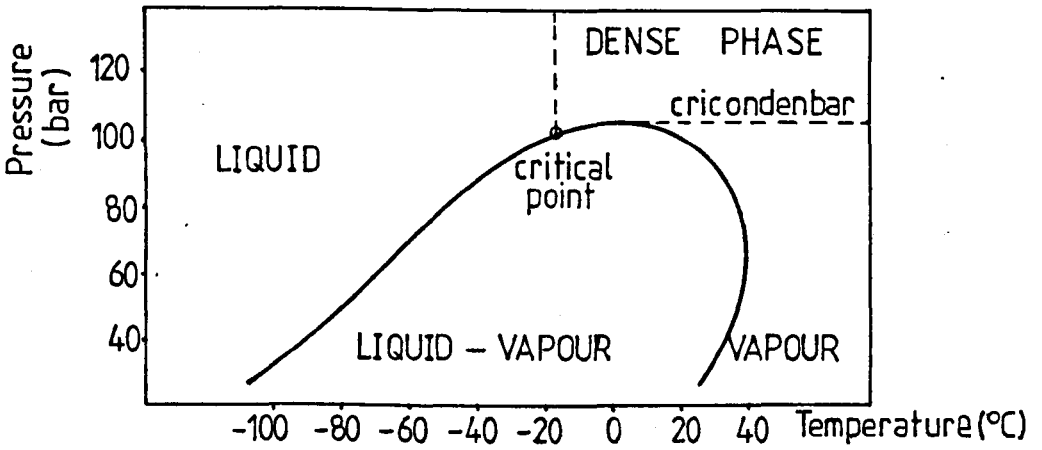


Figure 1.1. Typical Phase Diagram for Natural Gas

Two-phase flow in long distance pipelines is undesirable. The denser liquid phase tends to collect at the bottom of the pipe and since its flow velocity is less than that of the gas phase, the capacity of the pipe is reduced. Also, as the faster gas phase passes over the liquid phase, waves are created which can eventually build up across the entire cross-section of the pipe creating slug flow. This highly non-steady flow, accentuated by any changes in

elevation of the pipe, should be avoided since the slugs of liquid being propelled along the pipe can cause serious damage to pipe fittings and equipment. One way of avoiding this situation is by regularly pigging the line carrying the two-phase mixtures. This, however, incorporates substantial costs in the setting-up and operation of the pigging stations.

A more cost effective solution is to transport the natural gas at very high pressures in a single phase, known as the dense phase. The dense phase is defined by the critical point and the cricondenbar (the highest pressure at which separated liquid and vapour phases can co-exist). This is illustrated in Figure 1.1. It has been found that at these high pressures the gas mixture follows the same equations as single phase gas flow at lower pressure (Oranje, Graaff and Fagerland [1985]).

With these dense phase gases being transported in high pressure, large diameter trunk pipelines, the transient behaviour is of greater significance and economic concern than with the previous gas distribution network. A transient analysis is required to accurately forecast the possibility of the liquid phase appearing as well as improving the overall reliability of the system and optimising the operating conditions.

The transient flow situations that require modelling fall into two main categories, namely, the slow and rapid transients. Slow transients are those fluctuations in pressure and flow caused by

changes in demand, for example, on a daily cycle. A slow transient analysis is mainly concerned with the packing and unpacking of gas in the pipeline. There has been a considerable amount of research directed towards this type of transient and various computer software packages are available which model this type of flow (for example, Bender [1979], Goldfinch [1984], Guy [1967], and Heath and Blunt [1969]).

This is less true of rapid transients which are those caused by a linebreak (pipe rupture), compressor failure, or rapid shut-down or start-up of a system. Although a linebreak in a natural gas pipeline is unlikely to occur through operational error such as over-pressure, the risk of accidental pipe rupture from an external source (for example, by excavation work) cannot be ignored. Figure 1.2 details the distribution of causes of pipeline failures for a group of natural gas pipelines. This data was extracted from a performance analysis of the pipelines in Alberta, Canada, between 1975 and 1983 (Cameron [1984]).

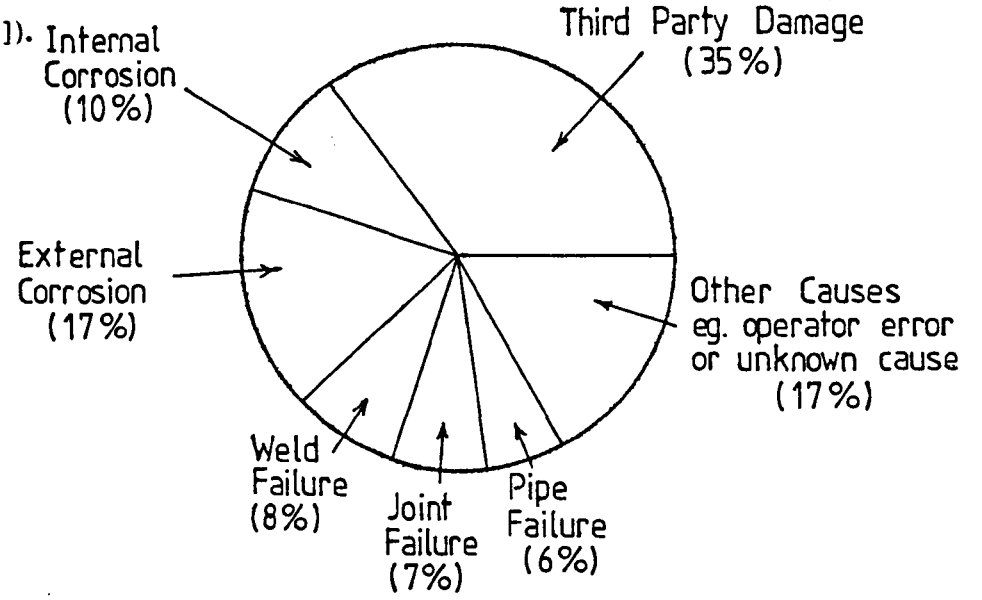


Figure 1.2. Pipe Failure Distribution for Natural Gas Lines

Some authors argue that since the rapid transients caused by an event such as a linebreak are rapidly attenuated in gas pipelines, they are of little significance compared with the slower transients caused by the packing and unpacking of the gas. However, the detection of a linebreak can be important both from an economic and a safety point of view. A transient analysis is therefore required which will simulate the conditions at the break and in the section of pipe either side of the break so that the potential hazard arising from such a situation may be assessed. The analysis could also provide a basis for the design of automatic valve closing devices and alarms which would minimise the effects of such an accident.

Although there have been a few computer programs developed which will model rapid gas transients (for example, Issa [1970], and van Dean and Reintsema [1983]), it was found, on examination, that these models had their limitations. One major consideration was that since the focus of this investigation was on high pressure, dense phase gas transportation, it was essential that the model could simulate the behaviour of a non-perfect gas following a linebreak. The inclusion of realistic estimates of the effects of friction and heat transfer in the model was also a requisite of the program. Therefore it was decided to develop a new computer code which would incorporate these features.

Equations modelling the unsteady gas flow in the pipe, including any effects of wall friction and heat transfer into the pipe, were derived and solved numerically using the method of characteristics.

The computer model that was developed featured a reduced grid size in the vicinity of the break in order to capture in detail the expansion waves created without excessively prolonging the computer run time. It also successfully simulated the flow reversal that would occur in the section of pipe downstream of the break.

Theoretical results produced from this program have been compared with experimental data obtained from various external sources. These were carefully selected to include realistic data from pipelines generally of the same size and containing gaseous fluids similar to those found in typical dense phase gas transmission lines, as well as data from some fundamental shock tube tests. Reasonable agreement was obtained between the theoretical and experimental results.

CHAPTER 2

THEORETICAL DEVELOPMENT OF THE BASIC EQUATIONS

2.1. INTRODUCTION

Basic equations describing homogeneous turbulent gas flow in a pipeline were derived from first principles by defining a control volume of fixed location or translating with uniform velocity (see Figure 2.1). This control volume was of length dx and had a cross-sectional area equal to that of the pipeline. It was assumed that the flow was geometrically one-dimensional, i.e. that all fluid properties were uniform over each cross-section of the pipe. This assumption was examined in detail by Goldwater and Fincham [1980], but briefly it may be stated that for high Reynolds number flows (as in gas transmission lines), the one-dimensional approximation has been shown to be very good for steady and slowly varying flows. There could, however, be some slight deviations when considering large, rapid disturbances.

It has also been assumed that the minor losses, arising from pipe bends, valves and joints, etc., were small compared with the distributed frictional losses, and that the pipe wall was inelastic. The basic partial differential equations describing the flow could then be derived by applying the laws of conservation of mass, linear momentum and energy over a time interval dt.

2.2. CONSERVATION OF MASS

The net rate of mass flow out of the control volume is equal to the rate of decrease of mass within the control volume. Referring to Figure 2.1 below:-

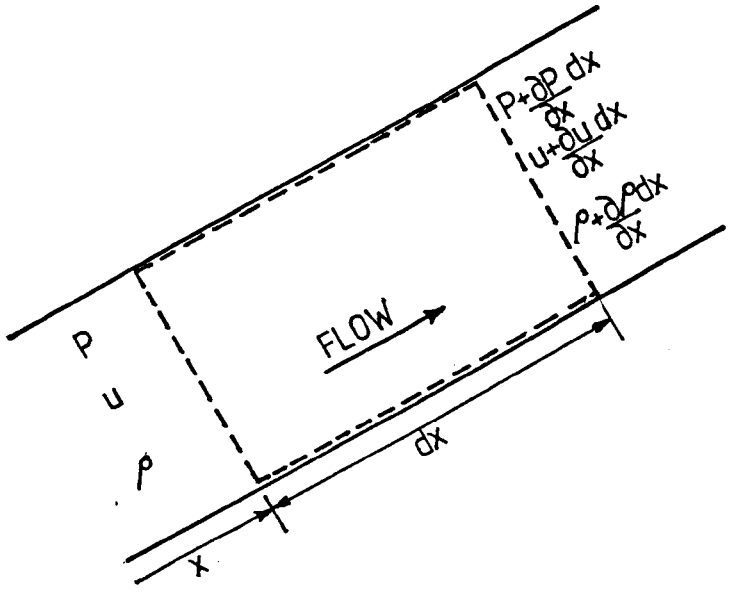


Figure 2.1. Control Volume illustrating the Conservation of Mass

$$A(\rho + \frac{\partial \rho}{\partial x} dx)\ (u + \frac{\partial u}{\partial x} dx) - \rho uA = - \frac{\partial}{\partial t} (\rho A dx)$$

Neglecting very small terms:

$$A(\rho \frac{\partial u}{\partial x} + u \frac{\partial \rho}{\partial x})\ dx = -A\ dx\ \frac{\partial \rho}{\partial t}$$

$$\therefore\ \frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x} (\rho u) = 0 \qquad (2.1)$$

2.3. CONSERVATION OF LINEAR MOMENTUM

The net force acting on the fluid within the control volume is equal to the time rate of change of momentum within the control volume plus the net loss of linear momentum flux. With reference to Figure 2.2:

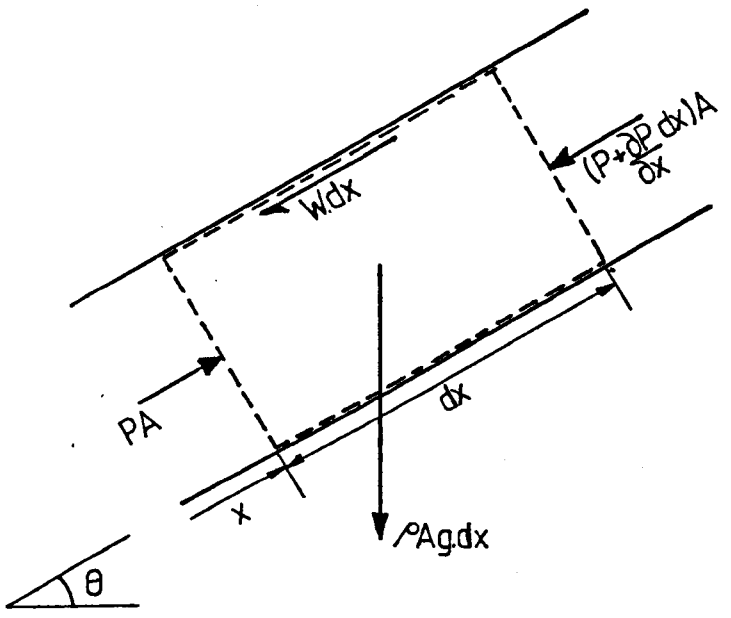


Figure 2.2. Control Volume illustrating the Conservation of Linear Momentum

$$PA - (P + \frac{\partial P}{\partial x}dx)\,A - Wdx - \rho Ag\sin\theta dx = \frac{\partial}{\partial t}(\rho Audx) + \frac{\partial}{\partial x}(\rho Au^2dx)$$

Dividing through by A·dx:-

$$\frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}(\rho u^2) + \frac{\partial P}{\partial x} + \frac{W}{A} + \rho g\sin\theta = 0$$

$$u\left\{\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u)\right\} + \rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial x} + \frac{\partial P}{\partial x} + \frac{W}{A} + \rho g\sin\theta = 0$$

But from equation (2.1):-

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho u) = 0$$

Therefore:-

$$\rho \frac{\partial u}{\partial t} + \rho u \frac{\partial u}{\partial x} + \frac{\partial P}{\partial x} = -\frac{W}{A} - \rho g \sin\theta \qquad (2.2)$$

2.4. CONSERVATION OF ENERGY

If heat is added to the system or work done by the system, the system energy must change according to the First Law of Thermodynamics. With reference to Figure 2.3.:

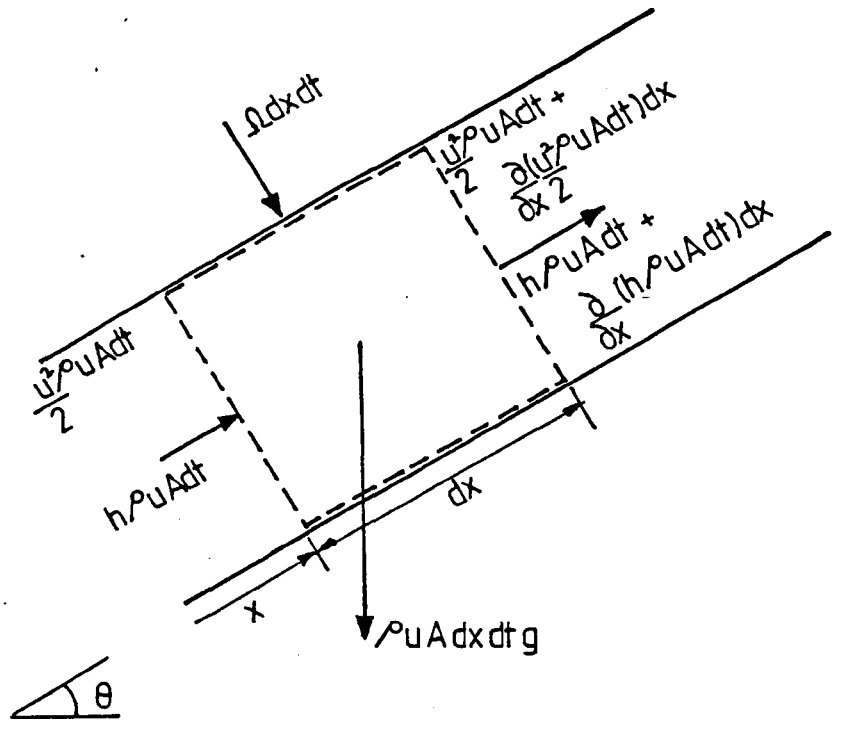


Figure 2.3. Control Volume illustrating the Conservation of Energy

$$\frac{\partial}{\partial x}\left\{\left(h + \frac{u^2}{2}\right)\rho uA\right\} + \frac{\partial}{\partial t}\left\{\left(e + \frac{u^2}{2}\right)\rho A\right\} + \rho Aug\sin\theta = \Omega$$

Dividing through by A:-

$$\frac{\partial}{\partial x}(\rho hu) + \frac{\partial}{\partial x}\left(\rho \frac{u^3}{2}\right) + \frac{\partial}{\partial t}\left(\rho \frac{u^2}{2}\right) + \frac{\partial}{\partial t}(\rho e) + \rho ug\sin\theta = \frac{\Omega}{A}$$

But

$$h = e + \frac{P}{\rho}$$

$$\Rightarrow \frac{\partial}{\partial t}(\rho e) = \frac{\partial}{\partial t}(\rho h) - \frac{\partial P}{\partial t}$$

Substituting back:-

$$\frac{\partial}{\partial x}(h\rho u) + \frac{\partial}{\partial t}(h\rho) + \frac{\partial}{\partial x}\left(\rho \frac{u^3}{2}\right) + \frac{\partial}{\partial t}\left(\rho \frac{u^2}{2}\right) - \frac{\partial P}{\partial t} + \rho ug\sin\theta = \frac{\Omega}{A}$$

Factorizing out:-

$$h\left\{\frac{\partial}{\partial x}(\rho u) + \frac{\partial \rho}{\partial t}\right\} + \rho u \frac{\partial h}{\partial x} + \rho \frac{\partial h}{\partial t} + \frac{u^2}{2}\left\{\frac{\partial}{\partial x}(\rho u) + \frac{\partial \rho}{\partial t}\right\} + \rho u^2 \frac{\partial u}{\partial x}$$

$$+ \rho u \frac{\partial u}{\partial t} - \frac{\partial P}{\partial t} = \frac{\Omega}{A} - \rho ug\sin\theta$$

But from equation (2.1):-

$$\frac{\partial}{\partial x}(\rho u) + \frac{\partial \rho}{\partial t} = 0$$

And from equation (2.2):-

$$u\left\{\rho \frac{\partial u}{\partial t} + \rho u \frac{\partial u}{\partial x}\right\} = -\frac{Wu}{A} - \rho ug\sin\theta - u \frac{\partial P}{\partial x}$$

Therefore:

$$\rho \frac{\partial h}{\partial t} + \rho u \frac{\partial h}{\partial x} - \frac{\partial P}{\partial t} - u \frac{\partial P}{\partial x} = \frac{\Omega + Wu}{A} \qquad (2.3)$$

2.5. BASIC EQUATIONS IN TERMS OF PRESSURE, VELOCITY AND TEMPERATURE

Equations (2.1), (2.2) and (2.3) were re-written with pressure, velocity and temperature as the dependent variables by using the equation of state for a real gas:-

$$P = z\rho RT,$$

and the thermodynamic identity given by Zemanksy [1968]:

$$dh = C_p dT + \left\{\frac{T}{\rho}\left[\frac{\partial \rho}{dT}\right]_p + 1\right\}\frac{dP}{\rho}$$

This method was adopted by van Deen and Reintsema [1983] and the following set of hyperbolic equations is produced:-

$$\frac{\partial P}{\partial t} + u\frac{\partial P}{\partial x} + \rho\, a^2_s \frac{\partial u}{\partial x} = \frac{a^2_s}{C_p T}\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p\right\}\frac{\Omega + Wu}{A} \qquad (2.4)$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{1}{\rho}\frac{\partial P}{\partial x} = -\frac{W}{A\rho} - g\sin\theta \qquad (2.5)$$

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + \frac{a^2_s}{C_p}\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p\right\}\frac{\partial u}{\partial x} = \frac{a^2_s}{C_p P}\left\{1 - \frac{P}{z}\left[\frac{\partial z}{\partial P}\right]_T\right\}\frac{\Omega + Wu}{A} \qquad (2.6)$$

The complete derivation of the above equations is given in Appendix I.

2.6. THE FRICTION TERM

In the basic equations, the friction term 'W' may be defined as the frictional force opposing the flow per unit length of pipe. Since it was assumed that the minor losses are small compared with the distributed losses, the frictional force, W, may be written:-

$$W = \frac{A}{d} \rho f \frac{u|u|}{2} \qquad (2.7)$$

where 'f' is the Darcy friction factor.

To date, there have been no friction factors defined for transient gas flows so it is common practice to adopt the steady flow definitions in cases of unsteady flow. The time-dependent friction factors that have been developed, for example by Brown [1969], Trikha [1975], and Zielke [1968] are only suitable for laminar liquid flows and cannot be adapted to suit the turbulent gas flow found in gas transmission pipelines. The use of a steady flow friction factor for transient flow causes very little error when the flow variations are of relatively low frequency and amplitude. However, when large, rapid disturbances are occurring, a significant error may be incurred. This fact had to be considered when selecting a friction factor and also in the subsequent calculations.

There are various types of steady flow friction factor. It had to be decided whether or not to use a flow dependent friction factor

and also whether to account for the possibility of the liquid phase appearing in the flow.

The key factor in the argument concerning the flow dependent friction factor is whether it could be assumed that fully developed turbulence was achieved in the pipe. If so, the Rough Pipe Law could be employed which is independent of the Reynolds number (and hence the flow). If, however, the flow was in the partially developed turbulent flow regime or even in the transition zone between partially and fully developed turbulence, then the friction factor would vary with changes in the Reynolds number. Examples of these different friction factor relationships are given in Appendix II.

Henry [1969] reported that a flow dependent friction factor should be used for high pressure gas pipelines so that the frictional losses could be determined to within 1%. Opposing this, Issa and Spalding [1972], Stoner [1969], Cronje et al.[1980], and Guy [1967], all claimed that at the high Reynolds numbers encountered, the friction factor could be assumed to be constant and they supported their claims with experimental data.

In this analysis it was decided to initially assume that fully developed turbulence was achieved so that a constant value friction factor could be used. If necessary, a flow dependent friction factor could be substituted into the analysis provided that the improvement in the results obtained justified the additional computing involved.

Since dense phase gases were of particular interest in this project, it was appreciated that during the rapid depressurization following a linebreak, a certain amount of condensation was likely to occur. The two common methods of allowing for the presence of the liquid phase are:-

1) Modification of the Reynolds number and Roughness terms of the friction factor equation. This method was employed by Oliemans [1976] when he modified the Colebrook equation in order to model two-phase flow.

2) Inclusion of an empirical two-phase friction multiplier in the friction term of the basic equations. This method has been used by Mathers et al.[1976], Kawabe [1982] and Chaudhry [1978].

Of these two methods the use of a two-phase friction multiplier was preferred since it did not involve changing standard terms in the equations. However, one important consideration had to be made in that when a linebreak occurs in a pipeline, condensation would not be uniformly spread along the length of the pipe. Instead, it would be localized in the immediate vicinity of the break. After examining the two-phase multiplier developed by Hancox and Nicoll [1972], it was felt that the additional computation involved in adapting this method for a varying dryness fraction along the pipe would not be feasible.

It was therefore decided that a constant value friction factor would be used as defined by a version of the Rough Pipe Law.

Although this friction factor would not initially account for any liquid phase being present, this could be compensated for by a certain amount of 'tuning', if necessary.

2.7. THE HEAT TRANSFER TERM

In the basic equations, the heat transfer term 'Ω' may be defined as the heat flow into the pipe per unit length and per unit time. Although it is considerably smaller in magnitude than the friction term, the heat transfer is still a necessary inclusion especially when considering long distance pipelines.

Typically either an isothermal or an adiabatic approach has been adopted by previous workers. For the case of slow transients caused by fluctuations in demand, it was assumed that the gas in the pipe had sufficient time to reach thermal equilibrium with its constant temperature surroundings. Similarly, when rapid transients were under consideration it was assumed that the pressure changes occurred instantaneously, allowing no time for heat transfer to take place between the gas in the pipe and the surroundings. These are the two extreme cases. In reality a certain amount of heat transfer does occur between the gas and its surroundings although thermal equilibrium will not always be reached.

The heat transfer occurs by means of forced convection through the turbulent boundary layer of the gas in the pipe, conduction

through the pipe wall, and by natural convection outside the pipe. This is shown diagramatically in Figure 2.4.

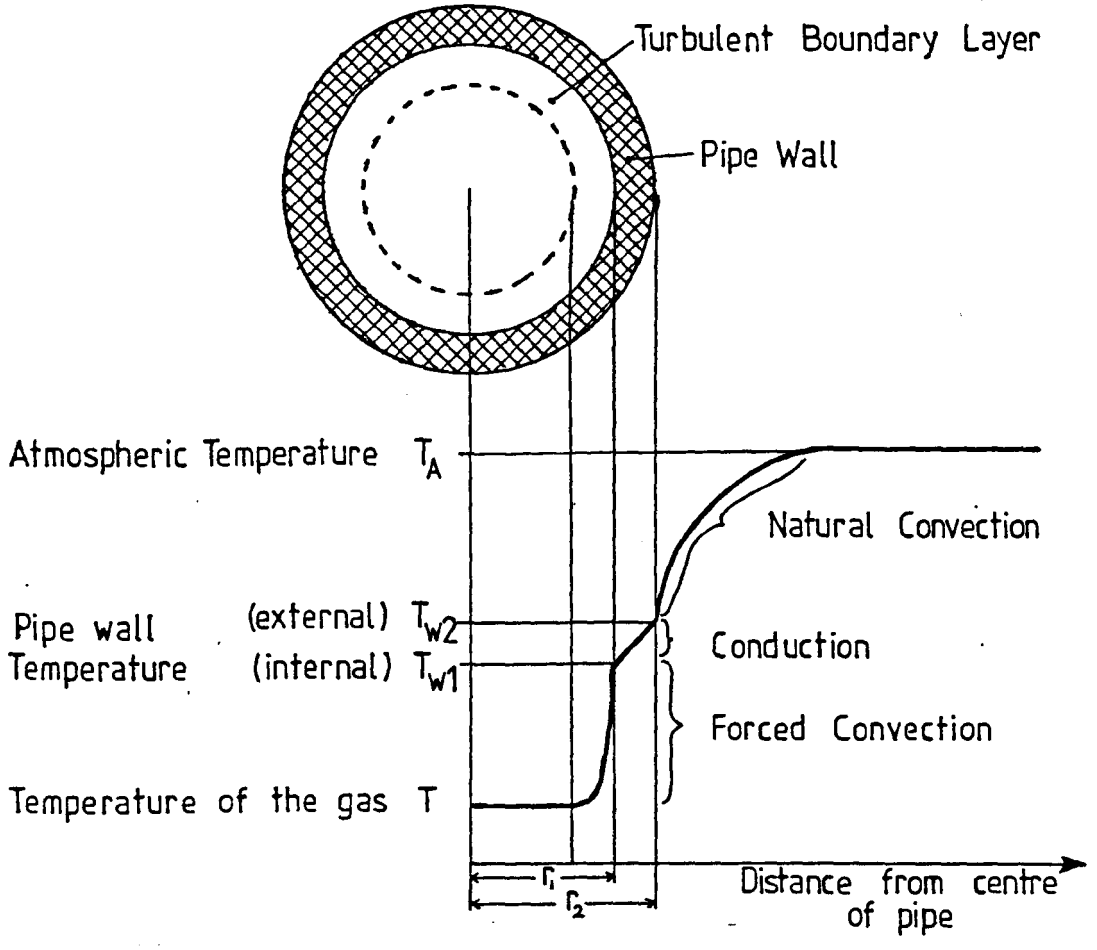


Figure 2.4. Heat Transfer into the pipe.

With reference to Figure 2.4., the heat transfer may be written:-

$$\Omega = 2\pi r_1 \underset{\sim}{h}(T_{w1} - T) = 2\pi \underset{\sim}{k} \frac{(T_{w2} - T_{w1})}{\ln (\frac{r_2}{r_1})} = 2\pi r_2 \underset{\sim}{h}_A (T_A - T_{w2})$$

where $\underset{\sim}{h}$ = convective heat transfer coefficient of the boundary layer

$\underset{\sim}{k}$ = thermal conductivity of the pipe wall

$\underset{\sim}{h}_A$ = convective heat transfer coefficient of the atmosphere.

Unless the pipe is lagged it can be assumed that the high conductivity of the pipe results in a negligible temperature difference between the internal and external pipe walls. A simplified model can then be used as shown in Figure 2.5.

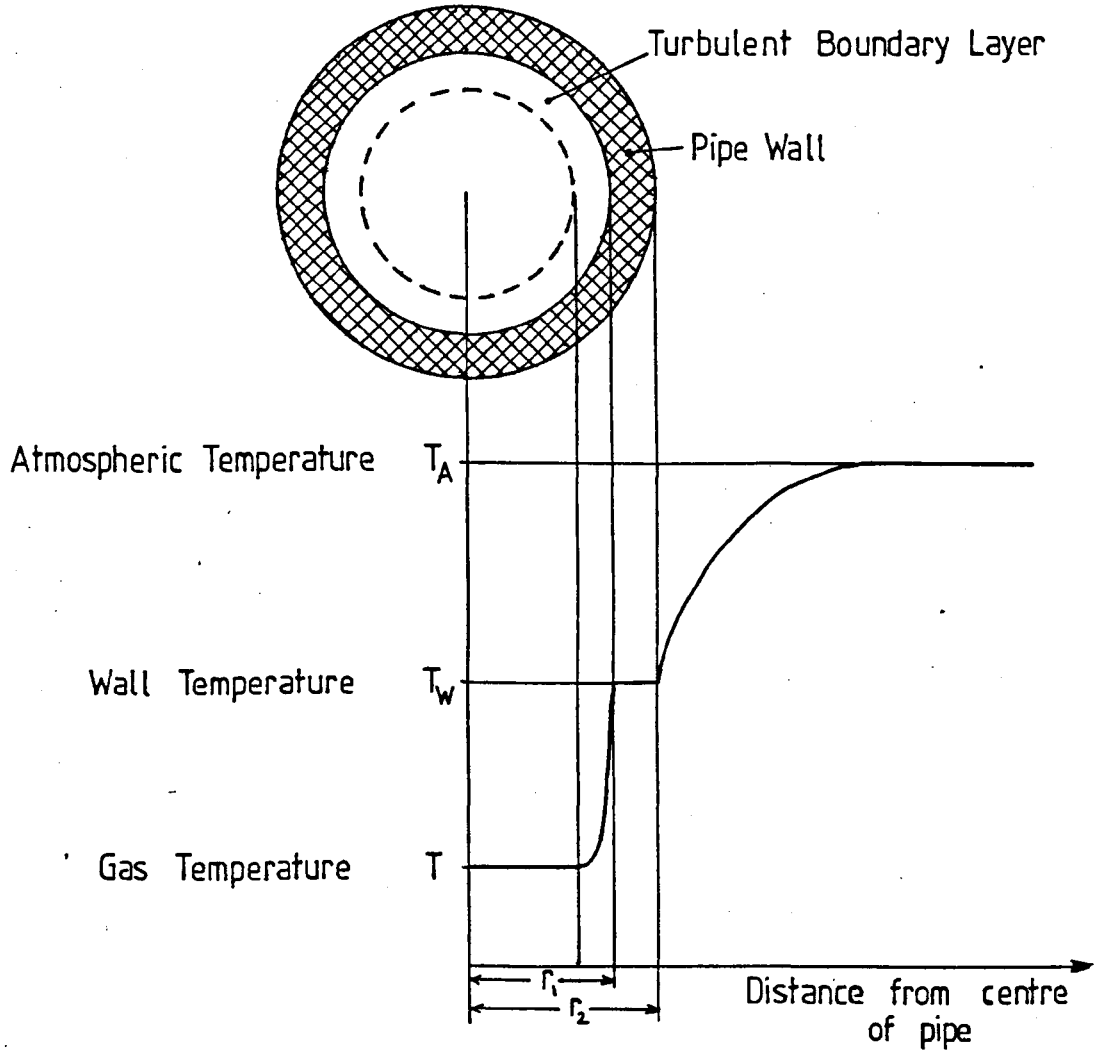


Figure 2.5. Heat Transfer into the Pipe - Simplified Model

Therefore the heat transfer may be defined as:-

$$\Omega = \pi \underset{\sim}{h} d\ (T_w - T) \qquad (2.8)$$

where d = pipe diameter

T_w = mean wall temperature.

Introducing dimensionless parameters, the Stanton number may be defined as the Nusselt number divided by the product of the Reynolds and Prandtl numbers.

$$St = \frac{Nu}{Re.Pr}$$

But,

$$Nu = \frac{\underset{\sim}{h} d}{\underset{\sim}{k}} ,$$

$$Pr = \frac{\nu}{\alpha} , \text{ where } \alpha = \frac{\underset{\sim}{k}}{\rho C_p} \text{ and } \nu = \frac{\mu}{\rho} ,$$

and $$Re = \frac{\rho u d}{\mu}$$

Therefore,

$$St = \frac{(\underset{\sim}{h} d/\underset{\sim}{k})}{(\mu C_p/\underset{\sim}{k})(\rho u d/\mu)}$$

$$= \frac{\underset{\sim}{h}}{\rho u C_p}$$

Substituting this back into equation (2.8):-

$$\Omega = \pi d \rho u C_p St(T_w - T) \qquad (2.9)$$

The Stanton number may initially be calculated from boundary layer theory or taken as a function of the Reynolds and Prandtl numbers. For example, Bakhtar [1956] used the relationship:

$$(St)\cdot(Re)^{0.2}\cdot(Pr)^{0.6} = \text{constant}$$

However, Issa and Spalding [1972] concluded that, as with the friction factor, variations in Stanton number with flow rate were not sufficient to warrant the additional computation involved.

Since the heat transfer term in the basic equations is comparatively small, it was decided to use a constant value Stanton number which may be tuned for each situation encountered.

2.8. THE COMPRESSIBILITY FACTOR

In the basic equations, the compressibility factor 'z' and its derivatives with respect to pressure and temperature are used. There are two methods available for defining the compressibility factor:-

i) Generalized Compressibility Chart

Readings of the compressibility factor may be taken direct from a compressibility chart. The relevant area of this chart for use with high pressure gas pipelines is shown in Figure 2.6.

Although this method may be used in order to obtain an approximate value for 'z', these readings may deviate by as much as 10% from the experimental value for a particular gas. Also, further

calculations are necessary in order to obtain the partial derivatives of 'z'.

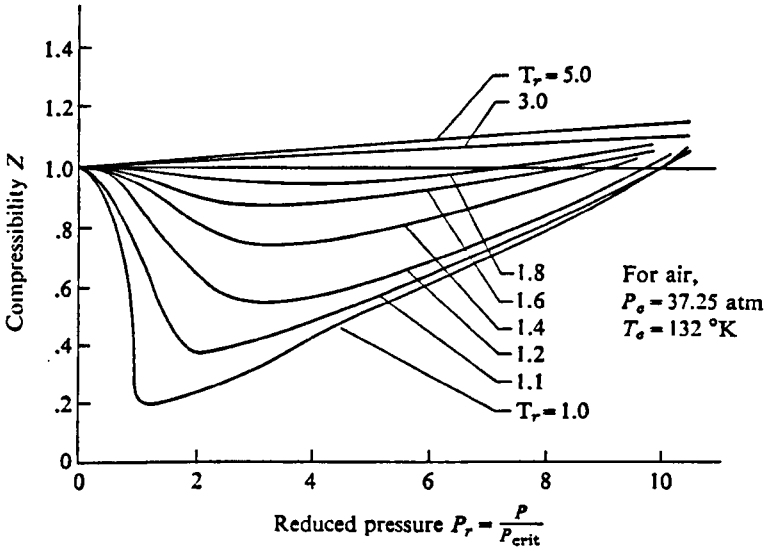


Figure 2.6. Generalized Compressibility Factor Chart

ii) Equations of State

The use of an equation of state to calculate 'z' has the advantages that it can be easily programmed into a computer and that it can also be solved for the derivatives of the compressibility factor. A number of equations of state have been developed which vary in accuracy and complexity. The choice of which equation to use in a given situation is dependent on the type of gas that is being modelled and the temperature and pressure ranges that are likely to be encountered. The amount of available computer time and space that can economically be used should also be considered.

With the modelling of fluid transients it was decided that a complicated equation of state would not be feasible in terms of calculation time. Therefore only the simpler equations were examined.

Van der Waals' Equation [1873]

$$P = \frac{RT}{v - b} - \frac{a}{v^2}$$ where a and b are constant for each gas, $v = \frac{1}{\rho}$.

At the critical point (subscript 'c'):

$$a = \frac{27}{64} \frac{R^2T^2_c}{P_c} \quad \text{and} \quad b = \frac{1}{8} \frac{RT_c}{P_c}$$

Therefore:

$$z = \frac{v'_r}{v'_r - \frac{1}{8}} - \frac{27/64}{T_r \cdot v'_r}$$

where $v'_r = v \cdot \frac{P_c}{RT_c} = \frac{1}{\rho} \frac{P_c}{RT_c}$

and $T_r = \frac{T}{T_c}$

This equation is quite accurate at low pressure, but is inaccurate near the critical point. It is therefore unsuitable for use in calculations for high pressure pipelines.

Dietrici's Equation [1899]

$$P = \frac{RT}{v - b} \cdot \exp\left(\frac{-a}{RTv}\right)$$ where a and b are constants for each gas

At the critical point:-

$$a = \frac{4R^2T^2_c}{P_c(\exp)^2} \qquad b = \frac{RT_c}{P_c(\exp)^2}$$

Therefore:

$$z = \frac{v'_r}{v'_r - (\exp)^{-2}} \cdot \exp\left(\frac{-4}{T_r\, v'_r\, (\exp)^2}\right)$$

This equation is reliable near the critical point for many organic fluids. However, errors are incurred in other regions far from the critical isotherm and hence it cannot be used for largely varying temperatures. The limitations on its use make it unsuitable for gas transient analysis.

Berthelot's Equation [1903]

$$P = \frac{RT}{v - b} - \frac{a}{Tv^2} \qquad \text{where a and b are constants for each gas}$$

At the critical point:

$$a = \frac{27}{64}\,\frac{R^2T^3_c}{P_c} \qquad \text{and} \qquad b = \frac{1}{8}\,\frac{RT_c}{P_c}$$

This equation produces comparatively accurate results for gases and vapours at low temperatures. Since the rapid expansion of a gas following a linebreak would create low temperatures in the pipe, this equation is the most suitable for use in conjunction with the basic equations.

Therefore:

$$z = \frac{v'_r}{v'_r - \frac{1}{8}} - \frac{27/64}{T^2_r\, v'_r}$$

Writing this equation in virial form:

$$z = 1 + \left\{\frac{9}{128} - \frac{27/64}{T^2_r}\right\}\frac{1}{v'_r} + \left\{\frac{9}{128}\right\}^2 \cdot \frac{1}{v'^2_r} + \left\{\frac{9}{128}\right\}^3 \cdot \frac{1}{v'^3_r} + \ldots.$$

In terms of the reduced pressure and temperature only (neglecting higher order terms):-

$$z = 1 + \left\{\frac{9}{128T_r} - \frac{27/64}{T^3_r}\right\} P_r$$

$$= 1 + \left\{\frac{9T_c}{128T} - \frac{27T^3_c}{64T^3}\right\} \frac{P}{P_c} \qquad (2.10)$$

Therefore, if the critical temperature and pressure of the gas are known, the compressibility factor can be calculated directly from the pressure and temperature in the pipeline.

From equation (2.10) the partial derivatives of z with respect to temperature and pressure can be deduced:-

$$\left\{\frac{\partial z}{\partial T}\right\}_P = \left\{\frac{-9T_c}{128T^2} + \frac{81T^3_c}{64T^4}\right\} \frac{P}{P_c} \qquad (2.11)$$

$$\left\{\frac{\partial z}{\partial P}\right\}_T = \left\{\frac{9T_c}{128T} - \frac{27T^3_c}{64T^3}\right\} \frac{1}{P_c} \qquad (2.12)$$

Equations (2.10), (2.11) and (2.12) can then be substituted back into the basic equations.

CHAPTER 3

REVIEW OF THE METHODS OF SOLUTION

3.1. INTRODUCTION

The three hyperbolic partial differential equations derived in the previous chapter (equations 2.4, 2.5 and 2.6) may be solved numerically. A number of different methods of solution have been developed some of which are discussed by P.Fox [1960], L. Fox [1962], Krivoshein et al [1976], Ames [1977] and more recently by Martin and Chaudhry [1983].

In this chapter some of the more popular methods used for modelling fluid transients will be reviewed and an optimum method selected for solving the ruptured pipe problem under investigation.

3.2. THE METHOD OF CHARACTERISTICS

The method of characteristics converts the partial differential equations describing the flow (equations 2.4, 2.5 and 2.6) to ordinary differential equations by using the natural co-ordinates of the system, otherwise known as the characteristics. These ordinary differential equations can then be solved numerically on either a grid of characteristics or on a rectangular grid.

Equations (2.4), (2.5) and (2.6) may be written in matrix form thus:

$$\underline{U}_t + \underline{A}\,\underline{U}_x + \underline{d} = 0 \qquad (3.1)$$

where subscripts t and x denote partial derivatives with respect to time and distances, and where

$$\underline{U} = \begin{bmatrix} P \\ u \\ T \end{bmatrix}$$

$$\underline{A} = \begin{bmatrix} u & \rho a_s^2 & 0 \\ 1/\rho & u & 0 \\ 0 & \frac{a_s^2}{C_p}\left(1+\frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right) & u \end{bmatrix}$$

$$\underline{d} = \begin{bmatrix} -\frac{a_s^2}{C_p T}\left(1+\frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right)\frac{\Omega + Wu}{A} \\ \frac{W}{A\rho} + g\sin\theta \\ -\frac{a_s^2}{C_p P}\left(1-\frac{P}{z}\left(\frac{\partial z}{\partial P}\right)_T\right)\frac{\Omega + Wu}{A} \end{bmatrix}$$

The eigenvalues (λ) of matrix $\underline{A}$ give the characteristic directions which are:-

$$\lambda_1 = u$$
$$\lambda_2 = u + a_s$$
$$\lambda_3 = u - a_s$$

In order to obtain the characteristic equations one needs to determine a transformation matrix $\underline{T}$ such that:

$$\underline{T}\,\underline{A} = \lambda\,\underline{I}\,\underline{T} \qquad (3.2)$$

Then the characteristic equations are given by:

$$\underline{T}\,\frac{d}{dt}(\underline{u}) + \underline{T}\,\underline{d} = 0 \qquad (3.3)$$

Let $\underline{T} = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}$

Solving equation (3.2):-

$$\begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix} \begin{bmatrix} u & \rho a_s^2 & 0 \\ 1/\rho & u & 0 \\ 0 & \frac{a_s^2}{C_p}(1+\frac{T}{z}(\frac{\partial z}{\partial T})_p) & u \end{bmatrix} = \begin{bmatrix} ut_{11} & ut_{12} & ut_{13} \\ (u+a_s)t_{21} & (u+a_s)t_{22} & (u+a_s)t_{23} \\ (u-a_s)t_{31} & (u-a_s)t_{32} & (u-a_s)t_{33} \end{bmatrix}$$

From the above:

$$t_{12} = t_{23} = t_{33} = 0$$

$$\rho a_s^2 t_{11} + ut_{12} + \frac{a_s^2}{C_p}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_p\right) t_{13} = ut_{12}$$

$$\Rightarrow \quad t_{11} = -\frac{1}{\rho C_p}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_p\right) t_{13}$$

$$ut_{21} + 1/\rho t_{22} = (u + a_s)t_{21}$$

$$\Rightarrow \quad t_{22} = \rho a_s t_{21}$$

$$ut_{31} + 1/\rho t_{32} = (u - a_s)t_{31}$$

$$\Rightarrow t_{32} = -\rho a_s t_{31}$$

Therefore the transformation matrix may be written:

$$\underline{T} = \begin{bmatrix} -\frac{1}{\rho C_p}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right) & 0 & 1 \\ \frac{1}{\rho a_s} & 1 & 0 \\ -\frac{1}{\rho a_s} & 1 & 0 \end{bmatrix}$$

and

$$\underline{T}\,\underline{d} = \begin{bmatrix} \frac{a_s^2}{\rho C_p^2 T}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right)^2\left(\frac{\Omega + Wu}{A}\right) - \frac{a_s^2}{C_p P}\left(1 - \frac{P}{z}\left(\frac{\partial z}{\partial P}\right)_T\right)\left(\frac{\Omega + Wu}{A}\right) \\ -\frac{a_s}{\rho C_p T}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right)\left(\frac{\Omega + Wu}{A}\right) + \frac{W}{A\rho} + g\sin\theta \\ \frac{a_s}{\rho C_p T}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right)\left(\frac{\Omega + Wu}{A}\right) + \frac{W}{A\rho} + g\sin\theta \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{a_s^2}{C_p P}\left(1 - \frac{P}{z}\left(\frac{\partial z}{\partial P}\right)_T - \frac{P}{\rho C_p T}\left[1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right]^2\right)\left(\frac{\Omega + Wu}{A}\right) \\ -\frac{a_s}{\rho C_p T}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right)\left(\frac{\Omega + Wu}{A}\right) + \frac{W}{A\rho} + g\sin\theta \\ \frac{a_s}{\rho C_p T}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right)\left(\frac{\Omega + Wu}{A}\right) + \frac{W}{A\rho} + g\sin\theta \end{bmatrix}$$

$$= \begin{bmatrix} -\frac{1}{C_p\rho}\left(\frac{\Omega + Wu}{A}\right) \\ -\frac{a_s}{\rho C_p T}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right)\left(\frac{\Omega + Wu}{A}\right) + \frac{W}{A\rho} + g\sin\theta \\ \frac{a_s}{\rho C_p T}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right)\left(\frac{\Omega + Wu}{A}\right) + \frac{W}{A\rho} + g\sin\theta \end{bmatrix}$$

Hence, the characteristic equations are:-

Along $\frac{dx}{dt} = u$:-

$$-\frac{1}{\rho C_p}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right)\frac{dP}{dt} + \frac{dT}{dt} - \frac{1}{C_p\rho}\left(\frac{\Omega + Wu}{A}\right) = 0 \qquad (3.4)$$

Along $\frac{dx}{dt} = u + a_s$:-

$$\frac{1}{\rho a_s}\frac{dP}{dt} + \frac{du}{dt} - \frac{a_s}{\rho C_p T}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right)\left(\frac{\Omega + Wu}{A}\right) + \frac{W}{A\rho} + g\sin\theta = 0 \quad (3.5)$$

Along $\frac{dx}{dt} = u - a_s$:-

$$-\frac{1}{\rho a_s}\frac{dP}{dt} + \frac{du}{dt} + \frac{a_s}{\rho C_p T}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right)\left(\frac{\Omega + Wu}{A}\right) + \frac{W}{A\rho} + g\sin\theta = 0 \quad (3.6)$$

The method of solving these characteristic equations on a grid of characteristics is known as the natural method of characteristics.

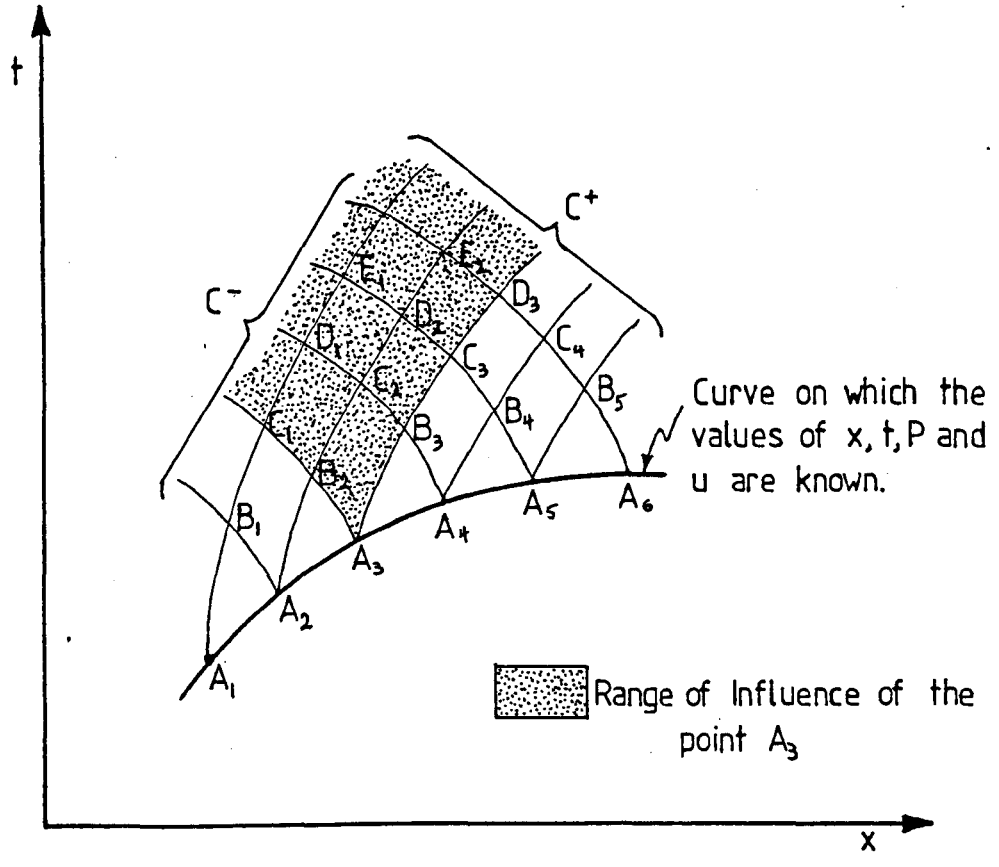


Figure 3.1 Two-dimensional natural grid of characteristics

For two dependent variables (as in the case of isothermal flow) there would be two characteristics through each point as shown in Figure 3.1, and the characteristic equations may be given by:-

$$\frac{du}{dt} \pm \frac{1}{\rho a}\frac{dP}{dt} + \frac{W}{A\rho} + g\sin\theta = 0 \quad \text{along} \quad \frac{dx}{dt} = u \pm a \tag{3.7}$$

A first order finite difference approximation to the C^+ characteristic (referring to the notation of Figure 3.1) produces the following equations:-

$$(u_B - u_{A1}) + \frac{1}{\rho a_{A1}}(P_B - P_{A1}) + \left(\frac{W}{A\rho} + g\sin\theta\right)_{A1}(t_B - t_{A1}) = 0 \tag{3.8}$$

$$\text{and } (x_B - x_{A1}) = (u_{A1} + a_{A1})(t_B - t_{A1}) \tag{3.9}$$

Similarly for the C^- characteristics:-

$$(u_B - u_{A2}) - \frac{1}{\rho a_{A2}}(P_B - P_{A2}) + \left(\frac{W}{A\rho} + g\sin\theta\right)_{A2}(t_B - t_{A2}) = 0 \tag{3.10}$$

$$\text{and } (x_B - x_{A2}) = (u_{A1} - a_{A2})(t_B - t_{A2}) \tag{3.11}$$

This linear approximation is shown below:-

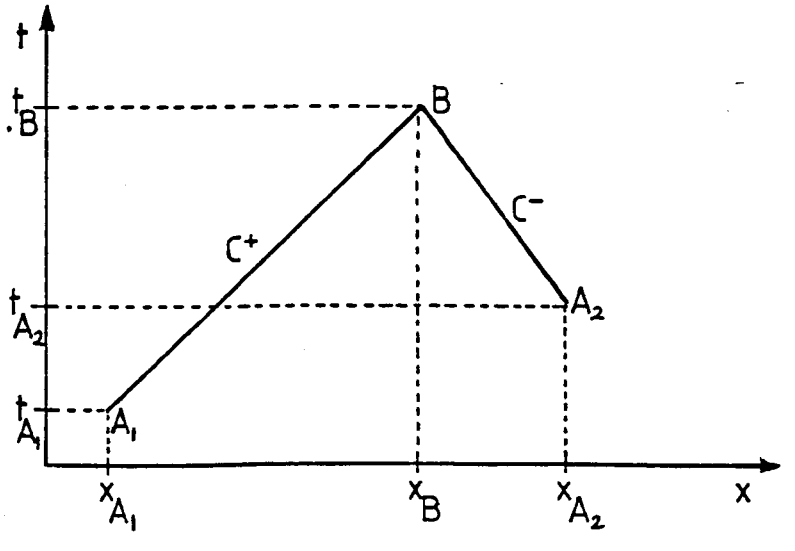


Figure 3.2. Linear characteristics on an x-t plane

Equations (3.8), (3.9), (3.10) and (3.11) can be solved simultaneously for the four unknowns (P_B, u_B, x_B, t_B). Hence it can be realised that if the values of x, t, P and u are known at points A_1, A_2, A_3, A_4, and A_5 in Figure 3.1, then the values of x, t, P and u can be calculated at all the other marked points.

This region of marked points is known as the "domain of dependence" as described by Courant and Friedrichs [1948]. Another feature of the characteristic grid is that the values of x, t, P and u at point A_3 will influence the values of x, t, P and u at succeeding points B_2, B_3, C_1, C_2, C_3, etc. This region, bounded by the characteristics through the point A_3, is known as the "range of influence" and is illustrated in Figure 3.1.

Instead of linearizing the characteristic grid, a second order approximation could be used as expressed by the trapezoidal rule formula.

$$\int_{x_0}^{x_1} f(x)dx \approx \tfrac{1}{2}(f(x_0) + f(x_1))(x_1 - x_0)$$

The use of this formula results in a set of non-linear equations which may be solved by iteration. Higher order methods have been constructed by Ansorge [1963] but, because the number of points to be considered grows exponentially with distance from the line of known values I, the range of applicability is limited. Higher order is more readily achieved by extrapolation although the numerical solution becomes unreliable in the presence of shocks.

The main advantages of the natural method of characteristics are that discontinuous initial data and shock waves do not lead to overshoot and that large time steps are possible since they are not restricted by a stability criterion. However, this method does have two main disadvantages when dealing with rapid gas transients. The first is that if more than two dependent variables are required to describe the system then the complexity of the computation increases and hence computing costs and time become unacceptably high. The second major drawback is that if the solutions of the dependent variables are required at fixed time intervals, then two-dimensional interpolation in the characteristic net is required and this can be very complicated. In order to overcome this second disadvantage, the mesh method of characteristics was developed which solves the characteristic equations on a rectangular coordinate grid. This method directly yields approximate values for the dependent variables at specified time-distance coordinates. However, whereas the natural method of characteristics is unconditionally stable, the mesh method of characteristics is only conditionally stable. The stability criterion, due to Courant-Friedrichs-Levy, is that the domain of dependence of the exact solution is contained within the domain of dependence of the numerical solution. In terms of mesh dimensions Δx and Δt:-

$$\frac{\Delta t}{\Delta x} \leqslant \frac{1}{|u| + a_s} \qquad (3.12)$$

The physical meaning for this stability criterion is given by Benson et al [1964] (page 142).

Taking the case of just two dependent variables, in order to make a direct comparison with the natural method of characteristics, the characteristic lines would appear on a rectangular grid as shown in Figure 3.3.

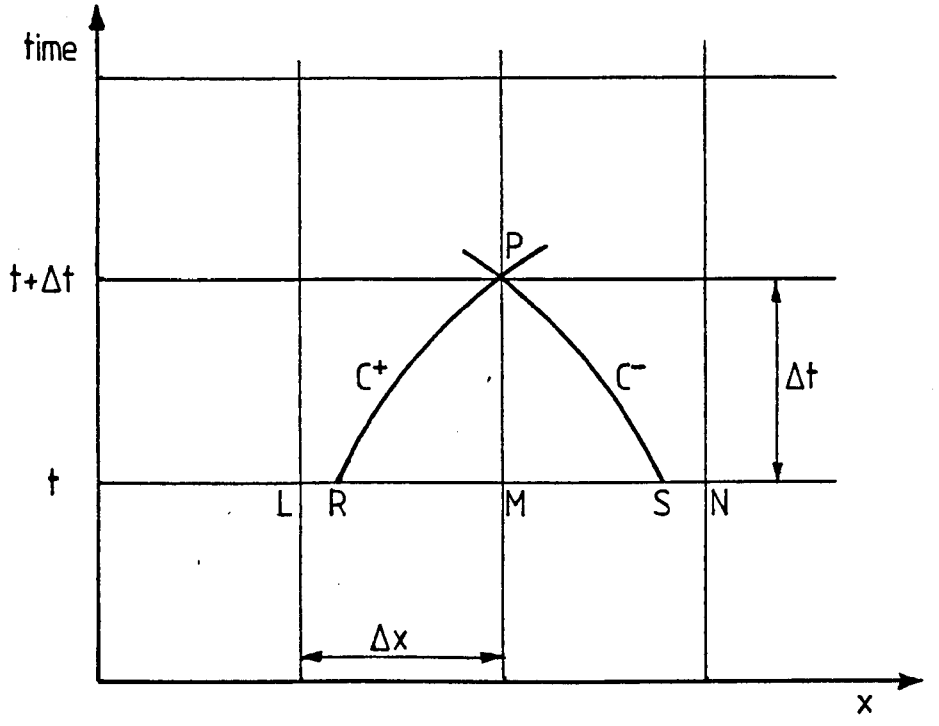


Figure 3.3 Characteristics on a rectangular grid for two dependent variables

The values of the dependent variables at point P can be found by using a first order method described by Courant et al [1952] which assumes that the sections of the characteristics being considered are straight lines. This assumption is valid provided that the time steps, Δt, are sufficiently small. The slopes of the C^+ and C^- characteristics through point M are calculated and these

values taken to be equal to the slopes of the characteristics through point P. From these gradients the positions of points R and S can be determined. The values of the dependent variables at points R and S can then be calculated by interpolating from the values at the grid points L, M and N. Finally the two characteristic equations (equation 3.7) are integrated from R and S up to point P to give the values of the dependent variables at P.

An extension of this method for calculating three dependent variables, as is required for transient non-isothermal gas flow, is used, for example, by Issa and Spalding [1972] and by Cronje et al [1980]. This extended method of solution is a development of that given by Hartree [1952] and it solves the three characteristic equations (equations 3.4, 3.5 and 3.6) on the grid shown in Figure 3.4.

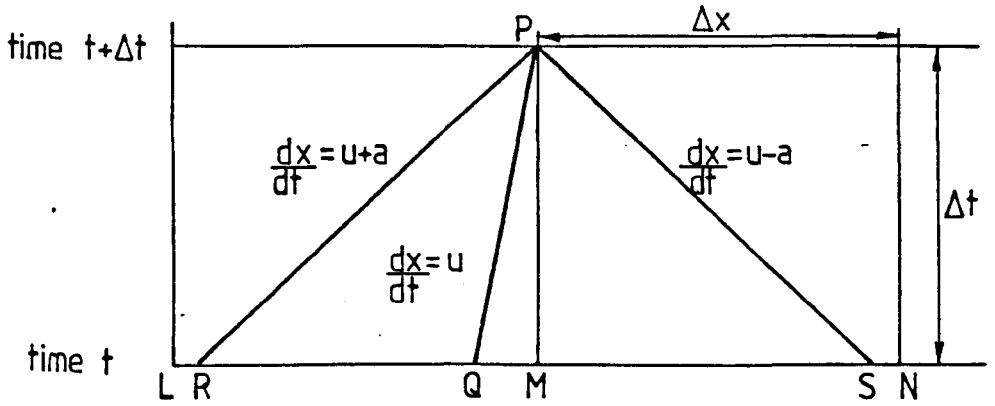


Figure 3.4. Characteristics for three dependent variables on a rectangular grid

In this method a first order approximation is obtained by taking the slopes of each of the characteristics through the point P to be equal to the arithmetic mean of the slopes of the relevant characteristic pertaining to the two adjacent grid points at time t. The procedure then continues as previously outlined for the two dependent variable case, extending the calculations to include the path characteristic through point Q.

Several methods have been developed to increase the accuracy of the solution obtained from the mesh method of characteristics. Lister [1960] describes a second-order method which obtains a higher degree of accuracy by using quadratic instead of linear interpolation. This method was used by Streeter and Lai [1963] to model the water hammer equations with a non-linear friction term included; they obtained good correlation between their theoretical and experimental results. Although Lister only examined the case of two dependent variables, the method could be easily extended to solve for three characteristics provided that the increase in computer time necessary to solve the three simultaneous equations at each iterative step did not create any difficulties. However, Spalding [1969] supported linear interpolation only for modelling three characteristics because "it is the simplest and because more complex procedures appear to have no advantages".

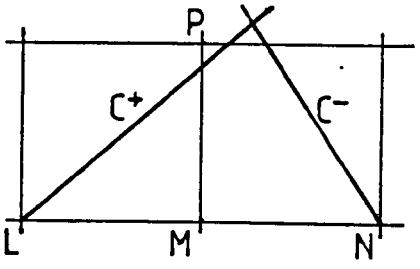
Another way of increasing the accuracy of the solution would be to use extrapolation procedures which enable the elimination of

higher order errors (again at the expense of increased computing time). Details of these methods are given by Hartree [1952] and Roberts [1958].

Although the mesh method of characteristics is only conditionally stable, there are certain circumstances in which adherence to the stability criterion can cause numerical dispersion of the waves. For example, problems arise when the absolute gradients of the C^+ and C^- characteristics differ significantly from each other (as would occur with high Mach numbers) or when the wavespeed varies significantly along the length of the pipe. These two cases are illustrated in Figure 3.5.

High Mach Number flow

- P lies outside the domain of dependence of L and N.



Varying wavespeed flow

- In order to satisfy the stability criteria in the high wavespeed region interpolation errors may be incurred in the low wavespeed region

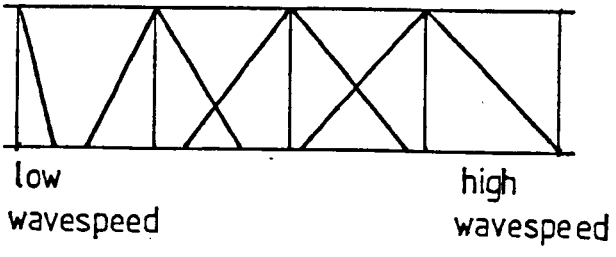


Figure 3.5. Possible Problem areas when using the Mesh Method of Characteristics

In order to overcome such difficulties Vardy [1976] proposed a method in which a variable mesh size is used. He concluded that in certain circumstances, such as high Mach number flows, increased accuracy and/or reduced computing costs could be obtained if $\Delta t/\Delta x$ grid ratios in excess of those permitted by the Courant-Friedrichs-Levy criterion were used, provided that the flow parameters at the base of the characteristic lines were still found by interpolation rather than extrapolation.

Another method of relaxing the stability criterion is by using an inertial multiplier, α. This concept was introduced by Yow [1971]. By assuming that the inertial effect in a natural gas system is insignificant, Yow multiplied the term $(\partial u/\partial t)$ by a factor of α^2 which increased the permissible time step by a factor of α. The choice of α is dependent on the severity of the transient being examined and the accuracy required. Streeter and Wylie[1970] used this method in conjunction with an implicit finite difference method in an attempt to reduce the computing time required to solve gas transients using the method of characteristics. Streeter [1972] also included the inertial multiplier in his discussion of numerical methods for transient flows. In favour of this method, Wylie and Streeter [1978] illustrated that with a 5% error margin, the time step could be increased by a factor of 6 for a rapid transient or by a factor of 40 for a slow transient. However, when utilizing this method, the assumption that the inertial effect of the system is insignificant, must be valid.

Further modifications to the method of characteristics are continually being proposed. For example, Chabrillac [1976] assumed

a linear variation in wavespeed between time steps in order to simplify his model of a loss of coolant accident in a reactor and Carver [1980] transformed the characteristic equations analytically into an equivalent set in which time derivatives are explicitly defined in order to avoid the necessity for iteration or matrix inversion.

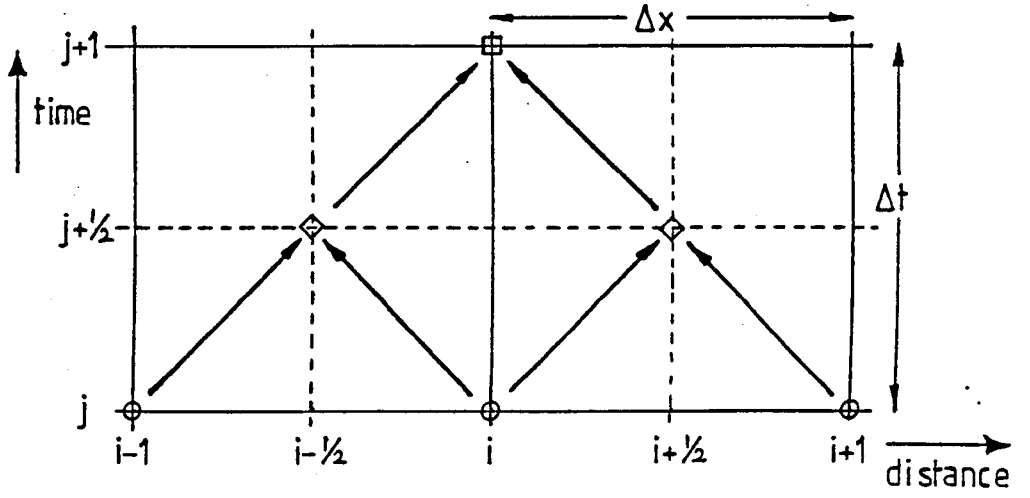
In conclusion, the mesh method of characteristics is a relatively accurate method of solution which can be readily adapted to solve the three dependent variables required for the analysis of non-isothermal, transient gas flow. With this method discontinuities can be handled and boundary conditions are properly posed. It is simple to program on a computer, although the main disadvantage is that it is a comparatively slow method when using a computer because the time steps are restricted by a stability criterion.

3.3. EXPLICIT FINITE DIFFERENCE METHODS

There are many different explicit finite difference methods, ranging from the single-step, first order schemes, such as the method of Lax described by Forsythe and Wasow [1960] (page 85), to the fourth order, four-step method of Abarbanel, Gottlieb and Turkel [1975]. Second order accuracy is normally regarded as sufficient for the analysis of gas transients. Niessner [1980] gives details of higher order methods.

Explicit finite difference methods integrate the basic partial differential equations by considering the changes in the dependent

variables (P, u and T) along the directions of the independent variables (x and t). This produces the solution values at evenly spaced points in the physical plane. The finite difference grid is shown in Figure 3.6.



○ Initial known values

◇ Values found from first step of calculation

□ Values from from second step of calculation

Figure 3.6. The Finite Difference Grid illustrating a two-step method

In order to solve the basic equations using an explicit finite difference method, they must first be written in the "conservative" form. This was defined by Lax and Wendrof [1960] as:-

$$\frac{\partial}{\partial t}(A) + \frac{\partial}{\partial x}(B) = C \qquad (3.13)$$

where A, B and C are functions of the dependent variables.

For the case of transient gas flow in pipes, the three basic conservation equations (equations 2.1, 2.2 and 2.3) may be written in conservative form thus:-

MASS $$\frac{\partial}{\partial t}(\rho) + \frac{\partial}{\partial x}(\rho u) = 0 \qquad (3.14)$$

MOMENTUM $$\frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}(\rho u^2 + P) = -\frac{W}{A} - \rho g \sin\theta \qquad (3.15)$$

ENERGY $$\frac{\partial}{\partial t}\left[(e + \frac{u^2}{2})\rho\right] + \frac{\partial}{\partial x}\left[(h + \frac{u^2}{2})\rho u\right] = \Omega - \rho u g \sin\theta \qquad (3.16)$$

The simplest explicit finite difference method is the forward Euler method. Applying this method to equation 3.13 (assuming that C is equal to zero) produces the following approximation:-

$$A_{(i,j+1)} = A_{(i,j)} - \frac{\Delta t}{2\Delta x}(B_{(i+1,j)} - B_{(i-1,j)}) \qquad (3.17)$$

This method is unconditionally unstable (stability criteria will be discussed later). To overcome this, a damping term must be added to produce:-

$$A_{(i,j+1)} = A_{(i,j)} - \frac{\Delta t}{2\Delta x}(B_{(i+1,j)} - B_{(i-1,j)}) + \frac{\omega_f}{4}(A_{(i+1,j)} - 2A_{(i,j)} + A_{(i-1,j)}) \qquad (3.18)$$

where $0 < \omega_f < 2$ and is the natural frequency of the oscillations. Equation (3.18) is known as the "Method of Lax" and is a single step, first-order method.

In general, a first-order approximation is not sufficiently accurate for modelling gas transients in pipelines and so attention is focused on the second-order methods. A single step second-order finite difference method is the "Method of Lax-Wendroff" as defined by Lax and Wendroff [1960] which can be written as:-

$$A_{(i,j+1)} = A_{(i,j)} - \frac{\Delta t}{2\Delta x}(B_{(i+1,j)} - B_{(i-1,j)})$$

$$+ \frac{1}{4}(\frac{\Delta t}{\Delta x})^2((\frac{\partial B}{\partial A}_{(i+1,j)} + \frac{\partial B}{\partial A}_{(i,j)})(B_{(i+1,j)} - B_{(i,j)})$$

$$-(\frac{\partial B}{\partial A}_{(i,j)} + \frac{\partial B}{\partial A}_{(i-1,j)})(B_{(i,j)} - B_{(i-1,j)})) \qquad (3.19)$$

This method has the disadvantage that additional computing time is required to evaluate $\partial B/\partial A$ as well as B at each step. To avoid the necessity of this calculation there have been numerous two-step methods developed. Probably the most well-known of these is the "Lax-Wendroff two-step". This method was used by Bender [1979] to simulate dynamic gas flows in networks and by Martin et al [1976] to simulate pressure wave propagation in two-phase bubbly air-water mixtures. Cheng and Bowyer [1978] used a generalised form of the Lax-Wendroff two-step method to develop a transient compressible flow code and Gorton [1978] applied the equations to transient steam flow problems. Since it has been used in transient gas analysis, a more detailed description of this method will be given. Taking equation (3.13), the Lax-Wendroff two-step approximation may be described as follows:-

FIRST STEP: $$A_{(i+\frac{1}{2},j+\frac{1}{2})} = \frac{1}{2}[A_{(i+1,j)}+A_{(i,j)}] - \frac{\Delta t}{2\Delta x}[B_{(i+1,j)}-B_{(i,j)}]$$

$$+ \frac{\Delta t}{2}[C_{(i+1,j)}+C_{(i,j)}] + O(\Delta x^2,\Delta t) \qquad (3.20)$$

SECOND STEP: $$A_{(i,j+1)} = A_{(i,j)} - \frac{\Delta t}{\Delta x}[B_{(i+\frac{1}{2},j+\frac{1}{2})}-B_{(i-\frac{1}{2},j+\frac{1}{2})}]$$

$$+\Delta t[C_{(i+\frac{1}{2},j+\frac{1}{2})}+C_{(i-\frac{1}{2},j+\frac{1}{2})}] + O(\Delta x^2,\Delta t^2) \qquad (3.21)$$

where $O(\Delta x^2,\Delta t^2)$ is the "truncation" or "rounding" error. On close examination of these equations, it can be seen that in the first step, the values at all the points at time $t = j+\frac{1}{2}$ can be found. These values are then used in the second step to derive the values at time $t = j+1$. This is illustrated in Figure 3.6.

The MacCormack method (MacCormack [1971]) is also a second order two-step method:-

FIRST STEP $$\bar{A}_{(i,j+1)} = A_{(i,j)} - \frac{\Delta t}{\Delta x}[B_{(i+1,j)} - B_{(i,j)}] \qquad (3.22)$$

SECOND STEP $$A_{(i,j+1)} = \frac{1}{2}\left\{A_{(i,j)} + \bar{A}_{(i,j+1)}\right\} - \frac{\Delta t}{2\Delta x}[\bar{B}_{(i,j+1)} - \bar{B}_{(i-1,j+1)}] \qquad (3.23)$$

Although this method is sometimes used for modelling unsteady gas flow, it produces a slight overshoot at discontinuities and shocks as does the Lax-Wendroff two-step method. This is clearly illustrated by Sod [1978] in his comparison of several finite difference methods.

Another second-order method is the "leap-frog" method described by Roache [1972]. This method involves three time levels within one time step and the approximation for equation (3.13) (assuming that C

is equal to zero) may be written:

$$A_{(i,j+1)} = A_{(i,j-1)} - \frac{\Delta t}{\Delta x}\left[B_{(i+1,j)} - B_{(i-1,j)}\right] \qquad (3.24)$$

This method shows no amplitude error and requires only one evaluation of the value for B at each node point. However, when C of equation (3.13) is not equal to zero, this method becomes unconditionally unstable and to regain stability the calculations become more complicated. Hence this method is not generally used for calculating effects of rapid gas transients.

One of the major drawbacks of the explicit finite difference methods mentioned is that, at best, they are only conditionally stable. For most cases the stability criterion is the same as that defined for the mesh method of characteristics, i.e.

$$\frac{\Delta t}{\Delta x} \leqslant \frac{1}{|u| + a_s}$$

If the Courant number (α_c) is defined as:

$$\alpha_c = (|u| + a_s)\frac{\Delta t}{\Delta x}$$

then this stability criterion can be given by:

$$\alpha_c \leqslant 1$$

One exception to this is the method of Lax (equation 3.18) in which there appears a variable ω_f such that $0 < \omega_f \leqslant 2$. The stability criterion for the method of Lax is:-

$$\alpha_c \leqslant \sqrt{(\frac{\omega_f}{2})}$$

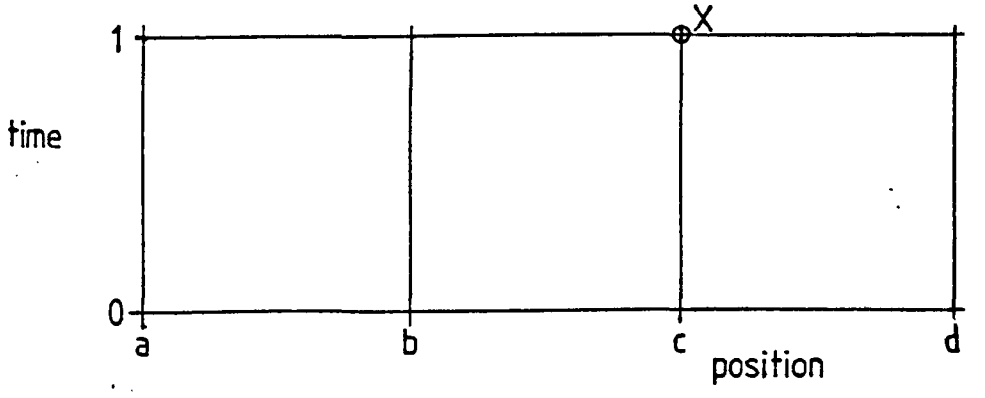
Since the stability criterion restricts the size of the time step which may be used, the explicit finite difference methods require a large amount of computer time and are hence not suitable for the analysis of large systems or for the evaluation of unsteady flows over long periods of time. They are, however, easy to program and need comparatively little computer memory space since they solve the equations directly rather than simultaneously. Explicit finite difference methods can also be used in systems in which a shock forms. To overcome the considerable overshoot and numerical oscillations set up by the shock when using a method of higher than first-order, a smoothing parameter is used. However, extreme care is necessary when using such numerical damping since it can tend to smooth out the transient peaks.

Another disadvantage of this type of method of solution is its inability to solve for the boundary conditions naturally. Considerable work has been concentrated on this area, for example by Gary [1978], Gottlieb and Turkel [1978] and more recently by Shokin and Kompaiets [1987] who also give an extensive review of previous work in this area.

In an attempt to overcome the drawbacks inherent in the explicit finite difference methods, modifications are continuously being made (for example Lakshminarayanan et al [1979]). With these modifications, the economy of this type of method with regard to computer space, and the ease with which it can be programmed make it an attractive method of solution for use with microcomputers.

3.4. IMPLICIT FINITE DIFFERENCE METHODS

The implicit finite difference methods have the advantage over the explicit methods of being unconditionally stable. This implies that the maximum practical time step is limited by the rate of change of the variables imposed at the boundary conditions rather than by a limitation required by a stability criterion. Some of the implicit finite difference methods that have been used in the solution of fluid transient problems are detailed below. The notation used for each method is that illustrated in Figure 3.7.



Property Φ at point X is denoted by Φ_{c1}.

Figure 3.7. An x-t grid for illustrating implicit finite difference methods

(i) Fully Implicit Method

This method is a backward difference method (whereas the explicit finite difference schemes are forward difference methods). It has been used in the analysis of flood propagation in channel systems. For the general equation in conservative form:-

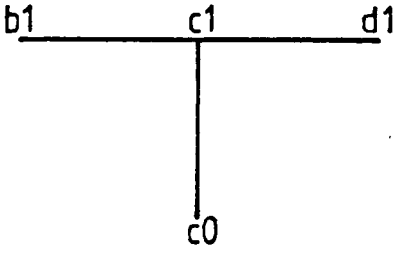
$$\frac{\partial}{\partial t}(A) + \frac{\partial}{\partial x}(B) = C \qquad (3.13)$$

the fully implicit finite difference approximation for the point (c,1) may be written:-

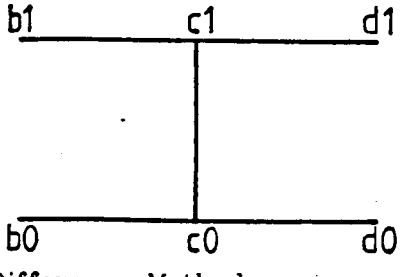
$$\frac{A_{c1} - A_{c0}}{\Delta t} + \frac{B_{d1} - B_{b1}}{2\Delta x} = C_{c1} \qquad (3.25)$$

The node points used in this approximation are shown in Figure 3.8(a).

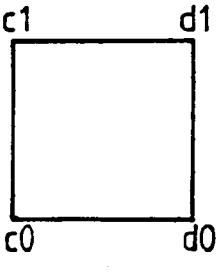
(a) The Fully Implicit Method.



(b) The Crank-Nicolson Method



(c) The Centred Difference Method



(d) The Characteristic Finite Difference Method

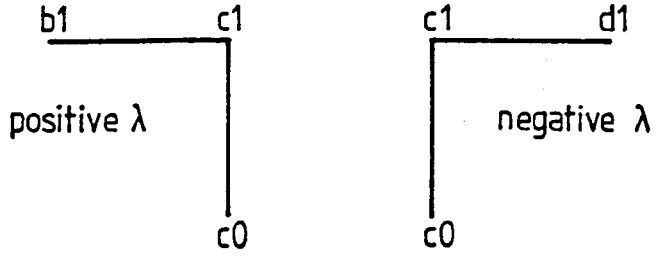


Figure 3.8. Grid points used in various Finite Difference Methods

(ii) The Crank-Nicolson Method

Forsythe and Wasow [1960] reported that the implicit difference methods "seem to have been used for the first time by Crank and Nicolson (1947)" What is now known as the Crank-Nicolson Method is a central difference solution of high order accuracy. This solution is, however, prone to oscillate about the true solution for sudden changes in forcing function. The Crank-Nicolson approximation for equation (3.13) at the point (c,1) may be written:-

$$\frac{A_{c1}-A_{co}}{\Delta t} + \frac{(B_{do}-B_{bo})+(B_{d1}-B_{b1})}{4\Delta x} = C_{c1} \qquad (3.26)$$

Figure 3.8(b) gives the nodal plan for this method. Guy [1967] and Heath and Blunt [1969] used the Crank-Nicolson method to solve the conservation of mass and the conservation of momentum equations for slow transients in isothermal gas flow. Both reseach teams neglected the elevation term ($\rho g \sin\theta$) and the differential of kinetic energy with distance ($\partial/\partial x(\rho u^2)$) in the momentum equation (equation 3.15).

The justification for these omissions is that the relative orders of magnitude of the terms $\partial/\partial x(\rho u^2)$: $\partial/\partial t(\rho u)$: $\partial P/\partial x$ are approximately 0.01: 0.1:1 so it is reasonable to neglect the non-linear term $\partial/\partial x(\rho u^2)$, and the elevation term is often considered to be insignificant.

This method of solution was found to be much simpler than those proposed by Wilkinson et al [1965]. It was easier to program, computed much faster, and could be readily extended to pipeline networks of any size.

(iii) The Centred Difference Method

Wylie et al [1974] used the centred difference method to solve for isothermal gas transients in a network. In this method the partial derivatives are calculated for sections of the pipeline rather than node points. For section c-d in Figure 3.7 the centred difference approximation for equation (3.13) is:-

$$\frac{(A_{d1}-A_{do})+(A_{c1}-A_{co})}{\Delta t} + \frac{(B_{d1}-B_{c1})+(B_{do}-B_{co})}{\Delta x} = \frac{C_{c1}+C_{co}+C_{d1}+C_{do}}{2} \qquad (3.27)$$

The node points used in this approximation are shown in Figure 3.8(c).

A development of this method incorporating upstream weighting was used by Taylor [1978]. This weighted finite difference approximation for equation (3.13) at points P as shown in Figure 3.9 is given by:-

$$\frac{\theta(A_{d1}-A_{do})+(1-\theta)(A_{c1}-A_{co})}{\Delta t} + \frac{\phi(B_{d1}-B_{c1})+(1-\phi)(B_{do}-B_{co})}{\Delta x}$$

$$= (1-\phi)(\theta C_{do}+ (1-\theta)C_{co})+ \phi(\theta C_{d1}+(1-\theta)C_{c1}) \qquad (3.23)$$

where θ and ϕ are the weighting factors

$$0 < \theta < 1 , \qquad 0 < \phi < 1$$

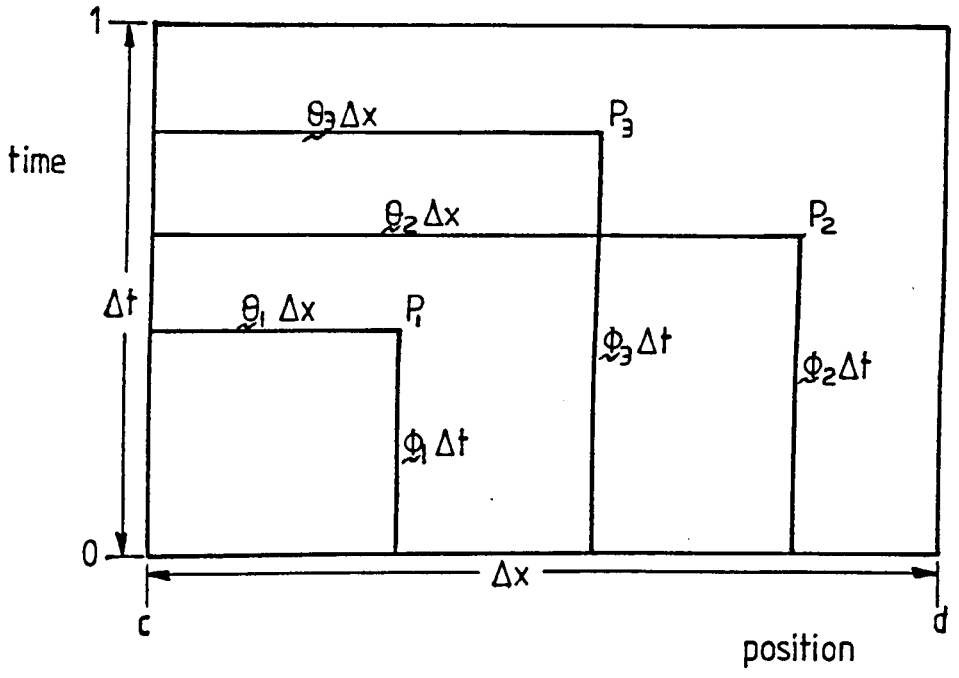


Figure 3.9. Weighted Finite Difference Approximations

(iv) Characteristic Finite Difference Method

The characteristic finite difference method was used by Banerjee and Hancox [1978] and by Chaudhry [1978] to simulate transient homogeneous two-phase flow. It is so called because instead of approximating the conservative form of the basic equations (equation 3.13) it uses the characteristic form of the equations. The characteristic form may be written:-

$$\underline{T} \frac{\partial u}{\partial t} + \underline{\Lambda}\, \underline{T} \frac{\partial u}{\partial x} = \underline{D} \qquad (3.29)$$

where $\underline{T}$ is the transformation matrix defined in equation (3.2), $\underline{u}$ is the column vector of the dependent variables, $\underline{\Lambda}$ is the diagonal matrix of the characteristic directions and $\underline{D}$ is equal to $-\underline{T}\ \underline{d}$ where $\underline{d}$ is defined in equation (3.1). Hence the characteristic form of the basic equations (2.4) to (2.6) is given by:-

$$\underline{T} = \begin{bmatrix} \frac{-1}{\rho C_p}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_p\right) & 0 & 1 \\ \frac{1}{\rho a_s} & 1 & 0 \\ \frac{-1}{\rho a_s} & 1 & 0 \end{bmatrix} \quad ;$$

$$\underline{u} = \begin{bmatrix} P \\ u \\ T \end{bmatrix} \quad ; \qquad \underline{\Lambda} = \begin{bmatrix} u & 0 & 0 \\ 0 & u+a_s & 0 \\ 0 & 0 & u-a_s \end{bmatrix}$$

$$\underline{D} = \begin{bmatrix} \frac{1}{C_p \rho}\left(\frac{\Omega + Wu}{A}\right) \\ \frac{a_s}{\rho C_p T}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_p\right)\left(\frac{\Omega + Wu}{A}\right) - \frac{W}{A\rho} - g\sin\theta \\ \frac{-a_s}{\rho C_p T}\left(1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_p\right)\left(\frac{\Omega + Wu}{A}\right) - \frac{W}{A\rho} - g\sin\theta \end{bmatrix}$$

The difference approximations at point (c,1) of Figure 3.7 for equation (3.29) may be written:-

$$\underline{T}_{co}\left(\frac{\underline{u}_{c1} - \underline{u}_{co}}{\Delta t}\right) + \underline{\Lambda}_{co}\,\underline{T}_{co}\left(\frac{\underline{u}_{c1} - \underline{u}_{b1}}{\Delta x}\right) = \underline{D}_{co} \qquad (3.30)$$

$$\underline{T}_{co}\left(\frac{\underline{u}_{c1} - \underline{u}_{co}}{\Delta t}\right) + \underline{\Lambda}_{co}\,\underline{T}_{co}\left(\frac{\underline{u}_{d1} - \underline{u}_{c1}}{\Delta x}\right) = \underline{D}_{co} \qquad (3.31)$$

Equation (3.30) is used for the positive characteristic directions and equation (3.31) is used for the negative characteristic directions. The relevant node points are shown in Figure 3.8(d).

Combining equations (3.30) and (3.31) gives the difference equation:-

$$\underline{M}_{bc}\,\underline{u}_{b1} + \underline{M}_{cc}\,\underline{u}_{c1} + \underline{M}_{cd}\,\underline{u}_{d1} = \underline{N}_c$$

where

$$\underline{M}_{bc} = -\underline{\Gamma}\ \underline{\Lambda}_c\ \underline{T}_c$$

$$\underline{M}_{cc} = ((\underline{I} - \underline{\Gamma})(\frac{\Delta x}{\Delta t})_{cd} + \underline{\Gamma}(\frac{\Delta x}{\Delta t})_{bc} - (\underline{I} - 2\underline{\Gamma})\underline{\Lambda}_c)\underline{T}_c$$

$$\underline{M}_{cd} = (\underline{I} - \underline{\Gamma})\underline{\Lambda}_c\ \underline{T}_c$$

$$\underline{N}_c = ((\underline{I} - \underline{\Gamma})(\Delta x)_{cd} + \underline{\Gamma}(\Delta x)_{bc})\underline{D}_c + ((\underline{I} - \underline{\Gamma})(\frac{\Delta x}{\Delta t})_{cd} + \underline{\Gamma}(\frac{\Delta x}{\Delta t})_{bc})\underline{T}_c\underline{u}_{co}$$

For subsonic flow:-

$$\underline{\Gamma} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$ if flow is from left to right

$$\underline{\Gamma} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$ if flow is from right to left.

The above definitions for $\underline{M}$ and $\underline{N}$ may be further complicated by the inclusion of a weighting matrix. Details of such a system are given in Banerjee and Hancox [1978] and Chaudhry [1978].

The four methods that have been described are the implicit finite difference methods most commonly used for gas transient analysis although there are others, such as the explicit-implicit methods used by Padmanabhan et al [1978] to solve for pressure transients in bubbly two-phase mixtures or the three time level implicit scheme discussed by Osiadacz [1984].

The major advantage of using an implicit finite difference method is that such methods are unconditionally stable and hence impose no restrictions on the maximum allowable time step. These methods do, however, require the solution of a set of non-linear simultaneous equations (usually by Newton-Raphson linearization) at each time step. For a complicated gas network the matrix becomes quite large, the computer storage requirements become very large and the solution time can become excessive. These drawbacks have been minimised though by the use of a sparse matrix procedure. Other disadvantages of these methods of solution (Streeter [1971]) are that they can yield unsatisfactory results for sharp transients and that some implicit methods have been known to produce erratic results during the imposition of some types of boundary condition. Although implicit methods are suitable for the analysis of slow transients on relatively large networks, the computer programs based on these methods do not allow easy extension.

3.5. SOLUTION OF A RIEMANN PROBLEM - RANDOM CHOICE AND FLUX DIFFERENCE SPLITTING SCHEMES

In order to overcome some of the difficulties presented by the finite difference methods when solving transient gas flow equations, Chorin [1976] developed a method of solution originally introduced by Glimm [1965].

The characteristic form of the basic equations (Equation 3.29) may be written in a general form (assuming $\underline{D} = 0$) as:

$$\underline{u}_t = (f(u))_x \tag{3.32}$$

where subscripts x and t denote partial differentiation.

If the time t and space x are divided into intervals of length k and h respectively, then the solution is to be evaluated at times $t = nk$ and $t = (n + \tfrac{1}{2})k$ and at points $x = \pm ih$ and $x = \pm (i + \tfrac{1}{2})h$ where n and i are integers. In order to do this the near constant initial data is replaced by discontinuous data:-

$$\left.\begin{aligned} \underline{u}(x,o) &= u^n_{i+1} \qquad \text{for } x \geqslant 0 \\ \underline{u}(x,o) &= u^n_i \qquad \text{for } x < 0 \end{aligned}\right\} \tag{3.33}$$

Equation (3.33) is the Riemann problem which is, by definition, the interaction between two adjacent and initially uniform states. Glimm [1965] introduced a random variable θ', equidistributed over the interval $(-\tfrac{1}{2}, \tfrac{1}{2})$ with values θ'_i, and using this defined the solution of the Riemann problem at the point $(\theta'_i h, k/2)$ to be $\underline{V}(\theta'_i h, k/2)$. This value was then allotted to $u^{n+\frac{1}{2}}_{i+\frac{1}{2}}$ and by a similar process the calculation could proceed to the next time step. The actual solution of the Riemann problem is obtained using an adapted version of the above developed by Godunov [1959].

The original aim of the random choice method was to be a numerical method for solving non-linear hyperbolic systems where a complex pattern of shock waves and slip planes exists (for example in combustion engines). Hence, although it has the advantage of being unconditionally stable, its complexity and execution time (2-3 times

that of a standard finite difference scheme) make it unsuitable for solving simpler situations with smoother solutions. Sod [1978] also found that when applying this method to a simple shock tube problem, the randomness of the method produced slight displacement of shock and contact discontinuities and small deviations in the rarefaction wave. However, since at best this method has only first order accuracy, the solution at boundaries creates no difficulties.

A method known as the λ formulation which also makes use of Riemann invariants was introduced by Moretti [1979]. This method has been successfully applied to multi-dimensional flow, obtaining sufficient accuracy with comparatively low computing time. The major drawback with this method is that in most cases, shock waves need to be treated explicitly to correctly evaluate their propagation.

One method that does capture the shock wave numerically is the flux-vector splitting method proposed by Steger and Warming [1981]. This method is very diffusive when a first order technique is used and when higher order techniques are employed, post-shock oscillations develop (as described by Mulpuru [1983]). However, this problem can be overcome by using a non-linear weighting procedure developed by Zalesak [1979]. This produces a hybrid scheme which can be extended to higher spatial dimensions through time splitting. The disadvantage in this is the increased computing costs that are incurred.

A second method for numerically capturing shock waves is the flux difference splitting method. This method has been developed during the last decade from the pioneering work of Godunov [1959]. In principle the following procedure is used.

The difference in flux between two adjacent node points is split into terms that will affect the flow evolution at points either side of the section under investigation. The initial continuous data is approximated by piecewise constant data which assumes that the flow at each node point and over the cell extending for half a grid interval either side of the node point is uniform. (Extensions to this are the use of a piecewise linear approximation by van Leer [1979] and the use of a piecewise parabolic approximation by Woodward and Colella [1982]). A discontinuity generally separates two neighbouring cells in the middle of the interval and the evolution in time of this discontinuity provides the criteria for splitting the flux difference over an interval into terms associated with waves that propagate up or down the pipe. This criteria for flux splitting was used by Roe {1} [1981], and by Osher and Soloman [1982], although other criteria have been used, for example by Lombard et al [1982].

A good detailed description of this method of solution for the basic equations with the source terms omitted is given by Roe and Pike [1984]. Pandolfi [1984] extends the analysis to the set of hyperbolic equations (previously defined) in the form:

$$\frac{\partial}{\partial t}(A) + \frac{\partial}{\partial x}(B) = C \qquad (3.13)$$

where, for the basic equations

$$A = \begin{bmatrix} \rho \\ \rho u \\ (e + \frac{u^2}{2})\rho \end{bmatrix}$$

$$B = \begin{bmatrix} \rho u \\ \rho u^2 + P \\ (h + \frac{u^2}{2})\rho u \end{bmatrix}$$

$$C = \begin{bmatrix} 0 \\ -\frac{W}{A} - \rho g \sin\theta \\ \Omega - \rho u g \sin\theta \end{bmatrix}$$

as given in equations (3.14) to (3.16).

With reference to the elemental section shown in Figure 3.10,

Let $B_{i+1} - B_i = \Delta_i B$

The term $\Delta_i B$ is the flux difference and the corresponding term $\Delta_i B.\Delta t/\Delta x$ can be interpreted as the contribution of the interval $(x_{i+1} - x_i)$ to the variation in time, from t_o to t_1 of the vector A. In general the waves will travel in both directions in the pipeline and so it is necessary to split the term $\Delta_i B$ into parts that will affect the points upstream or downstream of the interval under consideration.

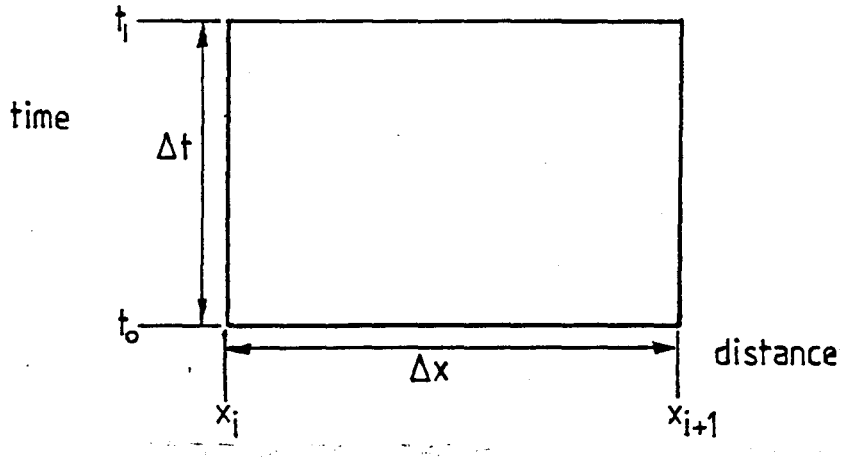


Figure 3.10. Elemental section for flux difference splitting

At time t_o let there be uniform flow B_i in the interval $(x_{i+½} - x_i)$ and uniform flow B_{i+1} in the interval $(x_{i+1} - x_{i+½})$. A discontinuity ($\Delta_i A$ and/or $\Delta_i B$) separates the two half intervals at the centre $(x_{i+½})$. This is shown in Figure 3.11. The evolution in time of this discontinuity is the solution of a Riemann problem.

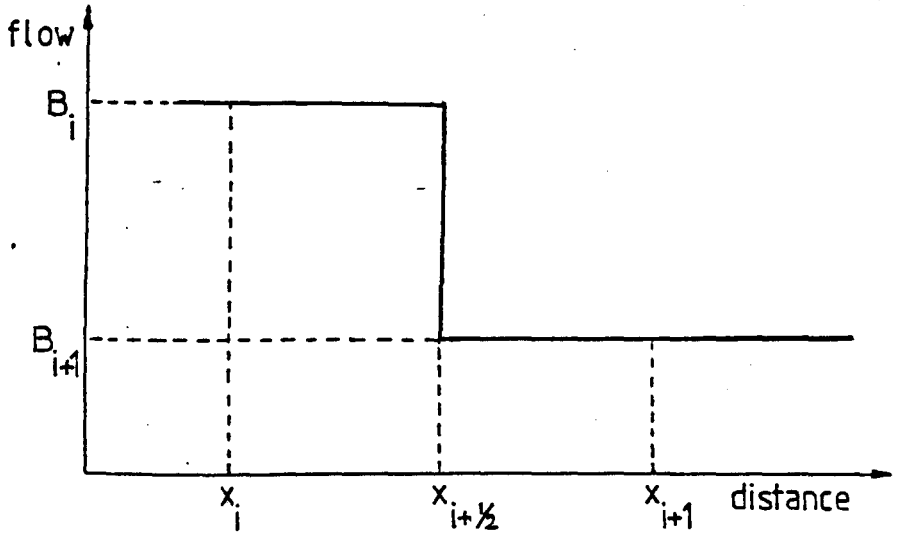


Figure 3.11. Discontinuity between two half sections

Since for this problem there will be three waves corresponding to the three characteristic directions (u+a, u, u-a) the difference of flux through the initial discontinuity $\Delta_i B$ is split into three terms:

$$\Delta_i B = (\Delta_i B)_1 , + (\Delta_i B)_2 + (\Delta_i B)_3$$

Roe {2}[1981] reported that the exact solution of this Riemann problem is not essential to obtain good numerical results, which is fortunate considering the large truncation errors that would be incurred in the iterative process required to obtain the exact solution. Instead, the Riemann problem is solved approximately to save on computing time, and it is the different ways of approximating that identify the different flux-splitting methods.

Although very good results have been obtained from numerical experimentation, these methods do have the basic disadvantage that a considerable amount of computer time is required to split the flux difference. Furthermore, if a second-order method is used for the integration the computational time is again increased. Also it has been noted that some inaccuracies can develop in cases such as the interaction of shocks.

3.6. FINITE ELEMENT ANALYSIS

In the past finite element methods have not been widely used for gas transients since the procedure is lengthy and tedious and hence the computing time and storage requirements are high. However,

they do offer some advantages over the finite difference methods in that the element size, shape and distribution is relatively flexible so that non-uniform internal distributions of nodal points is possible. They can also handle some boundary conditions better than finite difference methods.

The steps involved in the standard finite element method of solution are as follows:-

i) Subdivision of the pipeline into subregions or finite elements - the size, shape and distribution must be decided.

ii) Selection of the shape functions - the dependent variables may be approximated by different shape functions in each element. The shape functions are usually polynomials, the simplest of which is the linear or chapeau representation. The higher order polynomials yield more accurate solutions unless the solution contains discontinuities in which case this does not always hold true.

iii) Derivation of element behaviour - A relationship is obtained for a typical element and from this the behaviour of all the individual elements may be computed.

iv) Application of the boundary conditions - The boundary conditions are applied by modifying the overall algebraic equations.

v) Solution of the overall equations - These equations, especially when non-linear, are usually solved iteratively.

Rachford and Dupont [1974] used a Galerkin finite element method based on Hermite Cubic polynomials to simulate isothermal transient gas flow. The Galerkin method is a two-step method which reduces the partial differential equations to ordinary differential equations. Fincham and Goldwater [1979] examined the use of the Galerkin method for simulating gas transmission networks and Morton and Parrott [1980] explored the solution of first order hyperbolic equations using generalised Galerkin methods. However, because this type of solution takes a comparatively long time to execute, it is generally unpopular for transient analyses.

Watt, Boldy and Hobbs [1980] investigated the possibility of marrying-up a finite difference procedure with a finite element component for a system assuming negligible density variation. The computation involved with this idea would be a serious deterrent for expanding the system to three dependent variables.

The finite element methods have been developed over the years. For example, Van Goetham [1978] modelled unsteady compressible flow problems using a variable domain finite element method. However, the most promising development must be the moving finite element method of Wathen and Baines [1983]. In this method a set of ordinary differential equations is obtained by approximating the initial data using a piecewise linear function on a number of nodes.

These ordinary differential equations can then be solved using a simple explicit finite difference method. Spivack [1984] summarises this method and illustrates its application to solving the unsteady gas flow equations in the pipebreak problem. Further details of the application of this method to compressible flow problems are given by Baines [1986].

The only apparent drawbacks with this method of solution are that care is needed in the treatment of boundary conditions and that it is very complicated to program.

3.7. DISCUSSION

Each of the numerical methods in this chapter has its own advantages and disadvantages. The optimum method of solution for a particular fluid transient analysis is determined mainly by the accuracy and depth of detail that is required of the results and the type of transient waves that are being modelled.

In general, higher degrees of accuracy can be achieved at the expense of increased computational labour. Although the implicit finite difference schemes are often more economical then the explicit finite difference schemes or the method of characteristics, it is widely accepted that more accurate results can be achieved with the method of characteristics. When using the mesh method of characteristics, errors can be introduced when the characteristics are approximated to straight lines. These discretization errors can,

however, be reduced by employing parabolic arcs in place of straight lines to give a second order approximation. This does have the drawback of increasing the computational run time still further.

As well as the run time, the available computer memory space is an important factor when working on a microcomputer. The implicit finite difference method is unsuitable for small computers since it requires the solution of a set of non-linear simultaneous equations at each time step. For a complicated network, the matrix therefore becomes quite large and the computer storage requirements become very large. However, the computer storage and time required can be reduced by using sparse matrix algebra (Wylie et al [1974]).

The type of transient flow that will occur - whether it is compression or rarefaction waves that are being created must be considered. In the case of compression waves where shock waves are being set up in the system, a method must be chosen that will accurately represent the shock waves without smearing the details or overshooting. The Lax-Wendroff two-step explicit finite difference method is one of the most suitable for dealing with systems in which a shock wave forms. The natural method of characteristics is also accurate but requires special procedures for shock calculations. Alternatively, the mesh method of characteristics or an extension of this method such as the flux difference splitting scheme presented by Roe and Pike [1984] both recognise shocks and cause only small overshoot. However, the finite difference methods tend to produce

overshoot in the presence of shocks if methods of higher than first order are used, and the discontinuities tend to get rounded off due to numerical diffusion.

If the analysis is solely concerned with slow transients, such as those caused by fluctuations in demand in a network, then considerable savings can be made in computational time, and hence cost, by utilizing an implicit finite difference scheme which does not require a small time step for stability. However, with rapid transients such as those caused by a linebreak or compressor failure, a small time step is necessary and hence the Method of Characteristics would often tend to be favoured.

Another important factor to consider is the number of type of boundary conditions that will be imposed on the system. For example, the explicit finite difference methods cannot handle boundary conditions naturally and so calculations for networks with many branches becomes difficult. The mesh method of characteristics does have the advantage that the boundary conditions are properly posed whereas for most methods of solution care is needed in the treatment of the boundary conditions.

Finally, the size of the system (i.e. the number of pipes) to be analysed will influence the choice of method. Implicit finite difference methods are more suited to the analysis of large systems whereas the mesh method of characteristics and the explicit finite difference approaches are comparatively slow and so are more

appropriate for single pipelines rather than networks. However, for an analysis on an expanding network, the implicit methods may not be the optimum since they do not allow easy extension.

In this analysis of a ruptured gas pipeline, the system consists of a single pipe with no possibility of shock waves forming. The numerical method chosen must be capable of solving for three dependent variables and a variable wavespeed. The major requirement here, though, is for an accurate and reliable numerical method in order that the theoretical model may be assessed. The method of characteristics was selected as the most appropriate.

CHAPTER 4

SOLUTION OF THE CHARACTERISTIC EQUATIONS FOR A LINEBREAK SITUATION

4.1. INTRODUCTION

When a linebreak occurs in a high pressure pipeline, a pressure drop occurs virtually instantaneously at the break and rarefaction waves are transmitted up and down the pipeline. When the fluid in the pipe is a gas, these rarefaction waves are rapidly dissipated. For this reason, it was decided that in order to model these waves properly, a reduced grid size was required in the vicinity of the break.

A physical model was initially specified that reduced the grid size by a factor of 10 near the break and by a further factor of 10 immediately either side of the break. However, it was found that such a dramatic grid size reduction caused numerical instabilities in the solution. A second model was therefore developed that reduced the grid size near the break by a factor of 2 six times as shown in Figure 4.1. This gradual grid size reduction minimised the instabilities previously experienced.

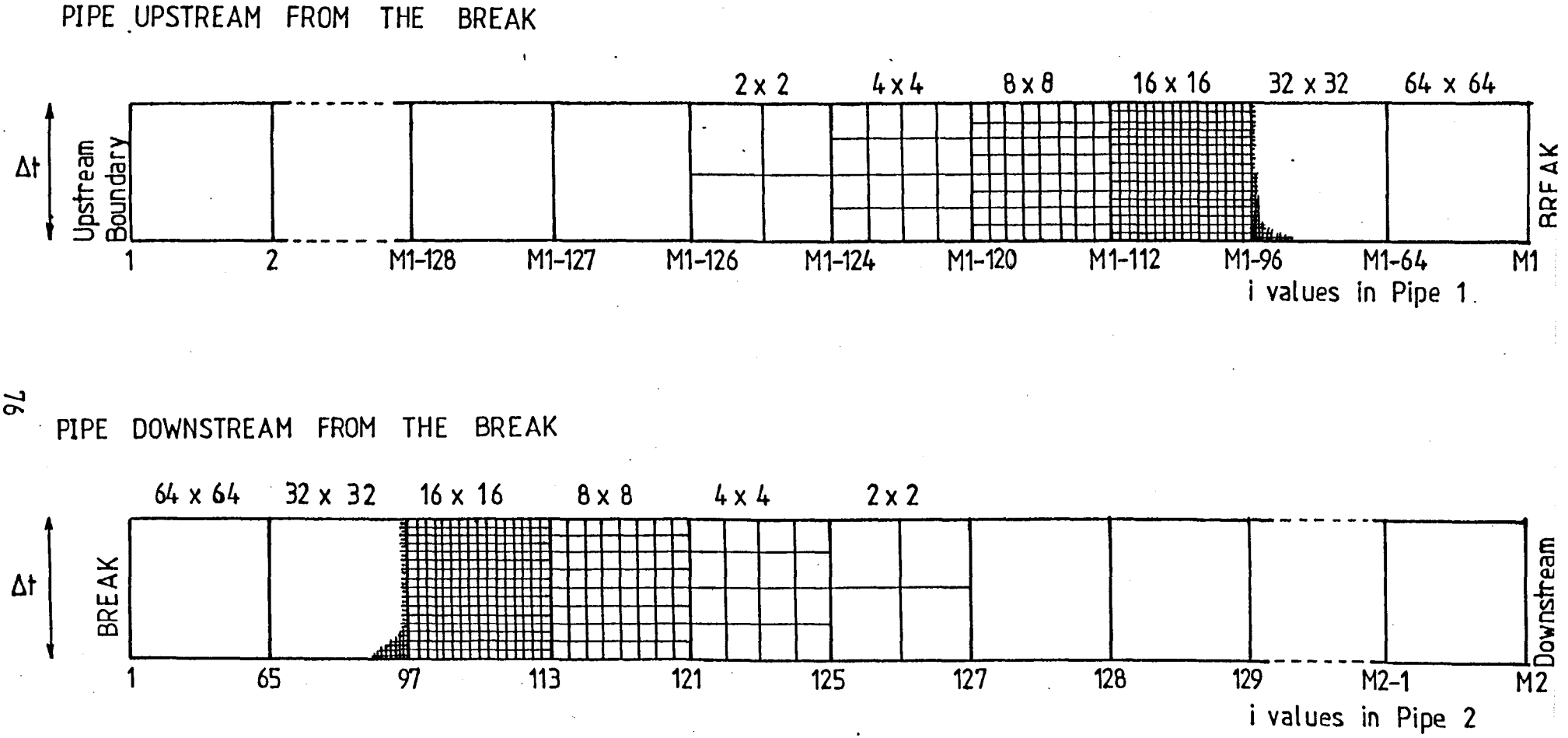


Figure 4.1. Grid Size Variation for Modelling a Linebreak

4.2. GENERAL SOLUTION FOR INTERNAL POINTS

Since the pressure drop along a pipeline is due primarily to the frictional effects (gravitational effects being small), the friction term (W) in the characteristic equations requires a second-order approximation. The frictional force may be written in terms of P, T and u:-

$$W = \frac{A}{d} \cdot f \cdot \frac{P}{RTZ} \frac{u|u|}{2}$$

Differentiating this:

$$dW = \frac{\partial W}{\partial P} dP + \frac{\partial W}{\partial T} dT + \frac{\partial W}{\partial u} du + \frac{\partial W}{\partial Z} dZ$$

Neglecting changes in compressibility factor, this expression simplifies to:

$$dW = \frac{W}{P} dP - \frac{W}{T} dT + \frac{2W}{u} du$$

If state 1 is defined as the conditions at the base of the characteristic being examined and state 2 as the predicted conditions at the point in question, then an average value for W over the interval from state 1 to state 2 may be calculated:

$$\overline{W} = \frac{W_1 + W_2}{2}$$

$$= W_1 + \left\{\frac{W_2 - W_1}{2}\right\}$$

$$= W_1 + \frac{dW}{2}$$

$$= W_1 \left\{1 + \frac{P_2 - P_1}{2P_1} + \frac{u_2 - u_1}{u_1} - \frac{(T_2 - T_1)}{2T_1}\right\}$$

This equation was substituted into the finite difference form of the characteristic equations and the following set of equations was produced.

Along dx/dt = u:

$$- \frac{1}{\rho C_p}\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p\right\}\left[P_2-P_1\right]+\left[T_2-T_1\right] - \frac{\Delta t \Omega}{\rho C_p A} - \frac{\Delta t u W}{\rho C_p A}\left[\frac{P_2}{2P_1} + \frac{u_2}{u_1} - \frac{T_2}{2T_1}\right] = 0$$

Along dx/dt = u + a_s:

$$\frac{(P_2-P_1)}{\rho a_s} + (u_2-u_1) - \frac{a_s \Delta t}{\rho C_p T}\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p\right\}\frac{\Omega}{A} + \frac{W \Delta t}{A\rho}\left[\frac{P_2}{2P_1} + \frac{u_2}{u_1} - \frac{T_2}{2T_1}\right]$$

$$- \frac{a_s \Delta t}{\rho C_p T}\left\{1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_p\right\}\frac{Wu}{A}\left[\frac{P_2}{2P_1} + \frac{u_2}{u_1} - \frac{T_2}{2T_1}\right] + g \cdot \Delta t \sin\theta = 0$$

Along dx/dt = u - a_s:

$$- \frac{(P_2-P_1)}{\rho a_s} + (u_2-u_1) + \frac{a_s \Delta t}{\rho C_p T}\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p\right\}\frac{\Omega}{A} + \frac{W \Delta t}{A\rho}\left[\frac{P_2}{2P_1} + \frac{u_2}{u_1} - \frac{T_2}{2T_1}\right]$$

$$+ \frac{a_s \Delta t}{\rho C_p T}\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p\right\}\frac{Wu}{A}\left[\frac{P_2}{2P_1} + \frac{u_2}{u_1} - \frac{T_2}{2T_1}\right] + g \cdot \Delta t \sin\theta = 0$$

In these equations, Δt is the length of the time step and all variables without a subscript refer to the value at state 1.

For simplicity, these equations were then written in matrix form:-

$$\begin{bmatrix} A(1) & A(2) & A(3) \\ A(4) & A(5) & A(6) \\ A(7) & A(8) & A(9) \end{bmatrix} \begin{bmatrix} P_2 \\ T_2 \\ u_2 \end{bmatrix} = \begin{bmatrix} B(1) \\ B(2) \\ B(3) \end{bmatrix} \qquad (4.1)$$

where:-

$$A(1) = \left[-\frac{1}{\rho C_p} \left\{ 1 + \frac{T}{z} \left[\frac{\partial z}{\partial T}\right]_p \right\} - \frac{\Delta t u W}{2\rho C_p A P} \right]_Q$$

$$A(2) = \left[1 + \frac{\Delta t u W}{2\rho C_p A T} \right]_Q$$

$$A(3) = \left[-\frac{\Delta t W}{\rho C_p A} \right]_Q$$

$$A(4) = \left[\frac{1}{\rho a_s} - \frac{a_s \Delta t}{\rho C_p T} \left\{ 1 + \frac{T}{z} \left[\frac{\partial z}{\partial T}\right]_p \right\} \frac{Wu}{2AP} + \frac{W\Delta t}{2A\rho P} \right]_R$$

$$A(5) = \left\{ \frac{a_s \Delta t W u}{2\rho C_p A T^2} \left\{ 1 + \frac{T}{z} \left[\frac{\partial z}{\partial T}\right]_p \right\} - \frac{W\Delta t}{2A\rho T} \right]_R$$

$$A(6) = \left[1 - \frac{a_s \Delta t W}{\rho C_p T A} \left\{ 1 + \frac{T}{z} \left[\frac{\partial z}{\partial T}\right]_p \right\} + \frac{W\Delta t}{A\rho u} \right]_R$$

$$A(7) = \left[-\frac{1}{\rho a_s} + \frac{a_s \Delta t W u}{2\rho C_p T A P} \left\{ 1 + \frac{T}{z} \left[\frac{\partial z}{\partial T}\right]_p \right\} + \frac{W\Delta t}{2A\rho P} \right]_S$$

$$A(8) = \left[-\frac{a_s \Delta t W u}{2\rho C_p A T^2} \left\{ 1 + \frac{T}{z} \left[\frac{\partial z}{\partial T}\right]_p \right\} - \frac{W\Delta t}{2A\rho T} \right]_S$$

$$A(9) = \left[1 + \frac{a_s \Delta t W}{\rho C_p A T} \left\{ 1 + \frac{T}{z} \left[\frac{\partial z}{\partial T}\right]_p \right\} + \frac{W\Delta t}{A\rho u} \right]_S$$

$$B_{(1)} = \left[- \frac{P}{\rho C_p} \left\{ 1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p \right\} + T + \frac{\Delta t \Omega}{\rho C_p A} \right]_Q$$

$$B_{(2)} = \left[\frac{P}{\rho a_s} + u + \frac{a_s \Delta t}{\rho C_p T} \left\{ 1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p \right\} \frac{\Omega}{A} - g \cdot \Delta t \sin\theta \right]_R$$

$$B_{(3)} = \left[- \frac{P}{\rho a_s} + u - \frac{a_s \Delta t}{\rho C_p T} \left\{ 1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p \right\} \frac{\Omega}{A} - g \cdot \Delta t \sin\theta \right]_S$$

The values at state 1 at the base of the characteristic were found by linearly interpolating between the grid points. Hence by solving these equations a first-order approximation was obtained for the predicted pressure, temperature and flow velocity.

Since the required stability and accuracy could not be achieved using this first-order approximation, this solution was used as an initial estimate in a second-order procedure. Although the exact procedure of this second-order model was dependent on the type of grid point being examined, in principle, new values for the variables at state 1 were found using quadratic interpolation. The coefficients in the characteristic equations were then calculated using these values. The coefficients were averaged with the previous state 1 coefficients and the results substituted back into the characteristic equations. By this method new values for the predicted pressure, temperature and flow velocity were obtained.

STANDARD INTERNAL POINT

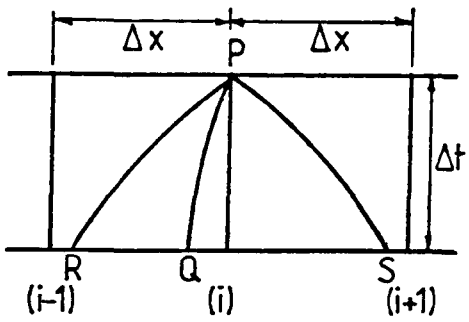
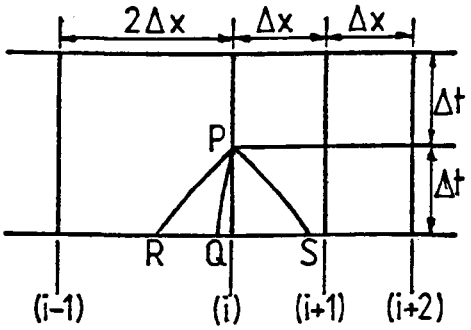


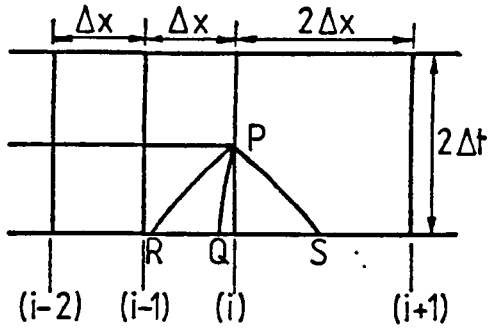
Fig. 4.2i

INTERNAL POINT BETWEEN TWO DIFFERENT GRID SIZES



Upstream of the Break

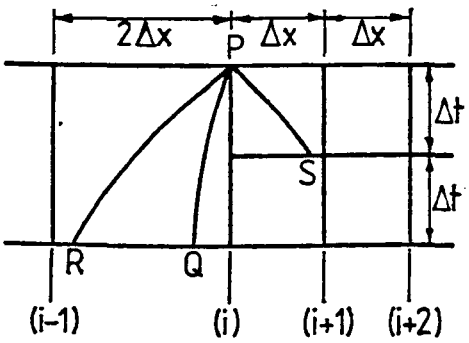
Fig. 4.2ii



Downstream of the Break

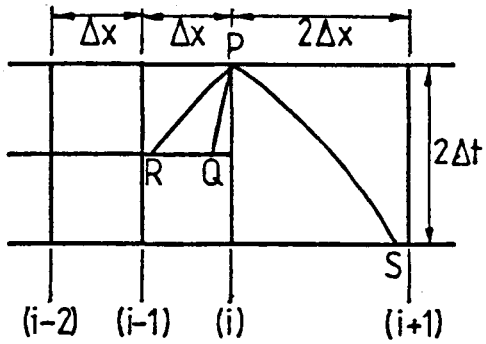
Fig. 4.2iv

INTERNAL POINT LINKING TWO DIFFERENT GRID SIZES



Upstream of the Break

Fig. 4.2iii



Downstream of the Break

Fig. 4.2v

Figure 4.2. **Different Internal Grid Point Configurations**

4.3. SPECIFIC SOLUTIONS FOR VARIOUS GRID POINT CONFIGURATIONS

There were a number of different grid point configurations for which the characteristic equations had to be solved. These configurations are detailed in Figure 4.2.

4.3.1. Internal point with equi-distant adjacent grid points (Fig.4.2(i))

FIRST ORDER APPROXIMATION:-

The positions of Q, R and S were defined thus:

$$posQ = 2\Delta t/({}^{1}/u(i) + {}^{1}/u(i-1)) \text{ assuming } u(i) \text{ or } u(i-1) \text{ do not equal } 0$$

$$posR = 2\Delta t/({}^{1}/(u(i) + a_s(i)) + {}^{1}/(u(i-1) + a_s(i-1)))$$

$$posS = 2\Delta t/(1/(a_s(i) - u(i)) + 1/(a_s(i+1) - u(i+1)))$$

where $u(i)$ = velocity at time t at point i in the pipe

and $a_s(i)$ = isentropic wavespeed at time t at point i in the pipe.

The values of the variables P, T, u, z $(\partial z/\partial T)_p$, ρ, a_s, Ω and W at points Q, R and S were then calculated by linear interpolation:-

$$P_Q = (posQ/\Delta x) \cdot P(i-1) + (1 - posQ/\Delta x) \cdot P(i)$$

$$P_R = (posR/\Delta x) \cdot P(i-1) + (1 - posR/\Delta x) \cdot P(i)$$

$$P_S = (posS/\Delta x) \cdot P(i+1) + (1 - posS/\Delta x) \cdot P(i)$$

where P_Q = value of P at the base of the path line (point Q)

P_R = value of P at the base of the C^+ characteristic (point R)

P_S = value of P at the base of the C^- characteristic (point S)

P(i) = value of pressure P at time t at point (i) in the pipe.

(Similarly for the other variables listed).

These values were then used in the general solution defined by equation (4.1) and first-order approximations were obtained for P, T and u at time $(t + \Delta t)$ (i.e. at state 2).

SECOND ORDER PROCEDURE:-

Using the predicted values of the variables P, T, and u obtained from the first-order approximations, predicted values for the variables z, $(\partial z/\partial T)_p$, ρ, a_s, Ω and W were calculated for point P (at time, $t + \Delta t$).

The new positions of points Q, R and S were then defined as:-

$posQ = 2\Delta t/(1/u_Q + 1/u_2(i))$ assuming u_Q and $u_2(i)$ do not equal zero.

$$posR = 2\Delta t/(1/(u_R + a_{sR}) + 1/(u_2(i) + a_{s2}(i)))$$

$$posS = 2\Delta t/(1/(a_{sS} - u_S) + 1/(a_{s2}(i) - u_2(i)))$$

where u_Q = value of u at the base of the path line

u_R = value of u at the base of the C^+ characteristic

u_S = value of u at the base of the C^- characteristic

a_{sR} = value of a_s at the base of the C^+ characteristic

a_{sS} = value of a_s at the base of the C^- characteristic

$u_2(i)$ = predicted value of u obtained from the first-order approximation

$a_{s2}(i)$= predicted value of a_s obtained from the first-order approximation

Taylor's theorem was then used to obtain a formula for quadratic interpolation so that new values of P, T, u, z, $(\partial z/\partial T)_p$, ρ, a_s, Ω and W could be calculated for points Q, R and S (the bases of the characteristics). Full derivation of the formulae for quadratic interpolation is given in Appendix III.

$$P_Q = P(i) - \frac{posQ}{2\Delta x}\left\{P(i+1) - P(i-1)\right\} + \frac{(posQ)^2}{2\Delta x^2}\left\{P(i+1)+P(i-1)-2P(i)\right\}$$

$$P_R = P(i) - \frac{posR}{2\Delta x}\left\{P(i+1) - P(i-1)\right\} + \frac{(posR)^2}{2\Delta x^2}\left\{P(i+1)+P(i-1)-2P(i)\right\}$$

$$P_S = P(i) + \frac{posS}{2\Delta x}\left\{P(i+1) - P(i-1)\right\} + \frac{(posS)^2}{2\Delta x^2}\left\{P(i+1)+P(i-1)-2P(i)\right\}$$

(Similarly for the other variables listed).

These values were then averaged with the predicted values obtained from the first-order approximation and the results used as coefficients in the difference form of the characteristic equations. Since the second-order approximation for W used in the first-order method was no longer required, the final values for substituting into equation (4.1) were:

$$A_{(1)} = \left[\left[\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]\right\}/\rho\right]_Q + \left[\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_P\right\}/\rho\right]_{P'}\right]/2C_P$$

$$A_{(2)} = 1$$

$$A_{(3)} = 0$$

$$A_{(4)} = \left[\left[1/\rho a_s\right]_R + \left[1/\rho a_s\right]_{P'}\right]/2$$

$$A_{(5)} = 0$$

$$A_{(6)} = 1$$

$$A_{(7)} = \left[\left[-1/\rho a_s\right]_S + \left[-1/\rho a_s\right]_{P'}\right]/2$$

$$A_{(8)} = 0$$

$$A_{(9)} = 1$$

$$B_{(1)} = A_{(1)} \cdot P_Q + T_Q + \frac{\Delta t}{2C_p A}\left[\left[\frac{\Omega + Wu}{\rho}\right]_Q + \left[\frac{\Omega + Wu}{\rho}\right]_{P'}\right]$$

$$B_{(2)} = A_{(4)} \cdot P_R + u_R + \frac{\Delta t}{2C_p A}\left[\left[\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p\right\}\left[\frac{\Omega + Wu}{\rho T}\right] \cdot a_s\right]_R\right.$$

$$\left. + \left[\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p\right\}\left[\frac{\Omega + Wu}{\rho T}\right] a_s\right]_{P'}\right] - \frac{\Delta t}{2A}\left[\left[\frac{W}{\rho}\right]_R + \left[\frac{W}{\rho}\right]_{P'}\right] - g \cdot \Delta t \sin\theta$$

$$B_{(3)} = A_{(7)} \cdot P_S + u_S - \frac{\Delta t}{2C_p A}\left[\left[\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p\right\}\left[\frac{\Omega + Wu}{\rho T}\right] \cdot a_s\right]_S\right.$$

$$\left. + \left[\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p\right\}\left[\frac{\Omega + Wu}{\rho T}\right] \cdot a_s\right]_{P'}\right] - \frac{\Delta t}{2A}\left[\left[\frac{W}{\rho}\right]_S + \left[\frac{W}{\rho}\right]_{P'}\right] - g \cdot \Delta t \sin\theta$$

Equation (4.1) was solved so that new predicted values for pressure, temperature and velocity were obtained for point P at time $t + \Delta t$. This second-order procedure was repeated, substituting in these new values until the required accuracy had been achieved.

If at the internal points downstream of the break flow reversal was occurring, the same method as outlined above would be used except that different values for posQ and p_Q etc., would be necessary. When the direction of flow is reversed in the pipeline the path characteristic is moved to the grid space containing the C^- characteristic as shown in Figure 4.3.

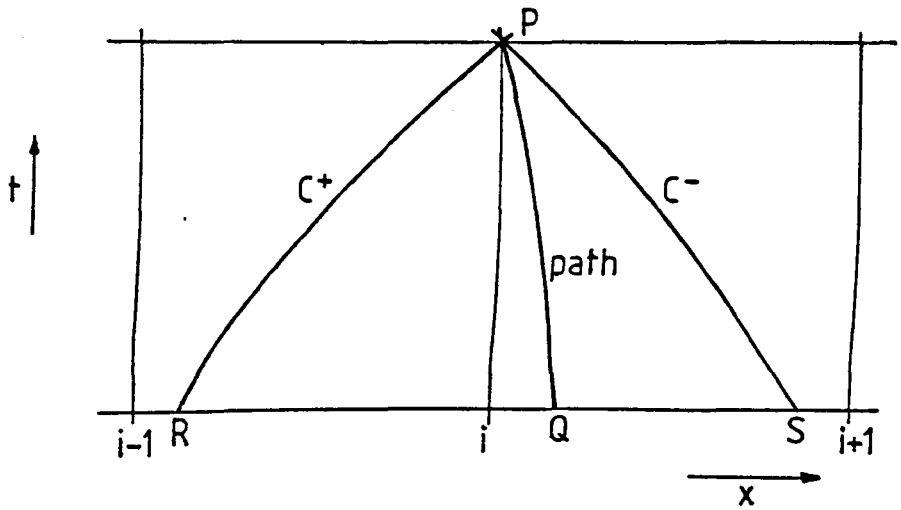


Figure 4.3. Effect of Flow Reversal on the Characteristics.

Therefore in this situation, for the first order approximation:

$posQ = 2\Delta t/(1/u(i) + 1/u(i+1))$ assuming neither u(i) nor u(i+1) equal zero

$$P_Q = \frac{posQ}{\Delta x} \cdot P(i+1) + (1 - \frac{posQ}{\Delta x}) \cdot P(i)$$

and for the second order approximation:

$$P_Q = P(i) + \frac{posQ}{2\Delta x}\left\{P(i+1)-P(i-1)\right\} + \frac{posQ^2}{2\Delta x^2}\left\{P(i+1) + P(i-1)-2P(i)\right\}$$

4.3.2. Internal Boundary Point between two different grid sizes

Figures 4.2(ii) and (iv) illustrate this type of grid point configuration. Examining firstly this point situated upstream of the break:-

FIRST ORDER APPROXIMATION:

The positions of Q, R and S were defined as for the standard internal points. The values of the variables at Q, R and S were then found by linear interpolation:

$$P_Q = (posQ/2\Delta x)\cdot P(i-1) + (1 - posQ/2\Delta x)\cdot P(i)$$

$$P_R = (posR/2\Delta x)\cdot P(i-1) + (1 - posR/2\Delta x)\cdot P(i)$$

$$P_S = (posS/\Delta x)\cdot P(i+1) + (1 - posS/\Delta x)\cdot P(i)$$

(Similarly for variables T, u, z, $(\partial z/\partial T)_p$, ρ, a_s, Ω and W).

These values were then used in the general solution defined by equation (4.1) to obtain first order approximations for P, T and u at time $t + \Delta t$.

SECOND ORDER PROCEDURE:

The same method as that of the standard internal points was followed and the new values for the positions for Q, R and S were calculated. The formulae for quadratic interpolation for non-equidistant adjacent points - derived using two separate Taylor's expansions - were then used to calculate new values of P, T, u, z, $(\partial z/\partial T)_p$, ρ, a_s, Ω and W at points Q, R and S. These formulae are given in equations (8), (9) and (10) of Appendix III. The new values for the variables were then used in the second order iterative method specified for standard internal points.

For this type of point situated downstream of the break, for the first order approximation:

$$P_Q = \frac{posQ}{\Delta x} \cdot P(i-1) + (1 - \frac{posQ}{\Delta x}) \cdot P(i)$$

$$P_R = \frac{posR}{\Delta x} \cdot P(i-1) + (1 - \frac{posR}{\Delta x}) \cdot P(i)$$

$$P_S = \frac{posS}{2\Delta x} \cdot P(i+1) + (1 - \frac{posS}{2\Delta x}) \cdot P(i)$$

and for the second order procedure equations (11), (12) and (13) of Appendix III were used. If flow reversal occurred then the variables were defined by:

$$P_Q = \frac{posQ}{2\Delta x} \cdot P(i+1) + (1 - \frac{posQ}{2\Delta x}) \cdot P(i) \quad \text{etc.}$$

in the first order approximation and equation (14) was used instead of equation (11) in Appendix III.

4.3.3. Internal Boundary Point linking two different grid sizes

This type of boundary point configuration is shown in Figures 4.2(iii) and (v). Since it is apparent that two time levels were necessary in order to predict values at a third level a more detailed diagram is given in Figure 4.4.

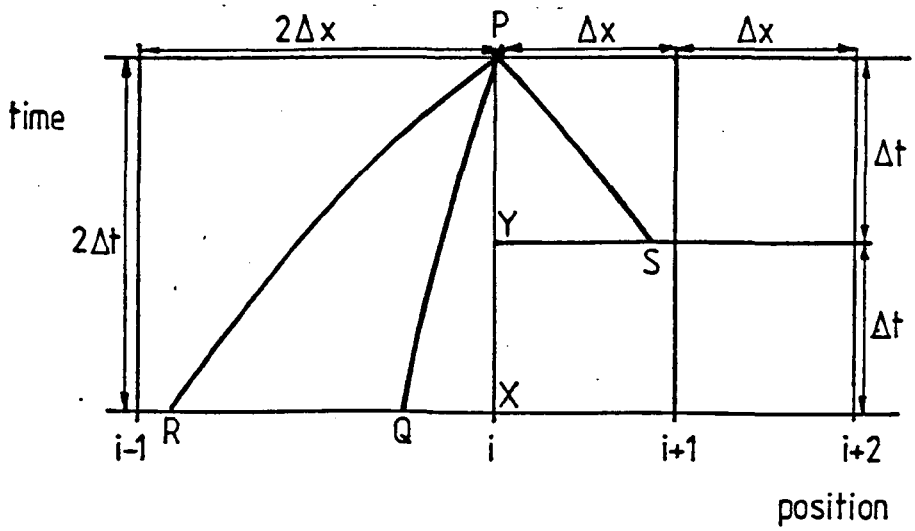


Figure 4.4. Grid point linking two different grid sizes

In the first order approximation the positions of Q, R and S were defined as:

$posQ = 4\Delta t/(1/u(i-1) + 1/u_X)$ assuming neither $u(i-1)$ nor u_X equal 0

$posR = 4\Delta t/(1/(u(i-1) + a_s(i-1)) + 1/(u_X + a_{sX}))$

$posS = 2\Delta t/(1/(a_s(i+1) - u(i+1)) + 1/(a_{sY} - u_Y))$

where subscripts X and Y denoted the values of the variable at positions X and Y.

The values of the variables at points Q, R and S were then determined using linear interpolation. For example:

$$P_Q = \frac{posQ}{2\Delta x} \cdot P(i-1) + (1 - \frac{posQ}{2\Delta x}) \cdot P_X$$

$$P_R = \frac{posR}{2\Delta x} \cdot P(i-1) + (1 - \frac{posR}{2\Delta x}) \cdot P_X$$

$$P_S = \frac{posS}{\Delta x} \cdot P(i+1) + (1 - \frac{posS}{\Delta x}) \cdot P_Y$$

These values were used in a modified form of the general solution given by equation (4.1). The modification to this equation was to replace Δt with $2\Delta t$ in the expressions for A(1), A(2), A(3), A(4), A(5), A(6), B(1) and B(2).

In the second order procedure the positions of Q, R and S were defined thus:

$posQ = 4\Delta t/(1/u_Q + 1/u_2(i))$ assuming neither u_Q nor $u_2(i)$ equal 0

$posR = 4\Delta t/(1/(u_R + a_{sR}) + 1/(u_2(i) + a_{s2}(i)))$

$posS = 2\Delta t/(1/(a_{sS} - u_S) + 1/(a_{s2}(i) - u_2(i)))$

and the new values for the variables P, T, u, z, $(\partial z/\partial T)_p$, ρ, a_s, Ω and W were calculated using the following equations:

$$P_Q = P(i-1) + \frac{(2\Delta x - posQ)}{4\Delta x}\left\{P_X - P(i-2)\right\} + \frac{(2\Delta x - posQ)^2}{8\Delta x^2}\left\{P_X + P(i-2) - 2P(i-1)\right\}$$

$$P_R = P(i-1) + \frac{(2\Delta x - posR)}{4\Delta x}\left\{P_X - P(i-2)\right\} + \frac{(2\Delta x - posR)^2}{8\Delta x^2}\left\{P_X + P(i-2) - 2P(i-1)\right\}$$

$$P_S = P(i+1) + \frac{(\Delta x - posS)}{2\Delta x}\left\{P(i+2) - P_Y\right\} + \frac{(\Delta x - posS)^2}{2\Delta x^2}\left\{P(i+2) + P_Y - 2P(i+1)\right\}$$

(These are equivalent to equations (15), (16), and (17) of Appendix III).

The second order procedure was again continued as outlined in Section 4.3.1.

For this type of point, positioned downstream of the break, the positions of Q, R and S for the first order approximation were defined by:

$$posQ = 2\Delta t/(1/u(i-1) + 1/u_Y) \text{ assuming neither } u(i-1) \text{ nor } u_Y \text{ are equal to 0}$$

$$posR = 2\Delta t/(1/(u(i-1) + a_s(i-1)) + 1/(u_Y + a_{sY}))$$

$$posS = 4\Delta t/(1/(a_s(i+1) - u(i+1)) + 1/(a_{sX} - u_X))$$

where subscripts X and Y denoted the value of the variable at the time levels X and Y shown in Figure 4.4; the values of the variables at Q, R and S were:

$$P_Q = \frac{posQ}{\Delta x} \cdot P(i-1) + (1 - \frac{posQ}{\Delta x}) \cdot P_Y$$

$$P_R = \frac{posR}{\Delta x} \cdot P(i-1) + (1 - \frac{posR}{\Delta x}) \cdot P_Y$$

$$P_S = \frac{posS}{2\Delta x} \cdot P(i+1) + (1 - \frac{posS}{2\Delta x}) \cdot P_X$$

The modification to the general solution in this instance was to replace Δt by $2\Delta t$ in the expressions for A(7), A(8), A(9) and B(3) of equation (4.1).

For the second order procedure, equations (18), (19) and (20) of Appendix III defined the variables at points Q, R and S with equation (21) of the same appendix being used in place of equation (18) if flow reversal occurred.

4.4. UPSTREAM BOUNDARY CONDITION

The only characteristic available at the upstream pipe end boundary condition was the C^- characteristic since it was assumed that the flow would always be in the positive direction in the section of pipe upstream of the break. Therefore two additional assumptions were required at the boundary in order to evaluate the three variables pressure, velocity and temperature. The two assumptions used in this analysis were constant pressure and constant mass flow rate (acceptable for this model since the upstream and downstream boundaries would have no effect on the section of pipe being examined).

Initially a first order approximation for the C^- characteristic was performed and the results from this were then used as initial values in the second order process (as with the previously described grid points).

The position of point S at the base of the C^- characteristic was calculated as before and a formula for quadratic interpolation was

derived using a Taylor expansion about the point i = 2. New values could then be calculated for the variables P, u, T, z, $(\partial z/\partial T)_p$, ρ, a_s, Ω and W at the point S. For example:

$$P_S = P((2) - (\Delta x - posS))$$

$$= P(2) - \frac{(\Delta x - posS)}{2\Delta x}\left\{P(3) - P(1)\right\} + \frac{(\Delta x - posS)^2}{2\Delta x^2}\left\{P(3)+P(1)-2P(2)\right\}$$

Using these values a second order approximation for the C^- equation was calculated and by implementing the assumptions of constant pressure and mass flow rate, the values for P, u and T were deduced for the next time level.

4.5. DOWNSTREAM BOUNDARY CONDITION

At the downstream end of the pipe it was decided for convenience to simulate a non-return valve to prevent flow reversal occurring in any pipes adjoining the test section that was being modelled. It was also assumed that this boundary was at constant temperature. With this additional assumption it was possible to calculate values for the variables P, u and T at a new time level.

As before, the initial values for a second-order procedure were calculated from a first-order approximation for the C^+ and path characteristics. The second-order approximation was then performed, using a Taylor expansion about the point adjacent to the boundary,

to calculate values for the variables at points Q and R (shown in Figure 4.5).

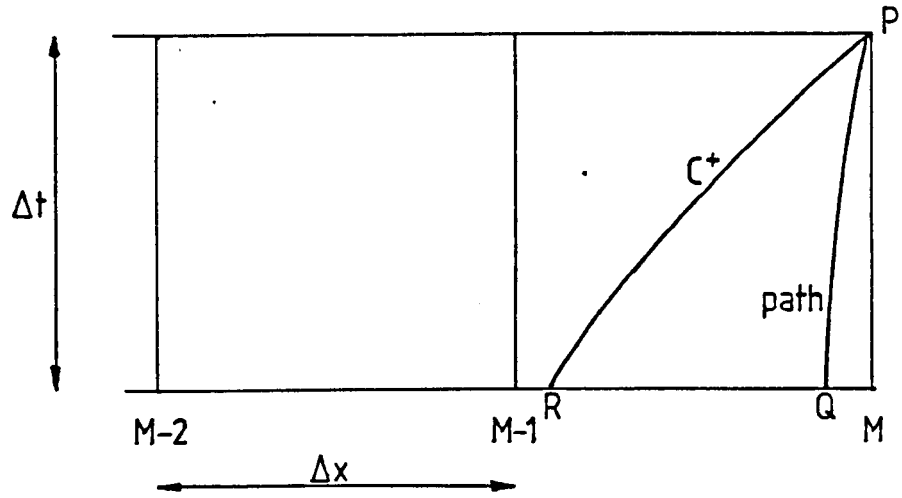


Figure 4.5. Downstream Boundary Condition

4.6. BREAK BOUNDARY CONDITION

An approximation to the situation of a linebreak occurring in a pipe is the sudden rupturing of a shock tube diaphragm separating two areas of different pressure. The main difference between these two situations is that in a gas pipeline there would be an initial flow velocity whereas the flow is initially stationary in the shock tube model. However, when considering high pressure pipelines, the effect of the initial flow velocity can be assumed to be negligible for the purpose of modelling the linebreak.

Kobes [1910] and Aschenbrenner [1937] examined the effect of suddenly removing a slide separating regions of high and low pressure in a pipe. They envisaged that the pressure at the

junction between the two regions would immediately change to a new value which was termed the 'equalization pressure'. Compression and rarefaction waves would propagate into the low and high pressure regions respectively, and the gas in the high pressure area would start to flow into the low pressure zone. A value for the equalization pressure may be determined by equating the particle velocities associated with the compression and rarefaction waves.

The particle velocity of a rarefaction wave, assuming isentropic conditions, (as derived by Earnshaw [1860]) is given by:

$$u = \frac{2}{\gamma - 1} a_o \left\{ 1 - \left[\frac{P}{P_o}\right]^{\frac{\gamma-1}{2\gamma}} \right\} \qquad (4.2)$$

where P_o = pressure in the high pressure zone prior to rupture

a_o = isentropic wavespeed prior to rupture $\left\{= \sqrt{\frac{\gamma P_o}{\rho_o}}\right\}$

A full derivation of this equation is given in Appendix IV.

The particle velocity of a steep-fronted compression wave (as derived by Bannister and Mucklow [1948] is expressed as:

$$u' = \frac{a_{AT} \left\{ \left[\frac{P}{P_{AT}}\right] - 1 \right\}}{\sqrt{\frac{\gamma}{2} \left\{ (\gamma + 1) \left[\frac{P}{P_{AT}}\right] + (\gamma - 1) \right\}}} \qquad (4.3)$$

where P_{AT} = pressure in the low pressure zone prior to rupture

a_{AT} = isentropic wavespeed in the low pressure zone prior to rupture $\left[= \sqrt{\frac{\gamma P_{AT}}{\rho_{AT}}}\right]$

At the junction between the two different pressure regions, the particle velocity of the rarefaction wave would be equal to that of the compression wave.

Therefore:

$$u = u'$$

$$\frac{2}{\gamma - 1} a_o \left\{1 - \left[\frac{P_e}{P_o}\right]^{\frac{\gamma-1}{2\gamma}}\right\} = a_{AT} \left\{\left[\frac{P_e}{P_{AT}}\right] - 1\right\} / \sqrt{\left\{\frac{\gamma}{2}\left\{(\gamma+1)\left[\frac{P_e}{P_{AT}}\right]+(\gamma-1)\right\}\right\}} \qquad (4.4)$$

Provided that the initial pressures and wavespeeds in both pressure zones were known, together with the ratios of specific heats, the equalization pressure, P_e, can be found iteratively.

Bakhtar [1956] simplified equation (4.4) by assuming that for 'moderate' pressure ratios the particle velocity of a steep-fronted wave is approximately the same as that for a non-steep wave. Therefore:

$$\frac{2}{\gamma - 1} a_{AT} \left\{\left[\frac{P_e}{P_{AT}}\right]^{\frac{\gamma-1}{2\gamma}} - 1\right\} = \frac{2}{\gamma - 1} a_o \left\{1 - \left[\frac{P_e}{P_o}\right]^{\frac{\gamma-1}{2\gamma}}\right\} \qquad (4.5)$$

If the ratio of specific heats is the same either side of the diaphragm and if the temperatures are the same in each pressure region, equation (4.5) could be further simplified to:

$$P_e = \left\{ \frac{1}{2} \left[\frac{1}{P_o^{\frac{\gamma-1}{2\gamma}}} + \frac{1}{P_{AT}^{\frac{\gamma-1}{2\gamma}}} \right] \right\}^{\frac{2\gamma}{\gamma-1}}$$

However, in the case of a linebreak in a high pressure gas pipeline, the ratios of specific heats and the temperatures would not be equal in the high and low pressure regions (i.e. inside and outside the pipe). Furthermore, an iterative method would be required to calculate the value of the equalization pressure, and hence there is little advantage in using the simplified equation (4.5). Therefore this analysis defined the equalization pressure using equation (4.4).

It was first necessary to test whether choked flow would occur at the break. The critical value for the equalization pressure may be defined as:

$$P_e^* = \left[\frac{2}{\gamma + 1} \right]^{\frac{2\gamma}{\gamma-1}} \cdot P_o \qquad (4.6)$$

where P_o is the pressure in the pipe at the break point prior to rupture

and γ is the ratio of specific heats of the gas in the pipe.

If the equalization pressure defined by equation (4.4) is less than or equal to the critical value defined by equation (4.6), then choking would occur at the break. In order to test for this condition, the critical value for the equalization pressure was substituted into both sides of equation (4.4). If it was found that the left-hand side of the equation was less than or equal to the right-hand side, then choked flow would occur. In this case the pressure at the break point at the instant of rupture would theoretically immediately fall to the critical equalization pressure as defined by equation (4.6).

If, however, the left-hand side of equation (4.4) was found to be greater than the right, then the equalization pressure would be found by an iterative process of the following form:

1. Using the value of the critical equalization pressure, the left-hand side of equation (4.4) is determined.
2. By re-arranging the right-hand side of equation (4.4) and by assuming that the pipe is surrounded by air (γ = 1.4), the following expression is obtained:

$$25\left[\frac{P_e}{P_{AT}}\right]^2 - \left(50 + 42\,\frac{(LHS)^2}{a_{AT}^2}\right)\left[\frac{P_e}{P_{AT}}\right] + \left(25 - 7\,\frac{(LHS)^2}{a_{AT}^2}\right) = 0 \qquad (4.7)$$

where (LHS) is the value obtained from the left-hand side of equation (4.4).

3. By solving the quadratic of equation (4.7) a new value for the equalization pressure is found.

4. This value for the equalization pressure is then substituted into the left-hand side of equation (4.4) in order to obtain a new value for (LHS).

Steps 3 and 4 were repeated until the required accuracy for the equalization pressure was obtained.

In this case when rupture occurs, the pressure at the break point will fall to this equalization pressure and then remain at that value. Realistically, however, the drop in pressure to either the critical or the iterative value for the equalization pressure cannot happen instantaneously, since such a pressure drop must occur over a finite period of time. Also, in the numerical model such a discontinuity in the pressure would cause severe instabilities. Therefore the pressure drop was modelled over a number of time steps. It was realized that a linear pressure drop would be an inaccurate model since this would create a discontinuity in the pressure gradient. This can be seen in Figure 4.6. Therefore various polynomial expressions were examined. These took the form of:

$$P(j) = (P_o - P_{eq})(1 - \frac{j}{64m})^n + P_{eq}$$

where P_o = pressure prior to rupture

P_{eq} = equalization pressure

j = number of small time steps after break has occurred

m = number of large time steps in the x-t grid over which the pressure drop is being modelled

n = index (an even integer) determining the severity of the pressure drop (as shown in Figure 4.6).

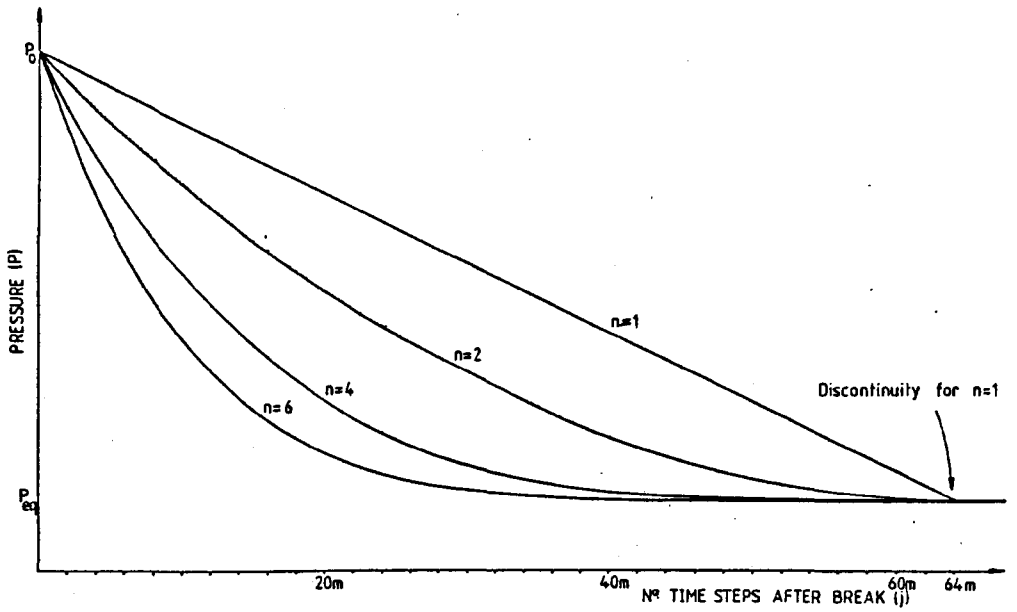


Figure 4.6. Pressure Drop at the Break

With the pressure being determined in this manner, the flow velocity and temperature could both then be calculated using the method of characteristics (as for the upstream and downstream boundary conditions).

CHAPTER 5

COMPUTER MODEL

5.1. INTRODUCTION

For a number of years there have been computer programs available for modelling certain transient gas flow situations, such as isothermal network analysis (Guy [1967], Heath and Blunt [1969], Bender [1979], Goldfinch [1984]), and loss of coolant accidents (Elliott [1968], Moore and Rettig [1973], Brittain and Fayers [1976], Banerjee and Hancox [1978]). Programs which can account for frictional and heat transfer effects have also been developed for modelling rapid transients (Van Deen and Reintsema [1983], Issa [1970]). However in order to solve the characteristic equations using the numerical method detailed in the previous chapter, allowing for the possibility of flow reversal downstream of a break as well as handling grid size reduction in the vicinity of the break, a new program has been developed.

This program performs a transient analysis on a given shock tube or single pipe, producing numerical output for the pressure, flow velocity and temperature at each time step. A second program then converts the required section of this numerical output into graphical form.

Both programs were written in FORTRAN 77 for use on a Gould PN 9005 mainframe computer.

5.2. TRANSIENT ANALYSIS PROGRAM

After prompting for the required input data, the program constructs the grid shown in Figure 4.1. pertaining to the pipe being examined. An isothermal steady flow analysis is then conducted along the length of the pipe to obtain initial values at each of the grid points. From these initial values an analysis using the method of characteristics is performed modelling the situation prior to linebreak or diaphragm rupture. This produces more accurate initial data from which the events following the pipe break are modelled using the transient analysis. The results are printed out after each major time step and a results data file created for use with the graphics program.

5.2.1. Main Program

The main program initially prompts for the gas and system data detailed on the data sheet (Figure 5.1). From the values for the pipe lengths and the required grid size near the break, the program forms the grid. It then calls up subroutines STEAD1 and STEAD2 to perform isothermal steady flow analyses on the pipes upstream and downstream of the position of the break. This produces initial values of pressure, temperature and flow velocity at every grid point.

The maximum time step that would not exceed the stability criterion is then calculated so that the required time step and run time may be entered.

DATA SHEET NUMBER ___

Gas Data

Specific heat	C_p	(kJ/kg K)	_____
Gas Constant	R	(kJ/kg K)	_____
Critical Temperature	T_c	('C)	_____
Critical Pressure	P_c	(kPa)	_____

Pipe Data

Diameter of Pipe	(m)	_____
Angle of Inclined Pipe	(degrees)	_____
Length of Pipe upstream of break point	(m)	_____
Length of Pipe downstream of break point	(m)	_____
Required Grid Size near the break	(m)	_____
Darcy friction factor		_____
Stanton Number		_____
Wall Temperature	('C)	_____
Atmospheric Temperature	('C)	_____
Atmospheric Pressure	(kPa)	_____
Initial Temperature along Pipe	('C)	_____
Initial Pressure at upstream end of Pipe	(kPa)	_____
Mass flow rate through Pipe	(kg/s)	_____

Run Data

Length of time step required	(msec)	_____
Total Run Time required	(secs)	_____

(Break details: Index n =

No.of steps x =)

Figure 5.1. Data Preparation Sheet

From the pressure, temperature and flow velocity values at each point, the program calculates the density of the gas, the compressibility factor and its partial derivatives with respect to temperature and pressure, the frictional force and heat transfer and the isentropic and isothermal wavespeeds. It then calls up subroutines SUB1 to SUB6 and BREAK1 and BREAK2 to calculate new values of pressure, temperature and flow velocity at the grid points marked (1) in Figure 5.2. The program can then calculate the density of the gas, etc., for these points and by calling up the subroutines again will produce new values of pressure, temperature and flow velocity for the grid points marked (2) in Figure 5.2. This procedure is repeated until the 64 small time steps at the break have been completed. Subroutines SUBUP and DOWN1 then calculate the new values of pressure, temperature and flow velocity at the upstream and downstream pipe boundaries respectively, thus producing steady flow values at each of the grid points which have been calculated by the method of characteristics. The program then prints out these initial values at the specified grid points.

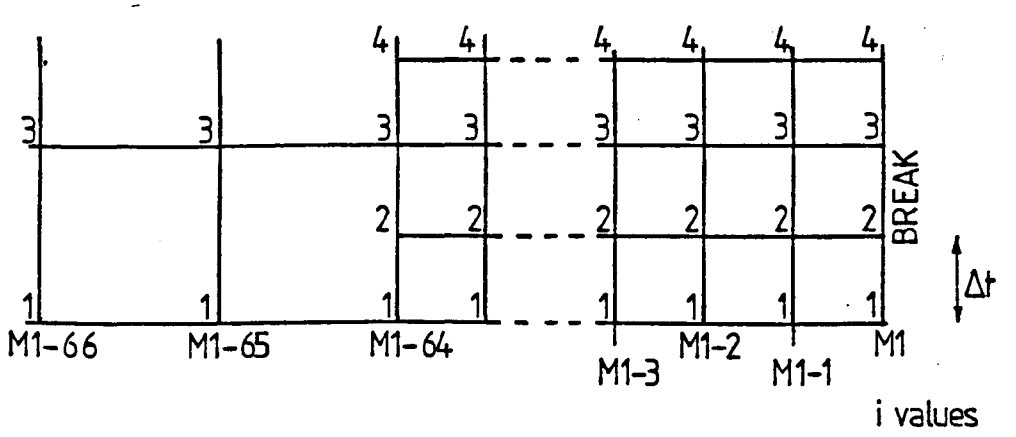


Figure 5.2. Marching Process of Calculation

At this point the program initiates the pipebreak. Having calculated the equalization pressure at the break from Earnshaw's theory and determined the number of time steps over which the pressure drop at the break occurs, the program calculates a new value for the pressure at the break point one time step after the rupture occurs. Subroutine BREAK3 is called up to calculate the temperature and flow velocity at the break point in the upstream section of pipe. Subroutine BREAK4 then calculates the temperature and velocity at the break in the downstream section of pipe. The new values of pressure, temperature and flow velocity at each of the internal points are calculated using subroutines SUB1 to SUB6 and the values at the pipe ends using SUBUP and DOWN1 as before.

The main program continues looping, printing out results after each major time step (equal to 64 time steps at the break), until the run time is reached. A full listing of the main program is given in Appendix V.

5.2.2. Subroutines STEAD1 and STEAD2

STEAD1 and STEAD2 calculate values of pressure, temperature and flow velocity at each point in the pipe (both upstream and downstream of the break point) assuming a steady isothermal flow. The equations used by these subroutines were derived from the linear momentum equation for steady one-dimensional flow:-

$$\Sigma F = \dot{m} (u_2 - u_1)$$

Applying this to the section of pipe shown in Figure 5.3:-

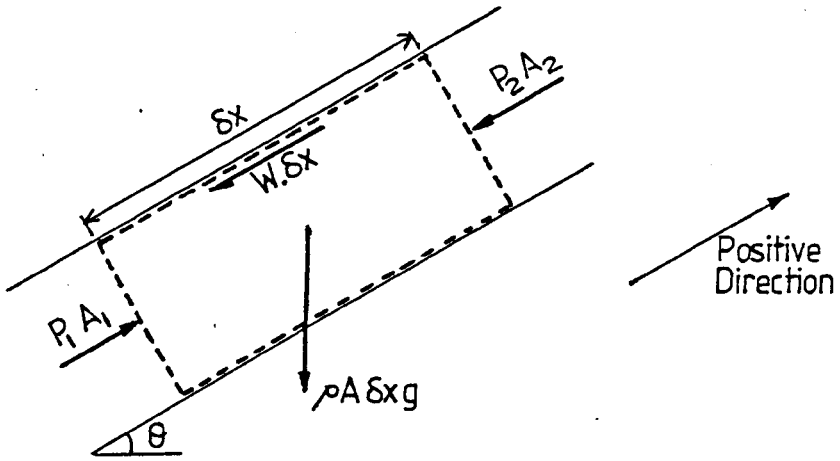


Figure 5.3. Linear Momentum Equation for Steady One-Dimensional Pipe Flow

$$(P_1 - P_2)A - W\delta x - \rho A\delta x g\sin\theta = \dot{m}\,(u_2 - u_1)$$

Dividing through by A and re-arranging:-

$$P_1 - P_2 + \frac{\dot{m}}{A}(u_1 - u_2) - \frac{\rho f u^2 \delta x}{2d} - \delta x \rho g \sin\theta = 0 \qquad (5.1)$$

But, from the equation of state assuming constant temperature and compressibility factor,

$$P_2 = \rho_2 RTZ$$

$$= \frac{\dot{m}}{u_2 A} RTZ \qquad \text{where } \dot{m} = \text{mass flow rate}$$

Substituting this into equation (5.1):-

$$\left\{P_1 - \frac{\dot{m}}{u_2} RTZ\right\} + \frac{\dot{m}}{A}(u_1 - u_2) - \frac{f\dot{m}\delta x}{2d}\left[\frac{u_1+u_2}{2}\right] - \left[\frac{\rho_1 + \dot{m}/u_2}{2}\right]\delta x g \sin\theta = 0$$

This may be re-written producing a quadratic equation in u_2.

$$\left\{\frac{f\dot{m}\delta x}{4d} + \frac{\dot{m}}{A}\right\} u_2^2 + \left\{\frac{f\dot{m}\delta x}{4d} u_1 + \frac{\rho_1}{2} \delta x g \sin\theta - P_1 - \frac{\dot{m}u_1}{A}\right\} u_2$$

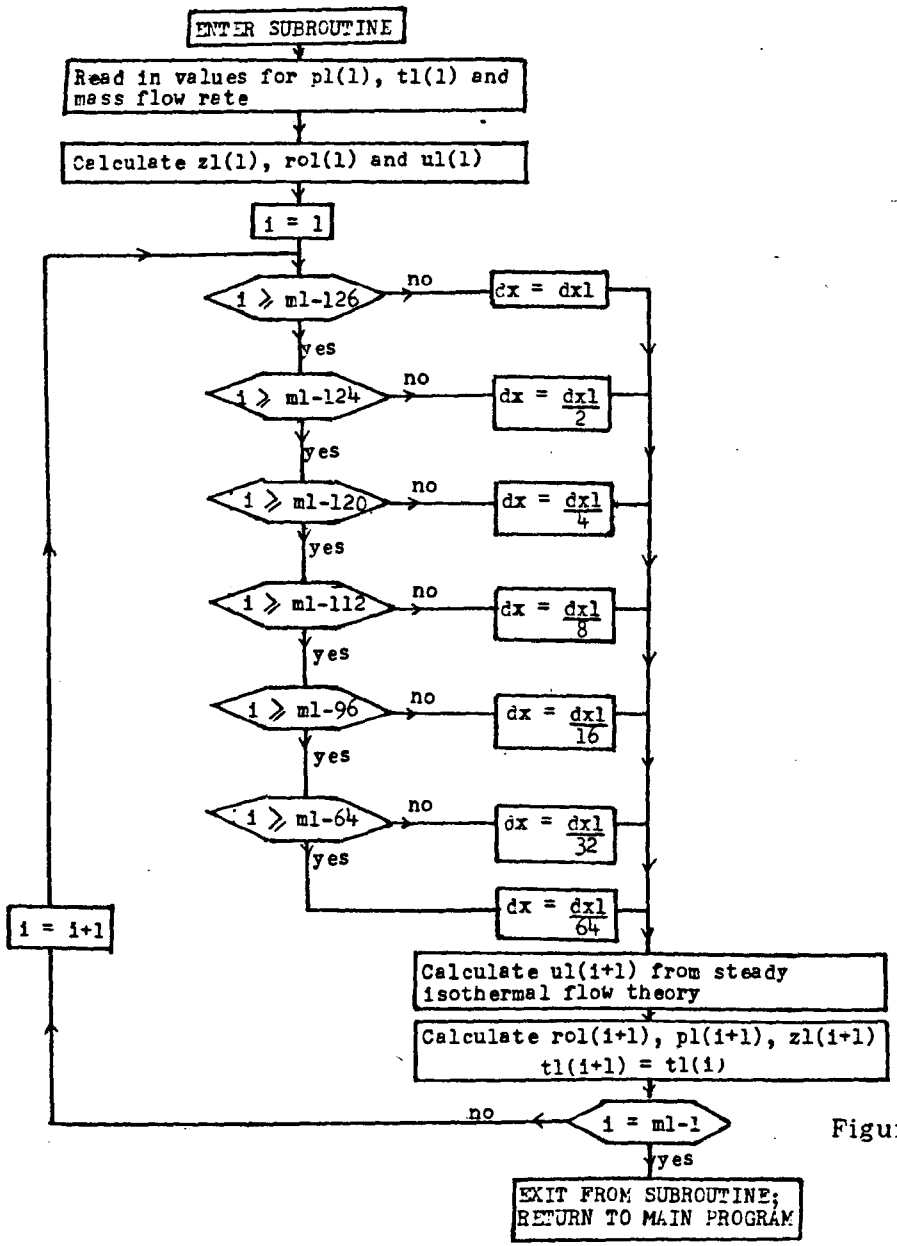
$$+ \left\{\frac{\dot{m}\delta x}{2} g \sin\theta + \dot{m}RTZ\right\} = 0$$

The subroutines STEAD1 and STEAD2 solve this quadratic and can then deduce the density and pressure at the next grid point (i.e. ρ_2 and P_2). By repeating this procedure, values can be calculated for each of the grid points shown in Figure 4.1 given the initial temperature of the gas, the pressure at the upstream end of the pipe and the mass flow rate. Flow diagrams of these subroutines are shown in Figure 5.4 and the subroutine listings are given in Appendix V.

5.2.3. Subroutines SUB1 to SUB6

Using the method of characteristics, these subroutines predict the values of pressure, temperature and flow velocity at the next time step for all the internal points. Six routines are required due to the different possible grid point configurations (Figure 4.2) and the possibility of flow reversal. Table 5.1 details the points for which each of these subroutines calculates new values.

The calculation procedures used by subroutines SUB1 to SUB6 are virtually identical differing only in the positioning of the base points of the characteristics (points Q, R and S). Subroutines SUB1, SUB2 and SUB5 operating on the section of pipe upstream from the



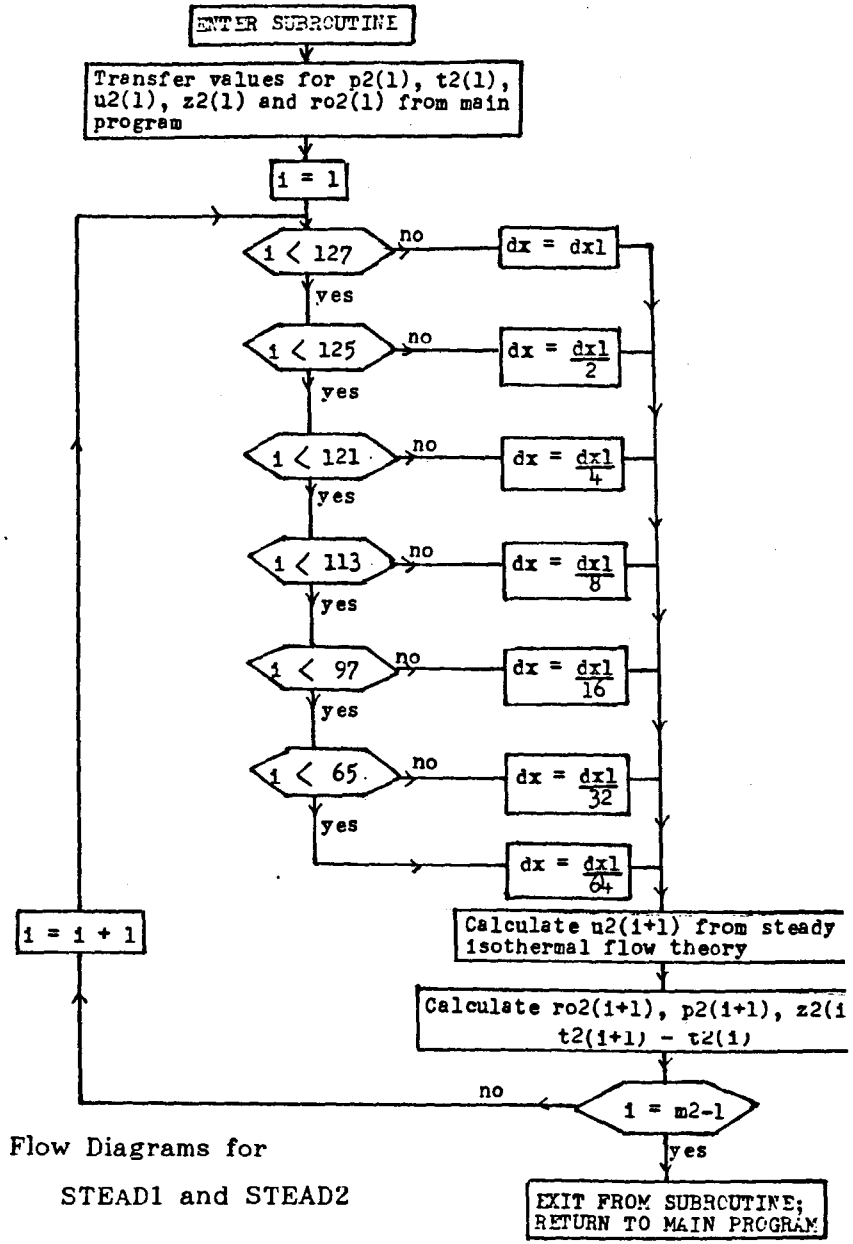


Figure 5.4. Flow Diagrams for STEAD1 and STEAD2

break follow the general procedure detailed in the flow diagram in Figure 5.5(a). For points downstream from the break, SUB3, SUB4 and SUB6 are used which determine the direction of flow prior to the numerical calculations detailed in Chapter 4. This is shown in the flow diagram in Figure 5.5(b).

Subroutine	Grid Point on which Subroutine Operates
SUB1	Normal internal points upstream of the break (Figure 4.2(i))
SUB2	Internal boundary points between different grid sizes upstream of the break (Figure 4.2(ii))
SUB3	Internal boundary points between different grid sizes downstream of the break (Figure 4.2(iv))
SUB4	Normal internal points downstream of the break
SUB5	Internal boundary points linking different grid sizes upstream of the break (Figure 4.2(iii))
SUB6	Internal boundary points linking different grid sizes downstream of the break (Figure 4.2(v)).

Table 5.1. Subroutines SUB1 to SUB6

5.2.4. Subroutines BREAK1 to BREAK4

These subroutines all calculate values of pressure, temperature and flow velocity at the position of the break. BREAK1 and BREAK2 are used to model the situation prior to rupture, and after rupture BREAK3 and BREAK4 model the situation immediately upstream and downstream of the break respectively. Examining each routine separately:-

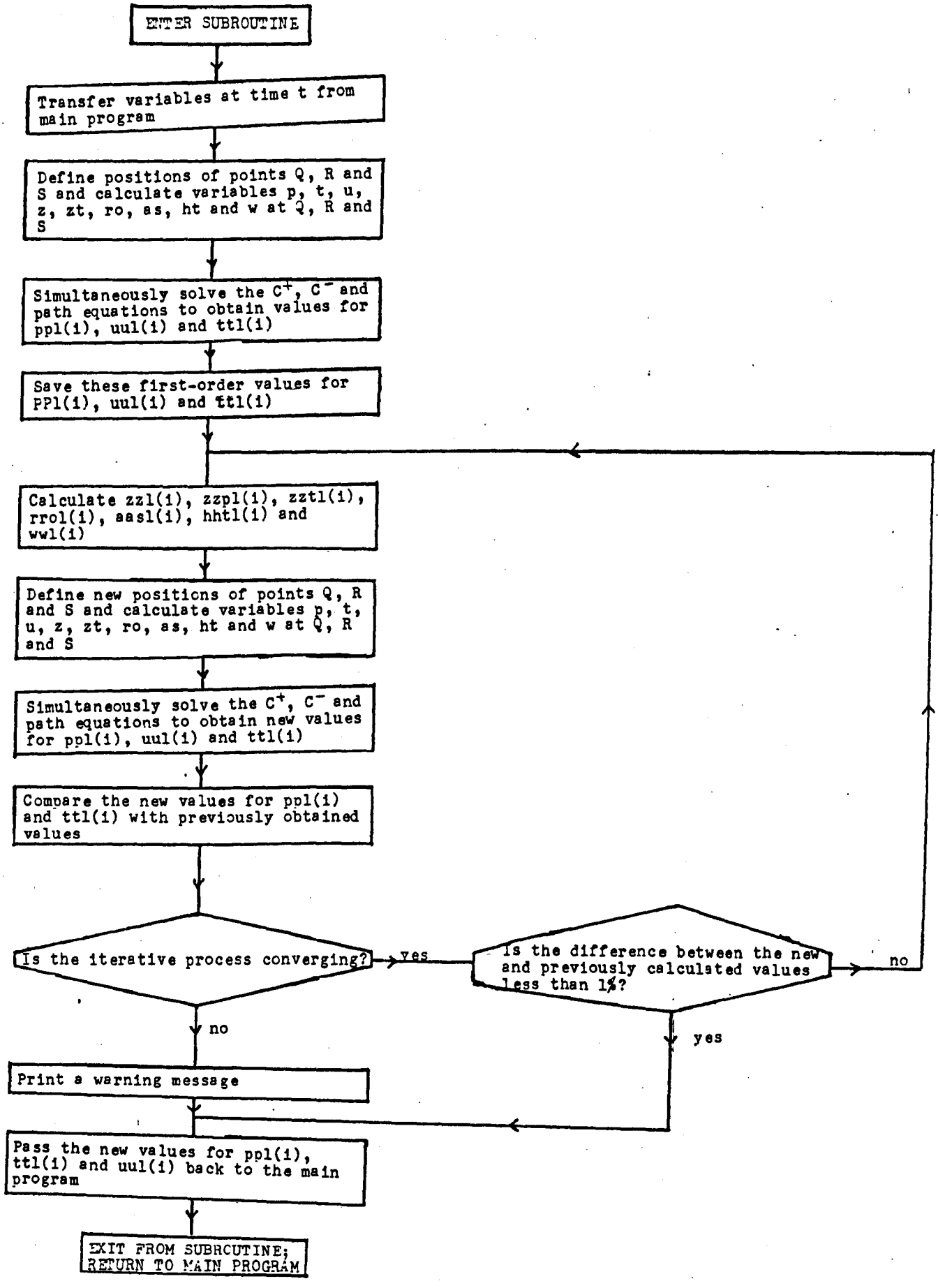


Figure 5.5(a). Flow Diagram for SUB1, SUB2 and SUB5

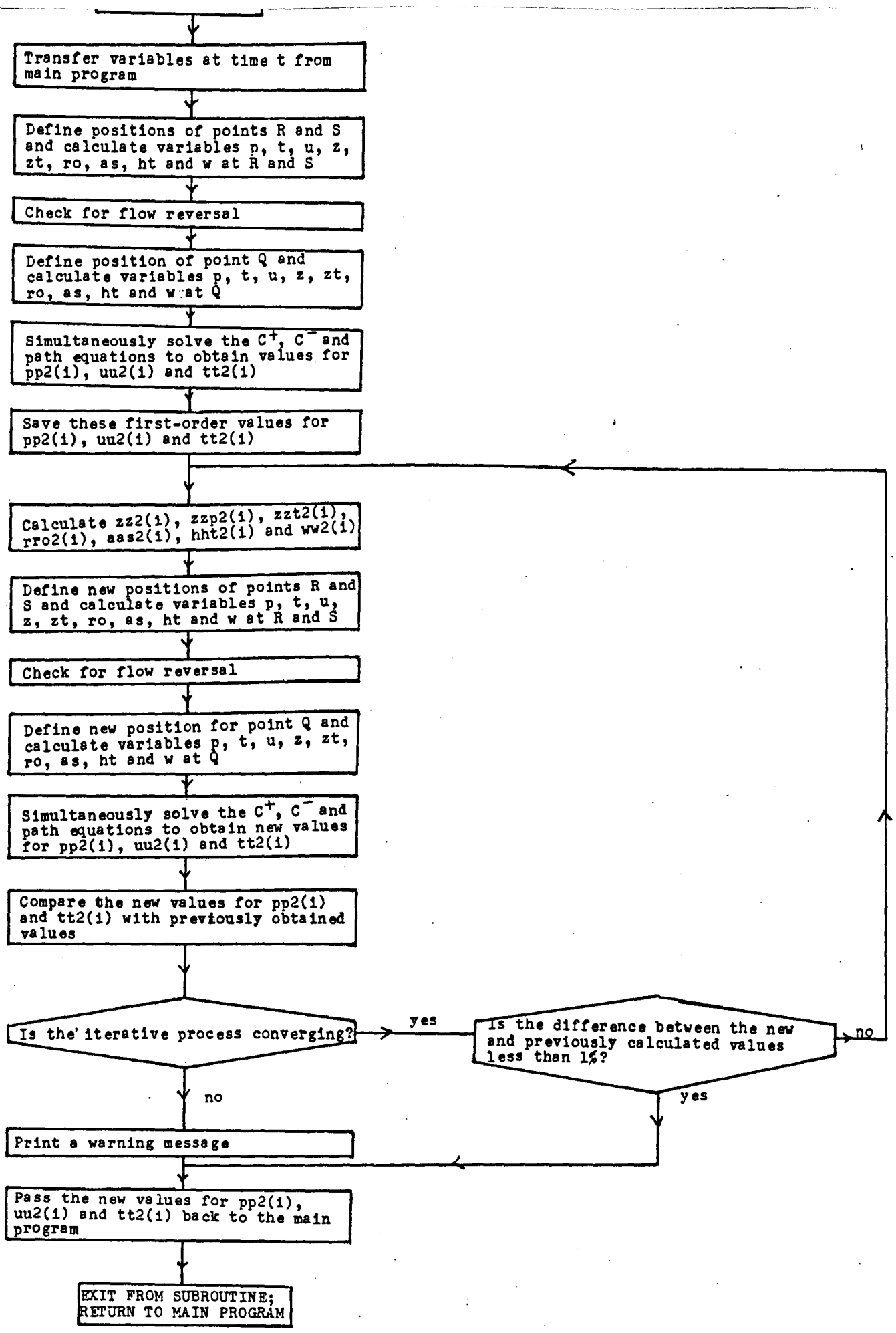


Figure 5.5(b). Flow Diagram for SUB3, SUB4 and SUB6

BREAK1

BREAK1 performs a method of characteristics analysis identical to that used for a normal internal point, using the last two points, $i = m_1 - 1$ and $i = m_1$ in pipe 1 (upstream of the break), and the first two points, $i = 1$ and $i = 2$ in pipe 2 (downstream of the break). Obviously, prior to rupture the points $i = m_1$ in pipe 1 and $i = 1$ in pipe 2 coincide. The method of characteristics calculations are therefore carried out on the section of grid shown in Figure 5.6.

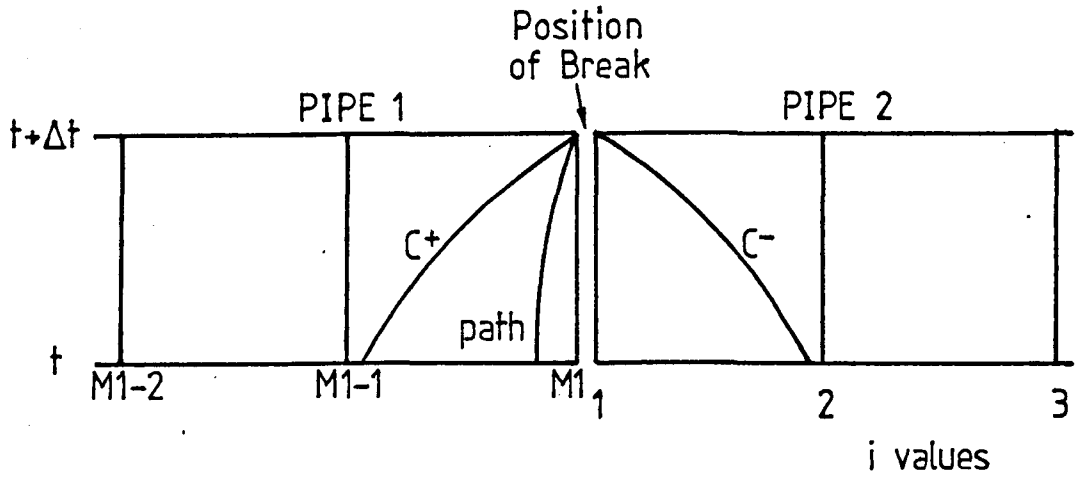


Figure 5.6. Break Point Prior to Rupture.

BREAK2

This subroutine simply defines the values at $i = 1$ in pipe 2 as being equal to the values calculated for $i = m_1$ in pipe 1 by subroutine BREAK1.

BREAK3

BREAK3 calculates the new values at the grid point i = m_1 in pipe 1 after a rupture has occurred. Referring to Figure 5.7, if the pressure at point i = m_1 at time t + Δt has been defined by the main program, the flow velocity and temperature may be calculated using the C^+ and path characteristics. Since the flow in pipe 1 will always be in the positive x-direction, there will always be a path line in this grid section.

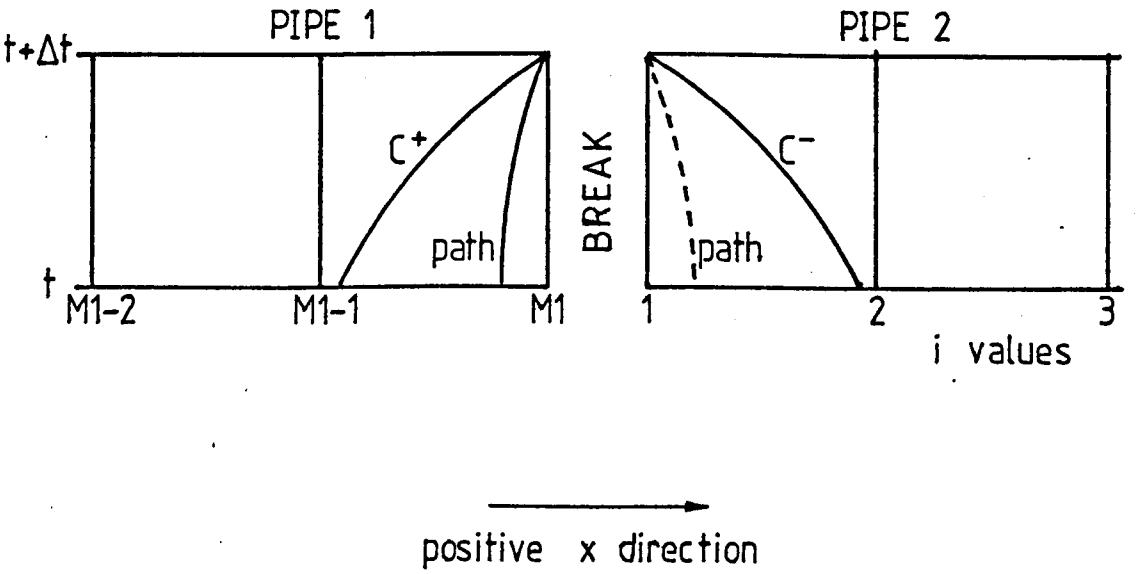


Figure 5.7. Break Point after Rupture.

BREAK4

BREAK4 initially ascertains whether flow reversal is occurring at point i = 1 in pipe 2. If the velocity at i = 1 at time t is less than zero then a path line is present as shown in Figure 5.7. Since the pressure at i = 1 has been defined by the main program, the solution of the C^- and path characteristics would determine the flow velocity and temperature at time t + Δt. If, however, the flow at i = 1 at

time t is greater than zero, only the C^- characteristic is present. A further assumption is therefore necessary and in this case it was assumed that the temperature at the point i = 1 in pipe 2 is equal to that at point i = m_1 in pipe 1. The justification for this is that since flow reversal would rapidly occur immediately downstream of a linebreak in a high pressure pipe, there would be insufficient time for the temperature prior to flow reversal to differ significantly from the temperature at the upstream side of the break. The subroutine can then solve the C^- characteristic using this value for the predicted temperature and hence obtain a value for the flow velocity at time t + Δt.

Flow diagrams for BREAK1, BREAK3 and BREAK4 are given in Figures 5.8(a), (b) and (c) respectively. The listings of each of these subroutines are in Appendix V.

5.2.5. Subroutines SUBUP and DOWN1

These two subroutines calculate the new values of pressure, temperature and flow velocity at the pipe ends (away from the break). At the end of pipe 1 (upstream from the break) there will only be the C^- characteristic and at the end of pipe 2 there is, at the most, only the C^+ and path characteristics. Therefore at both pipe ends certain assumptions are necessary in order to predict the flow conditions at time t + Δt. SUBUP models the upstream boundary in pipe 1 assuming a constant pressure and a constant mass flow

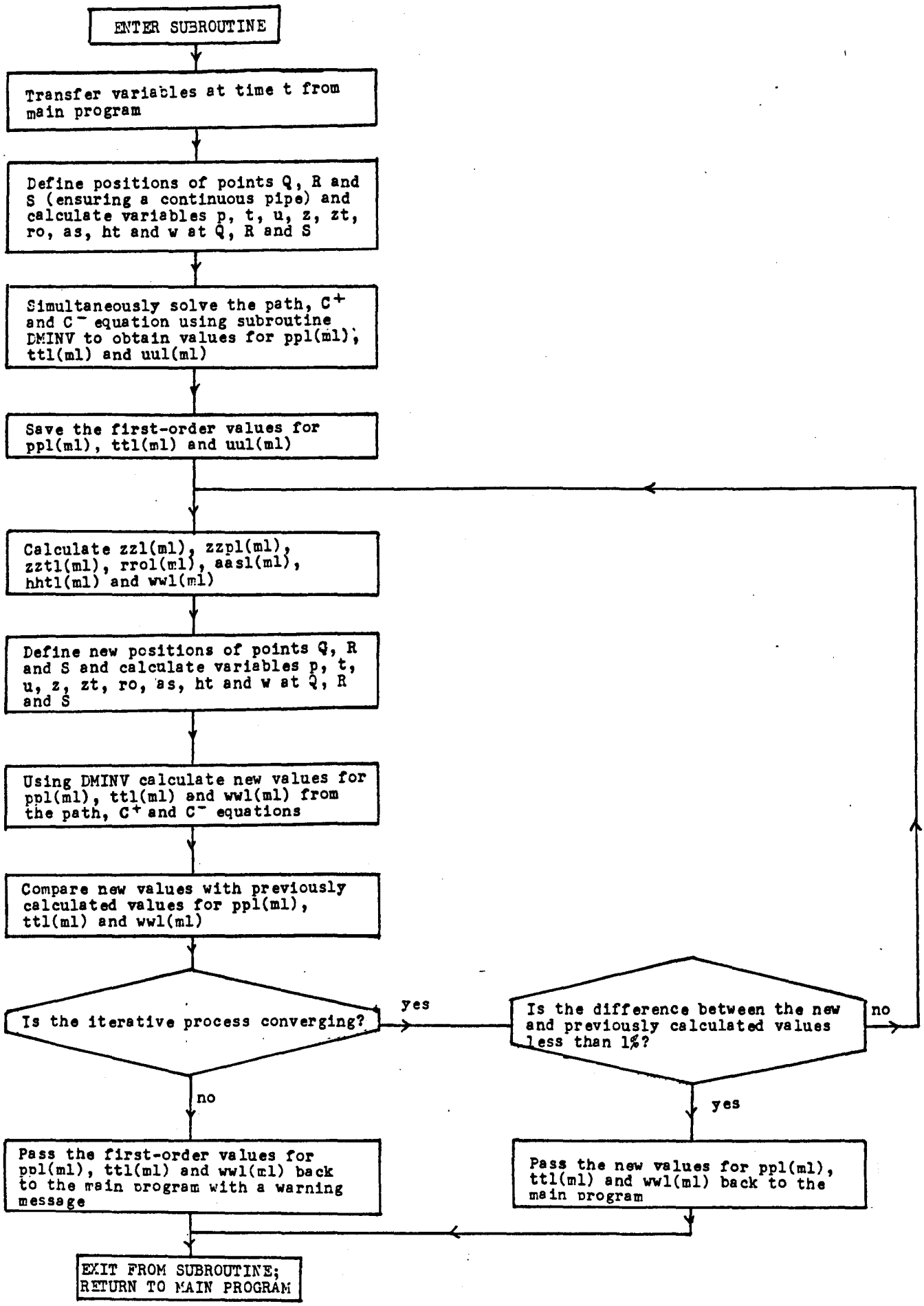


Figure 5.8(a) Flow Diagram for BREAK1

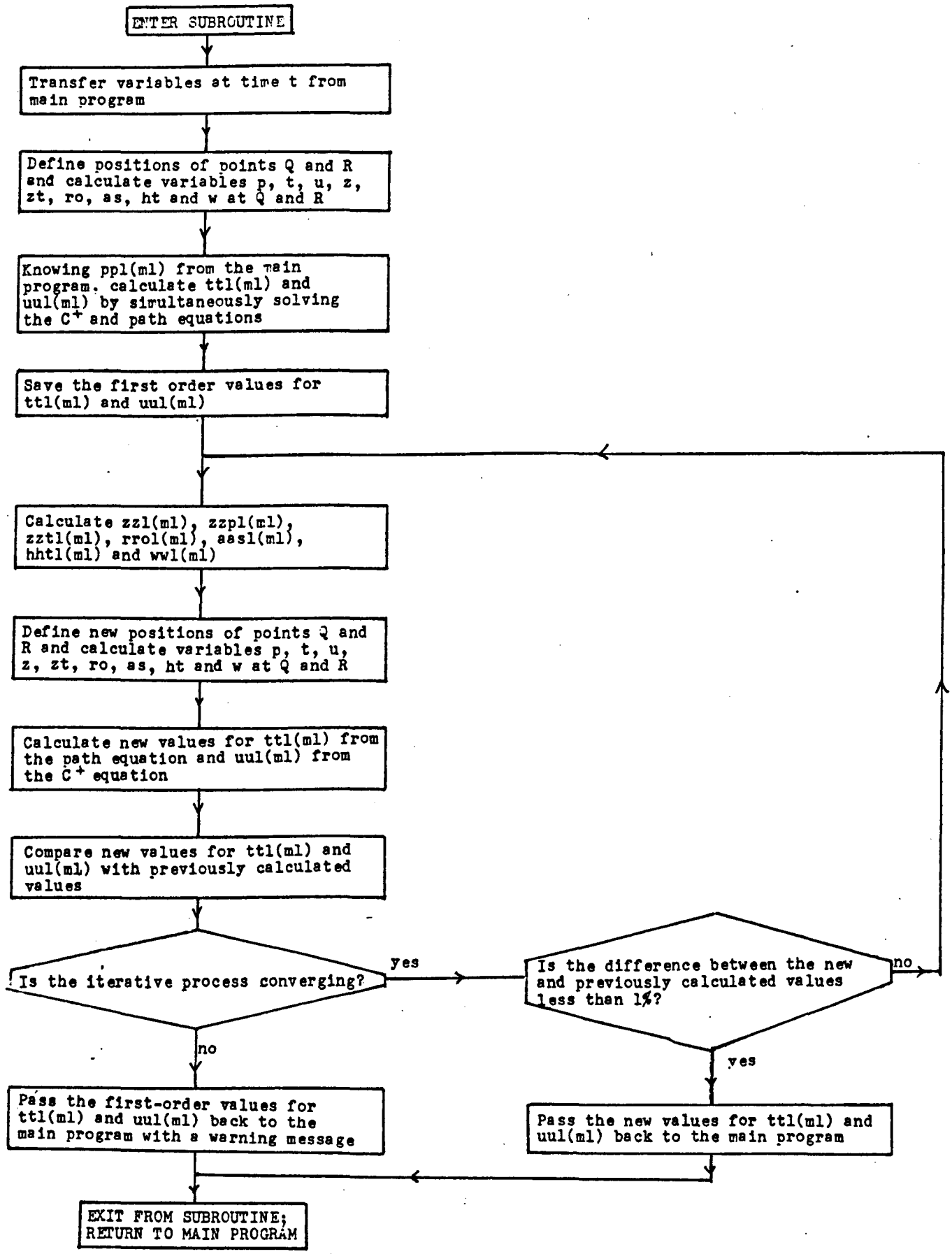


Figure 5.8(b) Flow Diagram for BREAK3

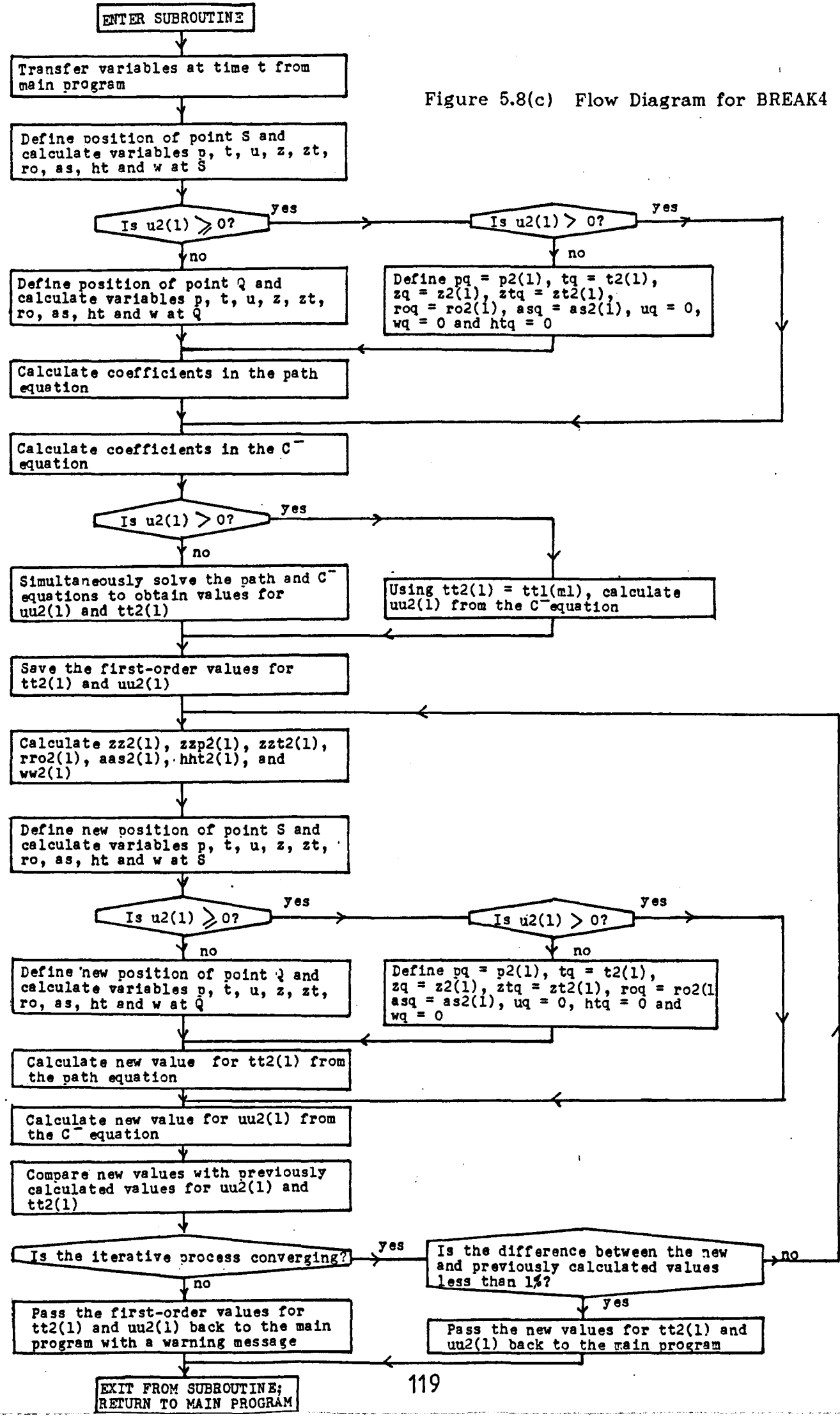


Figure 5.8(c) Flow Diagram for BREAK4

rate. With these two assumptions the flow velocity and temperature can be predicted using the C^- characteristic and the equation of state.

At the downstream end of pipe 2, DOWN1 assumes a constant temperature non-return valve situation. While the flow is positive the pressure and flow velocity are calculated by simultaneously solving the C^+ and path characteristics. When the flow rate falls to zero, the subroutine assumes that it is prevented from flowing back down the pipe and takes a value of zero for the flow velocity in any further calculations. With the constant temperature assumption as well, the pressure at this downstream boundary can then be found from the C^+ characteristic alone.

Figures 5.9(a) and (b) show the flow diagrams for these two subroutines; listings are presented in Appendix V.

Two further subroutines are used by the transient analysis program. The first GETFIL is a simple routine which opens a data file. The second DMINV calculates the inverse of a matrix. This routine is used to simultaneously solve the C^+, C^- and path equations. Listings of these two subroutines have been included in Appendix V.

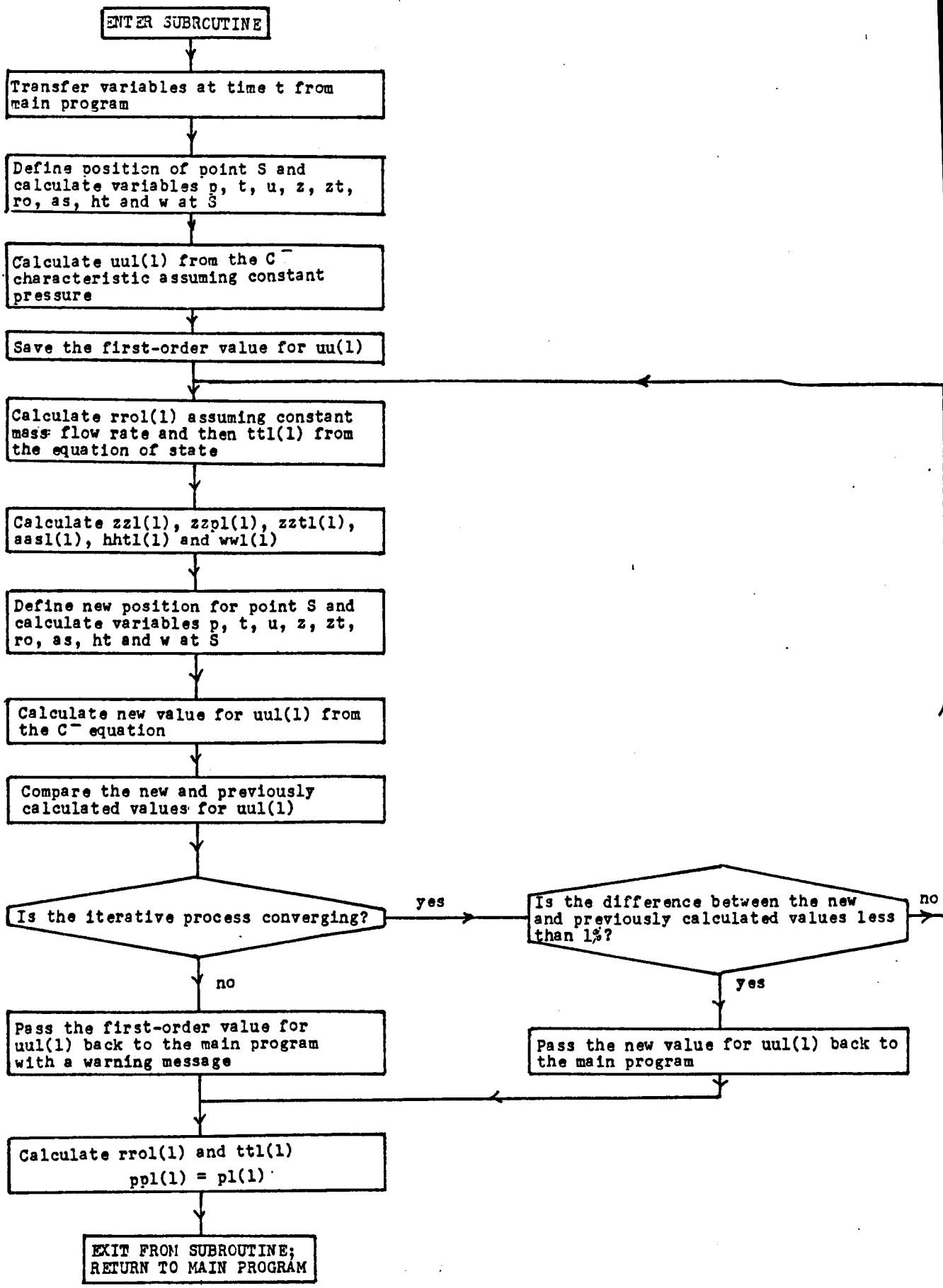


FIgure 5.9(a) Flow Diagram for SUBUP

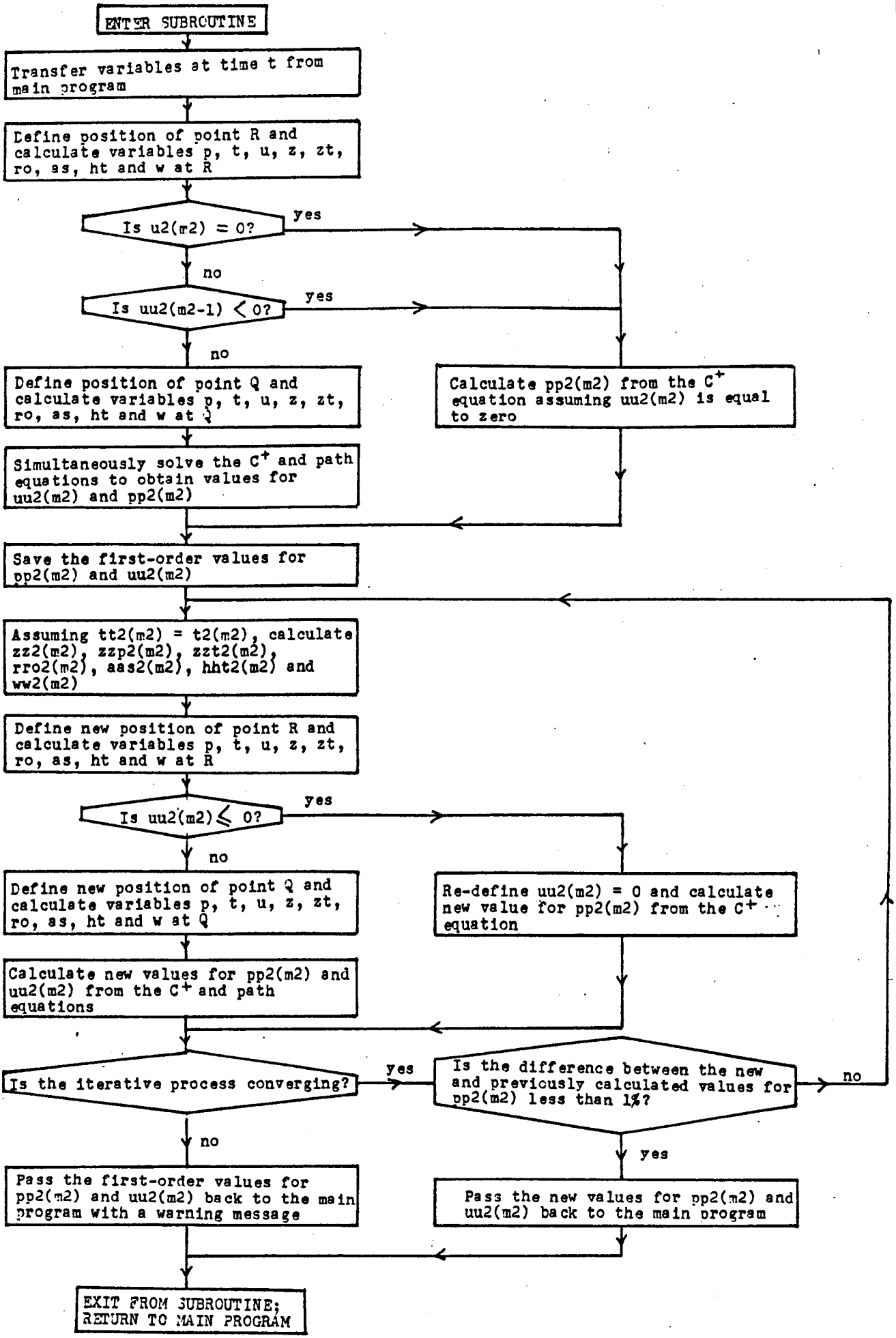


Figure 5.9(b) Flow Diagram for DOWN1

5.3. GRAPHICS PROGRAM

The purpose of this graphics program was to obtain graphical output from the results data file created by the transient analysis program. The program offers the option of pressure versus time or pressure versus wavespeed graphs so that direct comparisons with experimental data may be achieved.

To produce the graphs the program calls up a number of NAG routines from the mainframe Gould. The only input necessary are the grid values for which output is required. However, slight modifications to the program were necessary for each set of data in order to set the maximum values on the axes and to alter the title of the graph. A listing of this program is shown at the end of Appendix V with the modified sections highlighted and labelled.

CHAPTER 6

EXPERIMENTAL DATA

6.1. INTRODUCTION

In order to validate the theoretical linebreak model that has been developed, some comparisons with experimental data were necessary. Such comparisons effectively check that the idealizations and assumptions inherent in the theory are realistic.

Numerous experimental pressure transient investigations in gas and two-phase flow have been conducted over the past forty years. They may be categorized into two main subgroups, namely, the shock tube analyses and the full size tests. The following two sections of this chapter review much of the work that has been carried out.

To enable meaningful comparisons to be made with the theoretical model, a certain amount of detail of the experimental set-up and results must be obtainable. Section 6.4 examines more closely suitable experimental data and section 6.5 details the preparation necessary for the programming of each set of data.

6.2. REVIEW OF LABORATORY EXPERIMENTS

For many years shock tubes have been used to illustrate pressure wave phenomena by rupturing a diaphragm separating areas of high and low pressure fluid. Originally, the studies mainly concentrated on the low pressure section, examining the compression waves created in this section. However, the need to simulate blowdown in water cooled power reactors caused Edwards and O'Brien [1970] to investigate the effects of the expansion waves in the high pressure section. They slowly heated a water filled pipe (length 4.096 m, diameter 73 mm) to a fixed temperature and pressure above the saturation conditions. A glass bursting disc at one end of the pipe was then ruptured and the subsequent transient pressures and temperatures were recorded at seven tapping points along the length of the pipe. Transient void fraction readings were also taken at two of the stations and the end thrust exerted by the shock tube was measured. This blowdown test was repeated a number of times with various initial pressures and temperatures. The results were presented in the form of pressure x time, temperature x time, void fraction x time and end load x time graphs. From these results it could be seen that after rupture, the pressure in the pressurized section falls below the initial saturation value and, although recovering slightly, remains below this saturation value. It was also concluded that the decompression wave, caused by the rupture of the bursting disc, travelled upstream at approximately the isentropic speed of sound in the compressed liquid phase.

Further experimental data of Edwards and O'Brien, obtained using a 32 mm diameter shock tube, were presented by Hancox, Mathers and Kawa [1975] again in the form of pressure x time graphs.

Premoli and Hancox [1976] furthered the shock tube work of Edwards and O'Brien by using initially flowing subcooled pressurized water with heat addition. They used a vertical, uniformly heated test section. The blowdown was initiated by isolating this section and simultaneously rapidly opening a discharge valve. Extensive data are presented in their report, including depressurization rate, mass hold-up and discharge rate, as well as the heat transfer data.

Shock tube experiments using gas rather than water vapour have been performed by Groves et al.[1978] in an attempt to simulate a gas pipeline rupture. In order to describe the decompression wave associated with such a rupture, they examined the high pressure side of the diaphragm in a shock tube that was considerably longer than that used by Edwards and O'Brien (length 30.48 m, diameter 60.3 mm). Using methane, argon and natural gas as working fluids, the results obtained illustrated the variation in wavespeed of the expansion wave with pressure. Any discrepancies between experimental and theoretical results were accounted for in that the small diameter effects (e.g. heat transfer, flow restriction due to boundary layer build-up and successive condensations) were not included in their theoretical analysis.

Issa and Spalding [1972] appeared to obtain better agreeement between their theoretical analysis and the experimental shock tube data of Mack [1954] as used by Williams [1956]. The working fluid was assumed to be a perfect gas in the theoretical analysis but the effects of friction and heat transfer were included (which in practice weaken a shock wave). Although they did not compare the theoretical and experimental variations of wavespeed with pressure, the normalized velocity and mass velocity distributions obtained using their model (with carefully selected friction factor and Stanton number) compared well with those obtained experimentally.

British Gas have conducted numerous shock tube experiments using methane/ethane, methane/propane and natural gas mixtures in order to validate their theoretical rich gas decompression behaviour model. The shock tube used was of length 36.58 m and diameter 101.6 mm and decompression was initiated by explosively bursting a disc at one end of the tube. Pressure x time data were recorded using pressure transducers at a number of locations along the shock tube. The results of these tests are presented by Jones and Gough [1981].

Also presented by Jones and Gough are the results from some BMI experiments conducted using a shorter tube (length 6.1 m, diameter 101.6 mm). These results were obtained using natural gas as the working fluid and are presented in the form of pressure x wavespeed graphs. However, details of the experimental apparatus and procedure used to obtain these results have not been published.

6.3. REVIEW OF FULL SIZE TESTS

Some authors, for example, Cheeseman [1970], argue that since the rapid transients are quickly dissipated by friction, the main pressure transients of concern to the line operator are those arising from the packing and unpacking of gas in the pipeline. Indeed, the analysis of these long period transients is essential if full advantage is to be taken of a network's capacity for linepacking.

There have been several experimental studies on these slower transients many of which have been conducted using looped or branched systems. The first major series of experiments using full-size pipelines was conducted by Wilkinson et al.[1964] in the early 1960's. Five single pipelines were examined of various lengths, diameters and topography. Flow and pressure variations were imposed at the outlet of each pipe and the flow and pressure were recorded at both ends of the pipe. Good agreement was obtained between theoretical and measured input flows and pressures.

Following this, Heath and Blunt [1969] recorded the effects on a section of the British Gas Council high pressure grid when one supply point was rapidly shut down and left off for seven hours. The flows and pressures were monitored at each take-off point and tee at five minute intervals. Although the test was limited by having only one sudden flow change, the results obtained agreed well with those predicted from an isothermal analysis. It was realized,

however, that just one inaccurate reading could affect the pressures predicted throughout the network so extreme care had to be taken when recording the flow measurements.

Rachford and Dupont [1974] compared predictions from their isothermal analysis with recorded experimental data for slow transients in a looped network (the source of the experimental data not being disclosed). Although the network was fairly complicated, they managed to obtain quite accurate pressure history predictions for various points around the network for the ten hour test duration. They also carried out an experimental investigation on a 0.59 m diameter, 53 km long, two-leg gas pipeline. They imposed sudden flow variations at the inlet end of the pipe and slow variations at the outlet and then compared the calculated and observed pressure histories. Good agreement was obtained between their theoretical results and recorded values, the maximum discrepancy between them being 5 psi over the 12 hour simulation.

Weimann [1978] used both a branched network and a single four-leg pipeline to validate his isothermal model predicting the packing and unpacking of the gas. With the network he recorded supply and demand flows at one hour intervals and took pressure readings at fifteen minute intervals for a twenty-four hour period. From this study he discovered that if the boundary values were hour step functions, then one hour time steps had to be used in his simulations. With the four-leg, 78 km long refinery gas transmission pipeline, Weimann imposed transient supply and demand flows and

compared the measured pressure variations with those predicted for his isothermal analysis. Although the changes in flow rate each took place within one minute, the resulting effect was the gradual packing and unpacking of the pipeline. It was, therefore, comparatively slow pressure transients that were being measured.

More recently, Mekebel and Loraud [1983 and 1985] investigated unsteady flows and pressures in a 0.22 m diameter, 19.345 km long gas transmission pipeline operating at pressures up to 20 bar. They examined the effects of heat conduction between the pipe and its surroundings and concluded that this heat transfer was a necessary inclusion in the theoretical analysis. This contradicted the common assumption of isothermal flow in slow transient situations.

Although, as already mentioned, these slow transient analyses are important, it is the rapid transient simulations that are of greater significance to this project. Stoner [1969] was one of the first people to examine rapid transients in full size pipes. However, he concentrated on the compression wave caused by rapid downstream valve closure rather than the linebreak problem. He determined the wavespeed of the compression wave and then recorded the upstream and downstream pressure histories in a 0.31 m diameter, 22 km long pipe when the valves at both ends of the pipe were simultaneously closed.

At this time in France, Sens et al.[1970] were investigating the effects of rapidly opening a downstream valve in order to simulate a linebreak. Using a 1.065 m diameter, 11.8 km long pipe, they discovered that at a distance of 6 km from the venting point, the rapid opening of the discharge valve had the same effect as rupturing a bursting disc. This enabled them to repeat the experiment and compare recorded pressure histories with those predicted from their theoretical model. They found that, although the drop in pressure following a 'break' was slower in reality than in their calculations, the shape of the recorded curve was identical to that of their theoretical curve.

In the Netherlands, Van Deen and Reintsema [1983] have used experimental data from the Gasunie transport system to validate their theoretical model. They conducted two major experiments. In the first experiment a linebreak was simulated by rapidly opening a valve which connected the test pipe to a parallel pipe at lower pressure. The point on the test pipe at which the measurements were taken was 10 km downstream of this valve. The pressure history was recorded and a detailed comparison was made between this measured data and that obtained from their theoretical analysis. The second experiment involved rapidly opening a gate valve situated between two measuring points on a test pipe. The pipe was 90 km long with a diameter of 0.76 m. Gas was supplied at both ends of the pipe and delivered to a number of take-off points along the pipe. As the gate valve was opened, the flow, pressure and temperature were recorded

at both measuring points. The results showed that a fast pressure transient occurred at both measuring points due to the valve opening.

Further tests with natural gas have been conducted using short sections of pipe, investigating the events that occur in the immediate vicinity of a break. British Gas performed two tests on a 1.22 m diameter pipe for Foothills Pipelines (Yukon) Ltd. The test sections were 50 m and 51.2 m long with reservoir sections at both ends. After pressurizing the pipe to approximately 90 bar, a crack was initiated at the centre of the test section and the pressure histories were recorded at points either side of the break. The results from these tests have been presented as pressure x wavespeed graphs by Jones and Gough [1981] although further experimental data has not been published.

Jones and Gough also presented pressure x wavespeed graphs from three tests carried out by BMI on behalf of British Gas and from a test that British Gas conducted for Shell. The initial pressures in these tests were between 120 and 140 bar. The experimental details of the tests are given by Maxey, Syler and Eiber [1975] and by Hayes and Lux [1979].

Between December 1979 and April 1981, Foothills Pipelines (Yukon) Ltd., undertook a program of linebreak tests at the Northern Alberta Burst Test Facility. The main purposes of this test program were to

examine the effect of the gas composition on fracture behaviour, to establish the limiting values of Charpy toughness which would give fracture arrest, and to confirm the arrest capabilities of the test pipe. Short lengths (less than 100 m) of 1.4 m and 1.2 m diameter pipe were charged with natural gas of known composition and pressurized to between 74 and 87 bar. Fracture was initiated at the centre of the test section by detonating an explosive cutter. Further details of the test together with the results (in the form of pressure histories and timing wire data showing the crack tip position) are given by Rothwell [1981].

Finally, there have also been some transient network experiments conducted using steam as the working fluid. The purpose of these was to support simulations for boiler steam lines and reactor blowdown. For example, Ying and Shah [1978] investigated steam hammer in the main piping system of an oil fired power plant. They imposed transient conditions in the network by rapidly closing the turbine stop valves and then obtained oscilloscope traces of the pressure surges created.

Another example of steam transient experimentation is the work of Banerjee and Hancox [1978]. They conducted a series of blowdown experiments on a figure-of-eight loop containing pumps, heaters and heat exchangers. The blowdown was started by rapidly opening a quick-acting valve. Pressures, temperatures, coolant densities, and flow velocities were then recorded at various points around the circuit and the results obtained compared with those predicted from the computer code of Arrison et al.[1977].

6.4. SELECTION OF TEST DATA

The selection of experimental data for comparison with the theoretical model was determined by the following criteria:-

i) The variables required by the program (as detailed on the data sheet, Figure 5.1) must either be given in the experimental data or be calculable from it.

ii) The experimental results must be of a form that can be directly compared with the theoretical computer output, for example, in the form of pressure x time or pressure x wavespeed graphs.

iii) Details of the apparatus and procedure are necessary in order to assess the experimental error and evaluate the results obtained.

The following shock tube and full-size data were selected to validate the computer model:

i) The shock tube data of Groves et al.[1978]

This data was selected for its transient results obtained using the single gases of methane and argon since other suitable experimental results used only mixtures of gases. However, details of the apparatus and procedure used by Groves [1976] were unobtainable and so various assumptions had to be made regarding the shock tube material (in order to estimate the friction factor and

Stanton number), the effective rupture time of the diaphragm, and the accuracy and sensitivity of the measuring and recording devices. Therefore no assessment could be made of the experimental errors incurred.

The gas and pipe data provided by Groves et al.[1978] were pipe length and diameter, initial temperature and pressure for each test and the gas composition (accounting for the slight impurity of methane). Also recorded were experimental values of the sound speed for each test. From this initial data, graphs of pressure ratio x wavespeed were recorded. Figure 6.1 shows the dimensions of the shock tube and the positions of the pressure transducers.

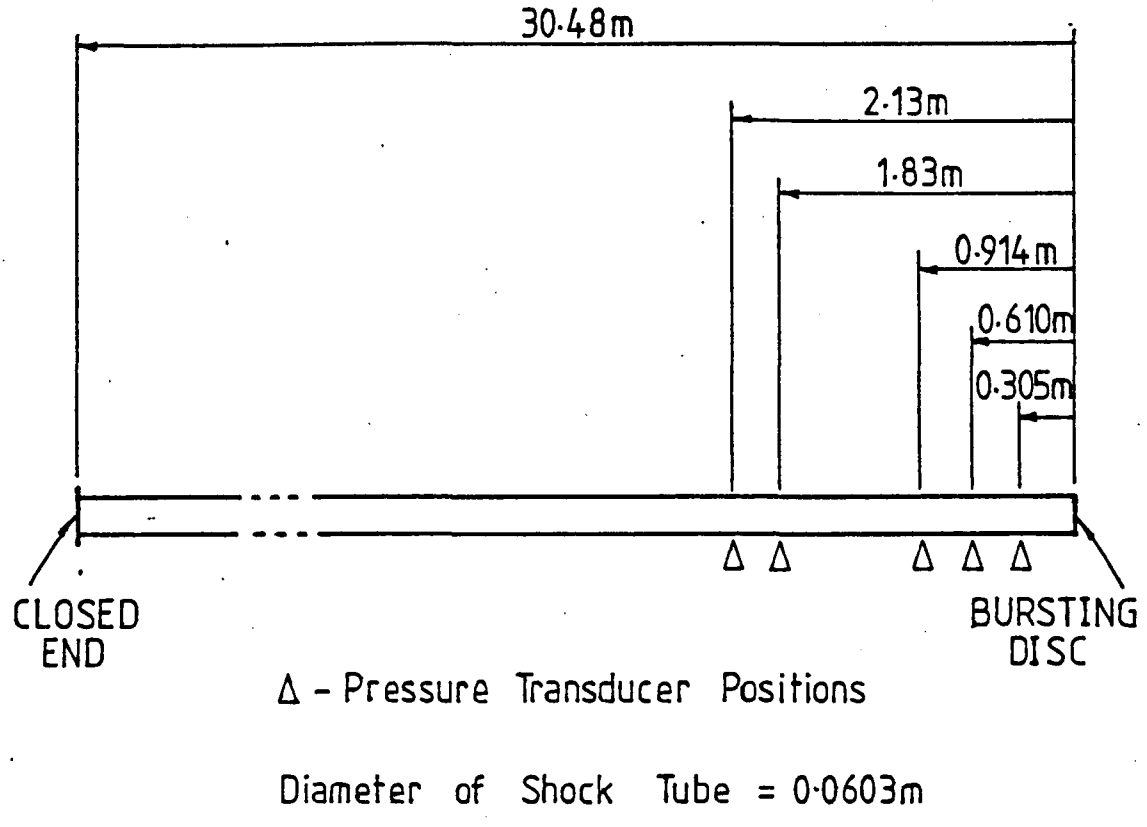


Figure 6.1. Shock Tube used by Groves et al.

ii) British Gas shock tube data (Jones and Gough [1981])

Three burst tests were carried out on each of the following gas mixtures using different initial pressures for each test:

a) 85% methane, 15% ethane

b) 90% methane, 10% propane

c) natural gas (of known molar composition).

The shock tube was constructed from seamless drawn steel tubing with a wall thickness of $^1/_{16}$" and a maximum pressure specification of 130 bar. It was welded to a girder with a heavy metal base to prevent any axial or longitudinal movement. A schematic of the shock tube is given in Figure 6.2.

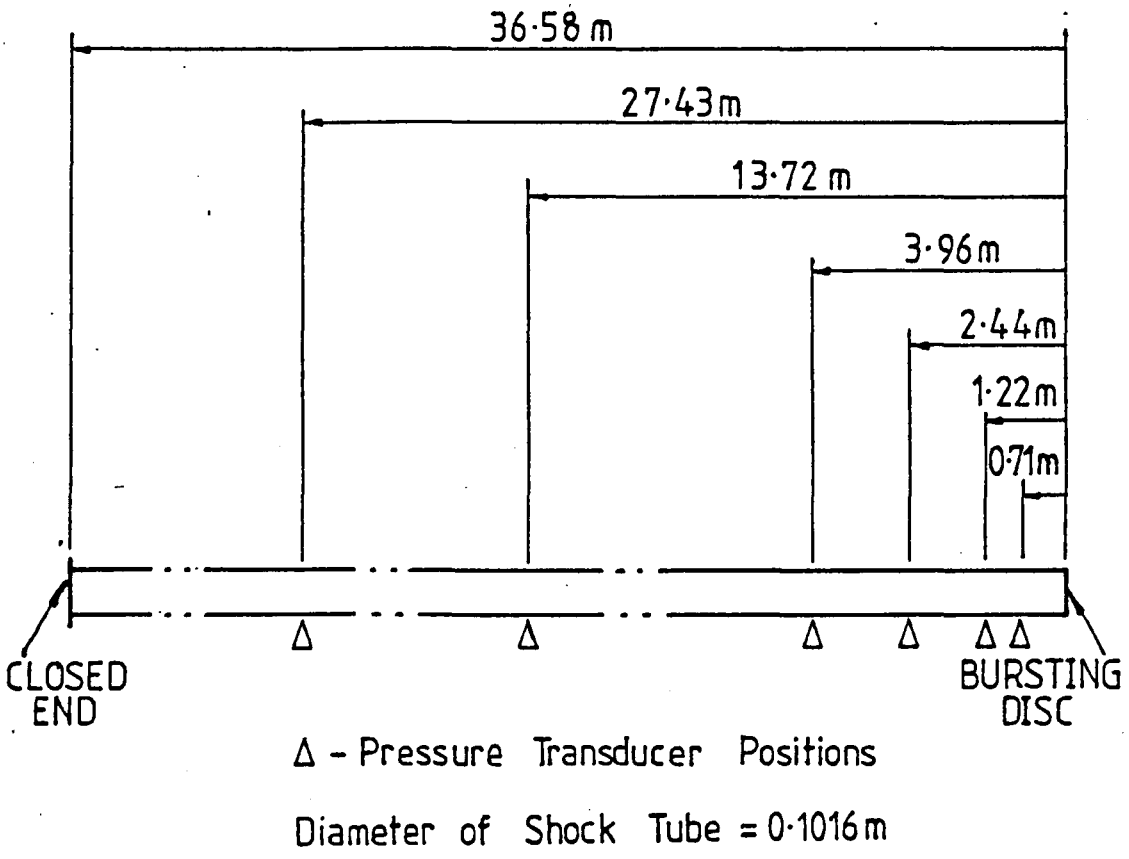


Figure 6.2. British Gas Shock Tube

The bursting disc was ruptured using a 0.123 oz explosive charge which would be insufficient to create any noticeable heat transfer into the pipe. However, the time taken for the rupture to occur was not recorded.

The initial temperature was measured using thermocouples mounted in the pipe wall. They were therefore measuring the pipe wall temperature rather than the initial gas flow temperature.

Strain gauge piezo-resistive pressure transducers with small diaphragms were used to measure the pressures in the shock tube. The positions of these transducers are shown in Figure 6.2. The pressure transducers (having an accuracy of 0.3%) were connected to a tape recorder producing a continuous analogue recording. The complete pressure measuring system was statically calibrated and demonstrated an accuracy to within 5%.

The gas composition of each test was measured and recorded although the methods of mixing and analysing the gas were not stated. The homogeneity of the gas and the accuracy of the recorded gas analysis are therefore open to speculation and could be a source of discrepancy between theoretical and experimental results.

For each test the initial pressure and temperature were noted and pressure x time histories were obtained for several positions along the shock tube.

iii) Foothills Pipelines (Yukon) Ltd. full size tests at the Alberta Burst Test Facility (Rothwell [1981])

Although results from six burst tests using natural gas are presented in this reference, only four sets were used for comparisons with the theoretical model since the last two tests gave no indication of the positions of the pressure transducers in the pipe. For each test the pipe diameter and length, wall thickness, weld type and initial temperature and pressure were recorded. Included in the test data used were clear diagrams showing the test section with the positions of the pressure transducers, strain gauges, timing wires and resistance temperature detectors. These have been reproduced in Figure 6.3.

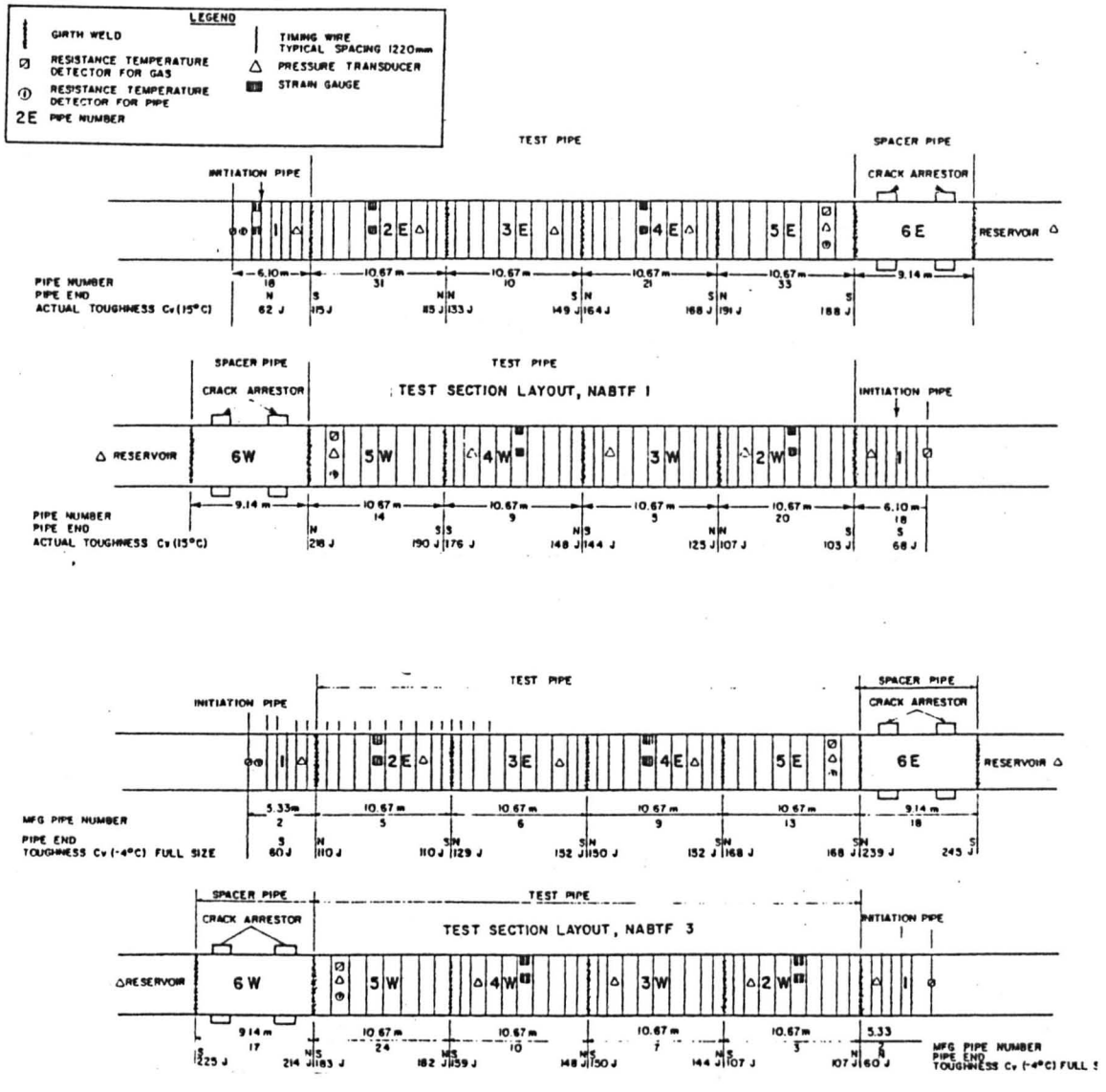


Figure 6.3. Test Sections used by Foothills Pipelines (Yukon) Ltd.

The gas mixtures were made up to specification and their compositions analysed using two gas chromatographs. The initial pressure and temperature were monitored and recorded by a data-logger and the timing wire, pressure transducer and strain gauge signals were recorded using tape recorders. The signals could then be retrieved for analysis by a high speed UV recorder. To ensure a time reference to within 0.1 ms, each magnetic tape carried a synchronization trace as well as the burst initiation signal.

The test section and process loop were initially pressurized to a level above the test pressure. Hydrocarbons were then added to reach the target composition for the gas. Each gas component was injected at a constant rate over a whole number of circulation cycles of the process loop in order to ensure homogeneity. The temperature was adjusted to the test temperature (± 1°C at all measuring points) and the pressure was slowly bled off to the test pressure (±5 kPa at all measuring points). The test section was then isolated from the process loop.

The fracture was initiated by detonating an explosive cutter placed at the centre of the test section. This cut approximately 75% of the way through the wall thickness which was sufficient to initiate pipe failure.

The test was monitored by high speed cameras which were triggered a few seconds prior to crack initiation and by a video system.

6.5. PREPARATION OF THE DATA

In order to conduct theoretical simulations of the tests outlined in section 6.4, the experimental details must be presented in terms of the specific gas and system data required by the program. The calculation procedures necessary to convert the available data into a programmable form are detailed below.

6.5.1. Preparation of Gas Data

For each test the specific heat at constant pressure, the specific gas constant and the critical temperature and pressure of the gas are required.

(i) Specific Heat at Constant Pressure C_p

In practice, the specific heats may vary over the considerable temperature and pressure ranges encountered in transient analyses. For the program, however, constant values were assumed for simplicity. (The consequences of such an assumption are discussed in Chapter 8).

The values used for the specific heats of the pure gases such as argon and the natural gas components, were calculated from the specific gas constant of the gas (R) and its ratio of specific heats (γ):

$$C_p = \frac{\gamma}{\gamma - 1} \cdot R$$

Using these values together with the molar compositions of the gas mixtures given in the experimental details, the specific heats of the various gas mixtures were calculated. Further details of this calculation procedure are given in Appendix VI.

(ii) Specific Gas Constant R

For each gas mixture a mean molecular weight was calculated and a value for the specific gas constant of the mixture was then obtained by dividing the universal gas constant by the mean molecular weight. This procedure is shown in Appendix VI.

(iii) Critical Temperature T_c

The critical temperatures of the natural gas mixtures were calculated using a formula for hydrocarbon mixtures detailed by Grieves and Thodos [1962]. This method was chosen in preference to calculating pseudo-critical temperatures as described by Kay [1936] since the critical temperatures of the individual components differed by more than 200% in extreme cases. This would lead to an error of more than 5% in Kay's calculated values. Details of the method of Grieves and Thodos are presented in Appendix VI.

(iv) Critical Pressure P_c

Although a number of correlations for computing the critical pressure of hydrocarbon mixtures have been developed (for example,

Mayfield [1942], Organick [1953], Etter and Kay [1961]), these generally have some difficulty in handling systems that contain methane. Grieves and Thodes [1963] overcame this problem to an extent but their method was relatively complex. Therefore, for this project the method of Prausnitz and Gunn [1958] for calculating pseudo-critical pressures was chosen for its simplicity.

It should be noted, however, that the accuracy of this method is dependent on the accuracy of the critical temperature and slight inaccuracies may be produced near the critical region.

These calculated values for the specific heat, specific gas constant and critical temperature and pressure for each of the gases used in the tests are presented in Table 6.1.

6.5.2. Preparation of System Data

The system data consists of the pipeline data and the initial flow conditions. Although the experimental data provides such details as the pipe dimensions and initial pressures and temperatures, the grid size and some variables (for example, friction factor and Stanton number) still need to be decided upon in order to obtain a theoretical plot. For each system, the procedure for determining the system data was as follows.

i) A suitable grid size was chosen which would produce stable results. For each separate set of test data, the program was adjusted so that output was obtained only at the relevant i values

TABLE 6.1. EXPERIMENTAL GAS DATA

DATA SOURCE AND GAS	C_p (kJ/kg K)	R (kJ/kg K)	T_c °C	P_c kPa
Groves' shock tube data:				
Methane	2.170	0.517	-82	4610
Argon	0.520	0.208	-122	4870
Natural Gas	1.960	0.432	-50	4888
British Gas shock tube data:				
Methane/ethane	2.125	0.458	-56	4872
Methane/propane	2.113	0.441	-40	5076
Natural Gas at 70 bar	1.941	0.408	-40	4873
Natural Gas at 100 bar	1.940	0.407	-40	4876
Natural Gas at 125 bar	1.946	0.411	-39	4921
Foothills Pipelines (Yukon) Full size tests at the Alberta Burst Test Facility:				
NABTF1 - Natural Gas	2.006	0.445	-53	4878
NABTF3 - Natural Gas	1.999	0.441	-50	4919
NABTF4 - Natural Gas	1.996	0.440	-50	4914
NABTF5 - Natural Gas	1.994	0.440	-50	4929

which coincided with the pressure transducer positions specified in the experimental data. The grid size used and details of the pressure transducer positions are included in the results.

ii) In the program, the gradient of the pressure drop at the break and the time taken for the break to occur are both functions of grid cycles rather than direct functions of time. It was therefore necessary to check that the break readings being produced by the program for each set of data were feasible. For example, the time step selected for the transient analysis may cause the pressure drop at the break to occur unrealistically slowly. To overcome this, modifications can be made both to the time taken for the complete pressure drop to occur and to the varying rate of pressure drop (Figure 4.6).

The equation for the pressure drop at the break, in general form, is:

$$P(j) = (P_o - P_{eq})\ (1 - \frac{j}{64x})^n - P_{eq}$$

where P(j) is the pressure at the break after j time steps

P_o is the initial pressure at the break point before rupture

P_{eq} is the equalization pressure.

Variables n and x affect the shape and duration of the pressure drop respectively. By increasing 'n' the pressure drop becomes much steeper initially but levelling out sooner and by decreasing 'x' the

duration of the pressure drop is decreased. However, care must be taken to ensure that such changes to n and x do not create an excessively steep initial pressure drop which would cause an instability in the program at the break point.

iii) Finally, values for the friction factor and Stanton number for each set of test data had to be determined. The friction factor was initially estimated from Haaland's formula (Appendix II):-

$$\frac{1}{\sqrt{f}} = -0.6 \log \left\{ \left[\frac{6.9}{Re}\right]^3 + \left[\frac{k/d}{3.7}\right]^{3.33} \right\}$$

Provided that the pipe material is known, a value for k/d can be substituted into the above equation. Assuming initially a Reynolds number of 10^5, the friction factor can be determined. This procedure was used to obtain an initial value for the friction factor which could then be tuned for each individual case.

An initial value for the Stanton Number was obtained using the empirical formula:

$$St \cdot (Re)^{0.2} \cdot (Pr)^{0.6} = 0.023$$

Values for the Prandtl number at atmospheric temperature and pressure were obtained for argon and methane. These values were then increased slightly to allow for the higher pressures and lower temperatures that are encountered in the linebreak modelling and assuming again a Reynolds number of 10^5, the Stanton number could

be determined. However, since the Stanton number is strongly dependent on the Reynolds number, it was realised that these initial values may then need tuning for each set of test data.

The initial value calculated for the friction factor in all the sets of experimental data was 1.8×10^{-2}. The Stanton number initial values were taken to be 2.7×10^{-3} for the methane containing systems and 2.9×10^{-3} for argon.

CHAPTER 7

RESULTS

7.1. INTRODUCTION

Results were obtained by performing a number of computer simulations for each of the sets of data detailed in Section 6.4. The length of the time step, the grid size and the break details were varied for each simulation to optimise convergence towards a stable solution. (The problems encountered with the numerical instabilities in this project are discussed further in Chapter 8).

The numerical results produced by the linebreak program were written to a data file which was then used as input for the graphics program. These theoretical results could then be compared graphically with the experimental results for ease of assessment.

The numerical results (produced in tabular form) included the temperatures that were being experienced in various parts of the pipe as well as the pressure, velocity and wavespeed data. There were, however, no available experimental results with which to compare these temperatures. One major use of the tabular output was for identifying areas of numerical instability.

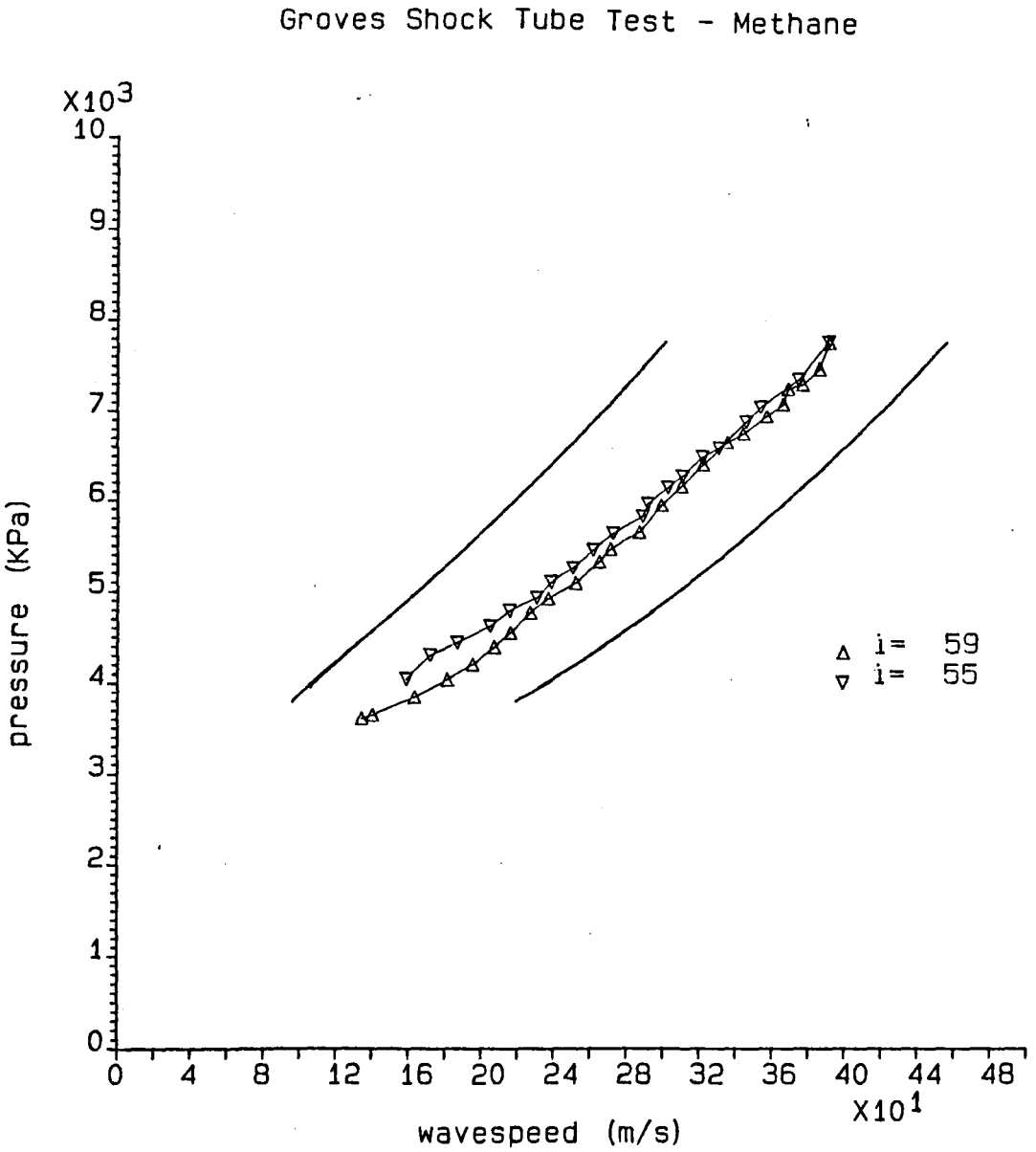
7.2. GROVES' SHOCK TUBE RESULTS

A stable simulation was obtained for each of the three gases tested by Groves et al. [1978]. These simulations used a grid size of 0.00953125 m near the break, time step length of 0.5 ms and break characteristics of index = 2, No. of steps = 40.

For each gas, pressure x wavespeed graphs were plotted for the transducer positions at which experimental data were available. This was done using both the isothermal and the isentropic wavespeeds calculated in the program. The experimental data points from Groves' graphs were then superimposed onto the theoretical graphs. These results are shown in Figures 7.1, 7.2, and 7.3. The table below details the transducer positions corresponding to the i values marked on the graphs.

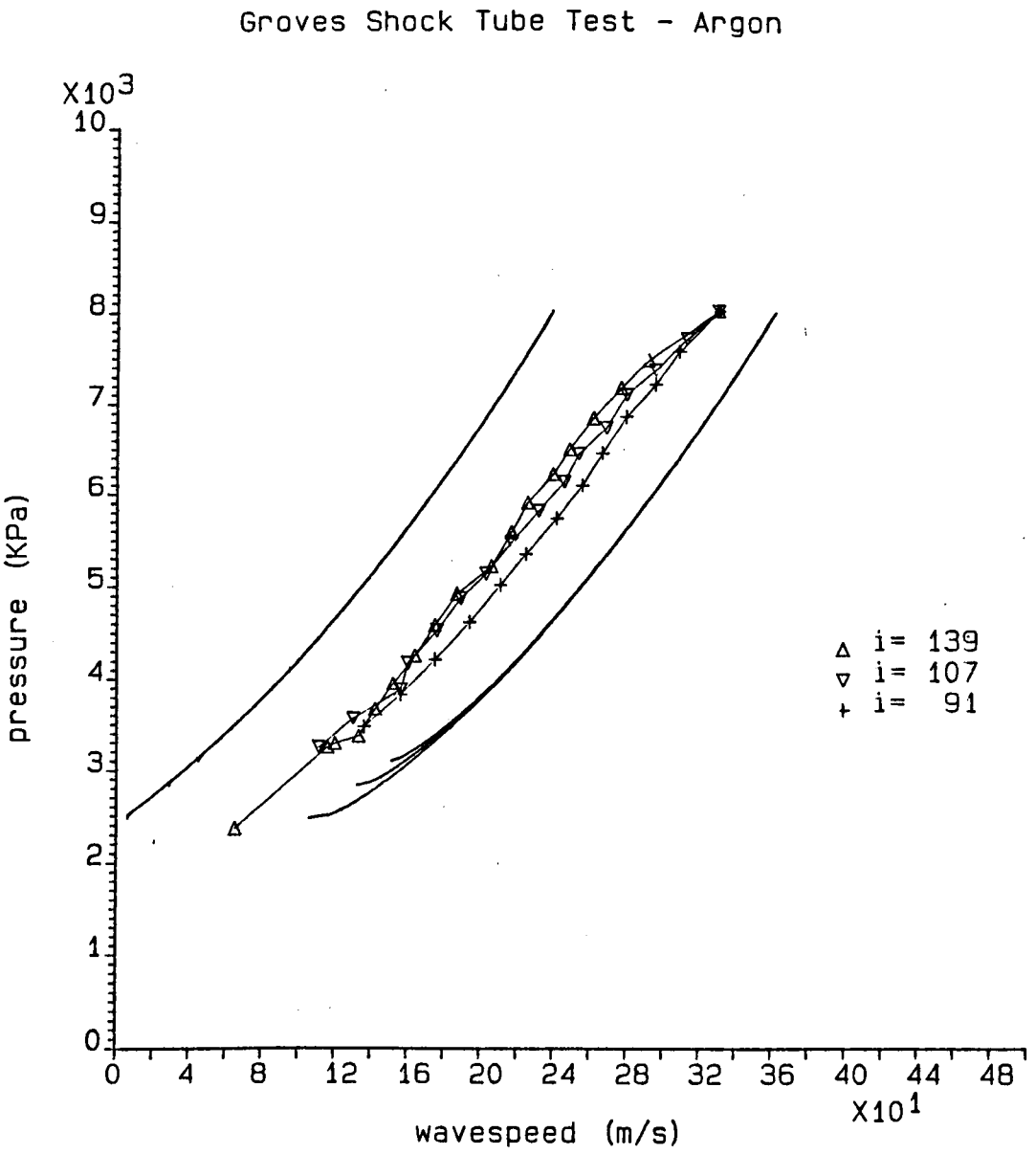
Position	i value
break	171
transducer 0.305 m from the break	139
transducer 0.61 m from the break	107
transducer 0.914 m from the break	91
transducer 1.83 m from the break	59
transducer 2.13 m from the break	55

In Figures 7.2 and 7.3, the isentropic wavespeed curves varied significantly at low pressures for the different i values. For these traces, the lowest curve was obtained for the highest i value (i.e. the point closest to the break). The lower i values (further from the break) showed higher pressures for a given wavespeed.



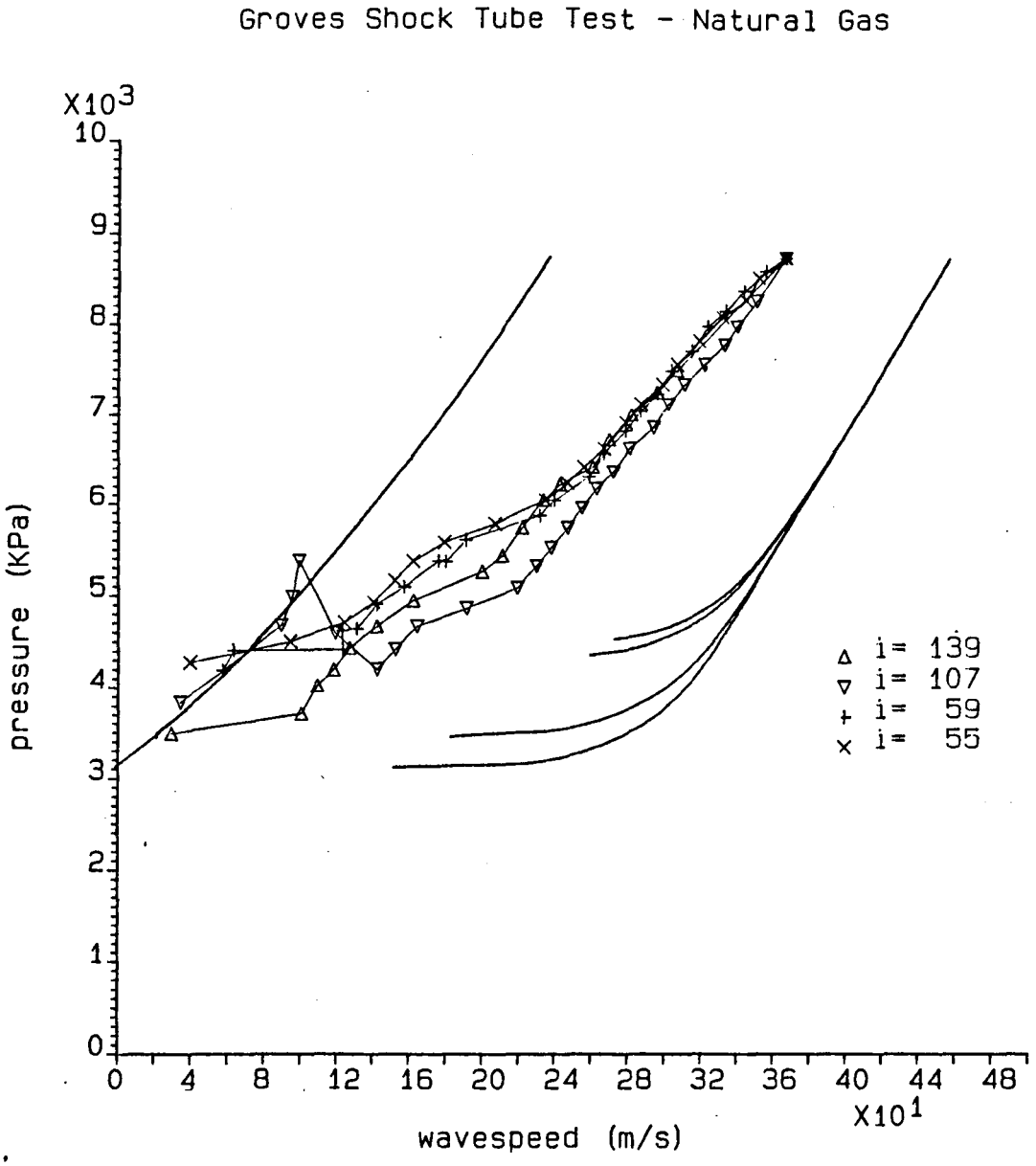
f = 0·018
St = 0·0027

Figure 7.1. P x ω Graph for Groves' Shock Tube Test - Methane



f = 0·018
St = 0·0029

Figure 7.2. P x ω Graph for Groves' Shock Tube Test - Argon



f = 0·018
St = 0·0027

Figure 7.3. P x ω Graph for Groves' Shock Tube Test - Natural Gas

7.3. BRITISH GAS SHOCK TUBE RESULTS

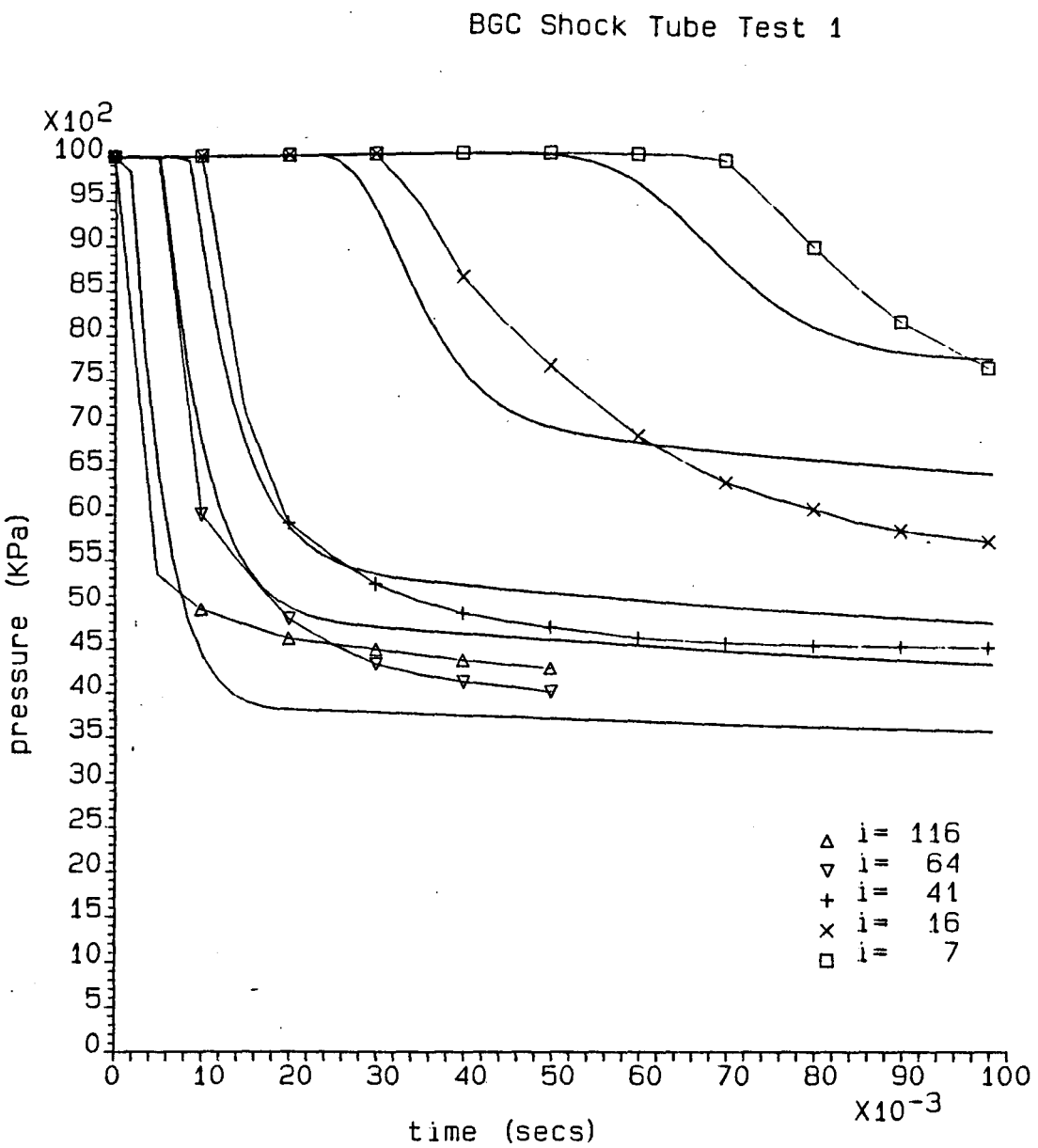
A number of attempts were made (using a grid size of 0.0254 m near the break) to obtain stable results for each of the nine shock tube tests recorded by British Gas. Figures 7.4 to 7.9 show the pressure x time traces obtained for British Gas tests 1, 2, 4, 5, 7 and 8 (Jones and Gough [1981]). In these graphs the theoretical (unmarked) curves were derived using friction factor and Stanton Number values of 0.018 and 0.0027 respectively as calculated in Section 6.5.2.

Examination of these graphs led to the modification of the friction factor to 0.01 in tests 1 and 2 (for the methane-ethane mixture) and to 0.03 in tests 4, 5, 7 and 8. The traces obtained using these new values are presented in Figures 7.10 - 7.15. Different values for the Stanton Number were also experimented with but these showed negligible difference.

The experimental plots shown with all these theoretical traces (Figures 7.4 to 7.15) were obtained by reading data points from the BGC plots presented by Jones and Gough and re-plotting them onto the theoretical graphs. The transducer positions corresponding to each of these experimental plots were obtained through private communication with D.G. Jones and are listed below:

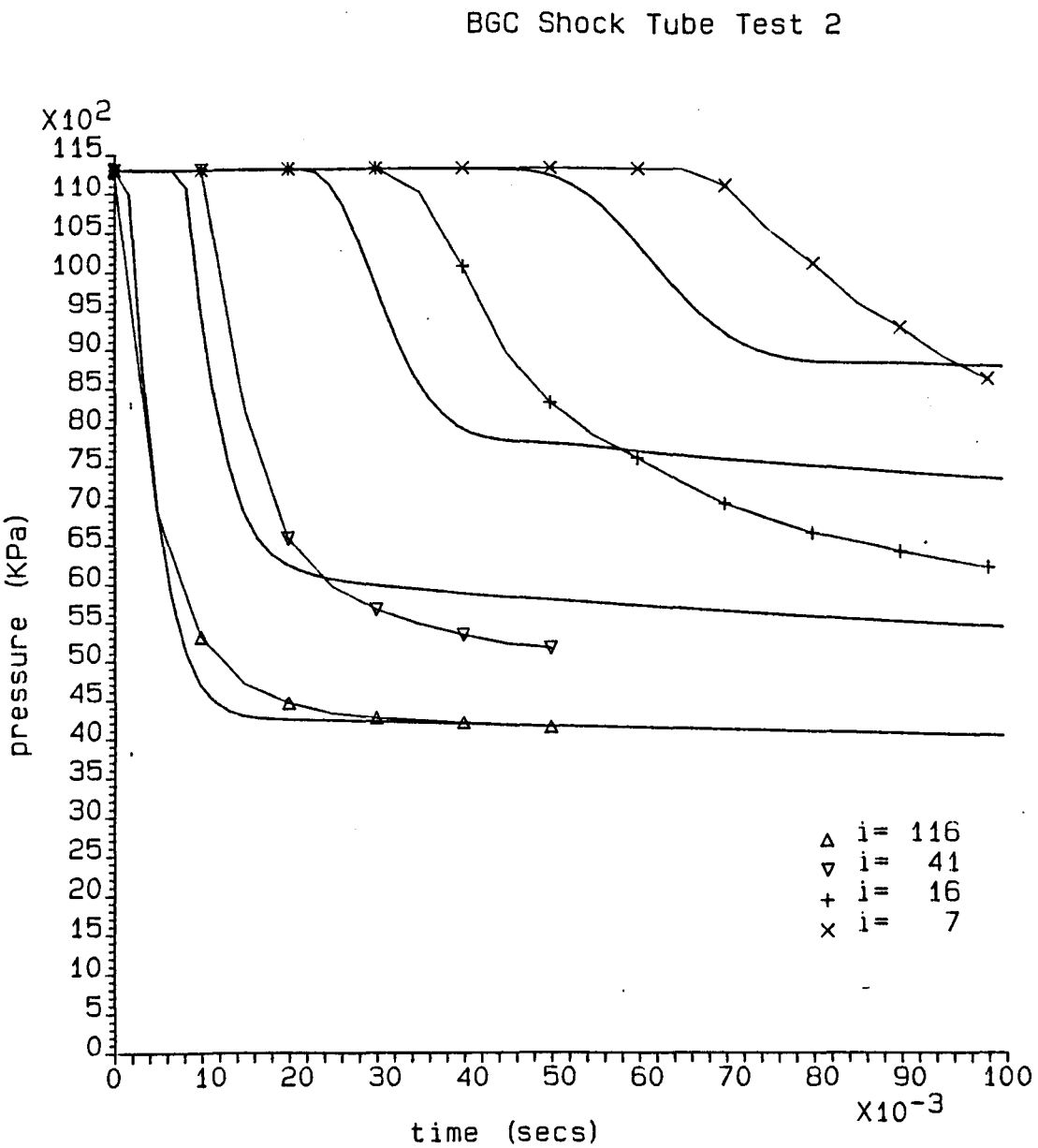
Position	i value
break	144
transducer 2 ft 4 in from the break	116
transducer 4 ft from the break	96
transducer 8 ft from the break	64
transducer 13 ft from the break	41
transducer 45 ft from the break	16
transducer 90 ft from the break	7

Unfortunately, it was not possible to achieve stable theoretical results corresponding to the British Gas tests 3, 6 and 9. In these tests the initial pressure was higher and instabilities appeared in the solution when a zero initial flowrate was used.



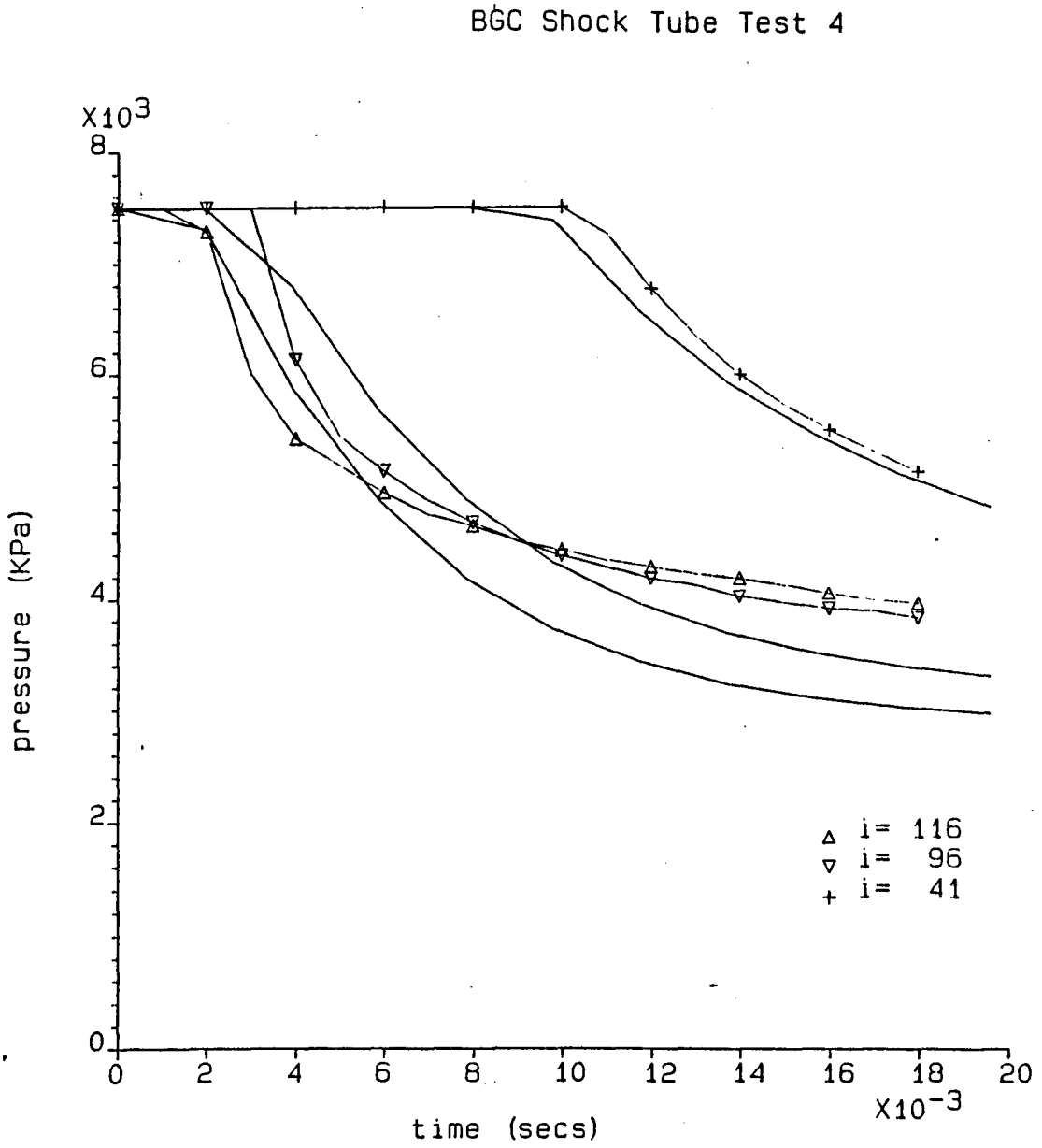
f = 0·018
St = 0·0027

Figure 7.4. P x t Graph for British Gas Shock Tube Test 1



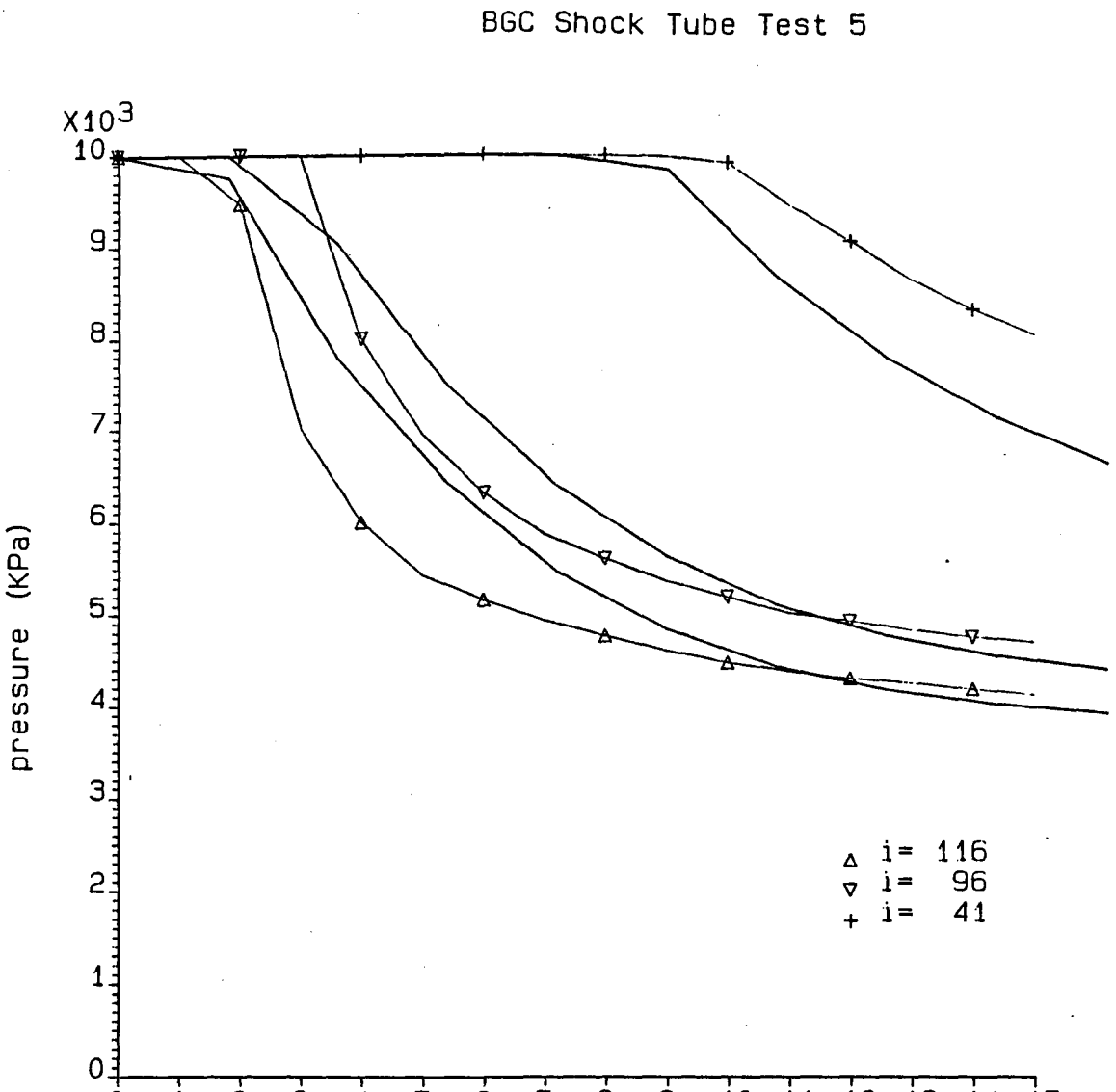
f = 0·018
St = 0·0027

Figure 7.5. P x t Graph for British Gas Shock Tube Test 2



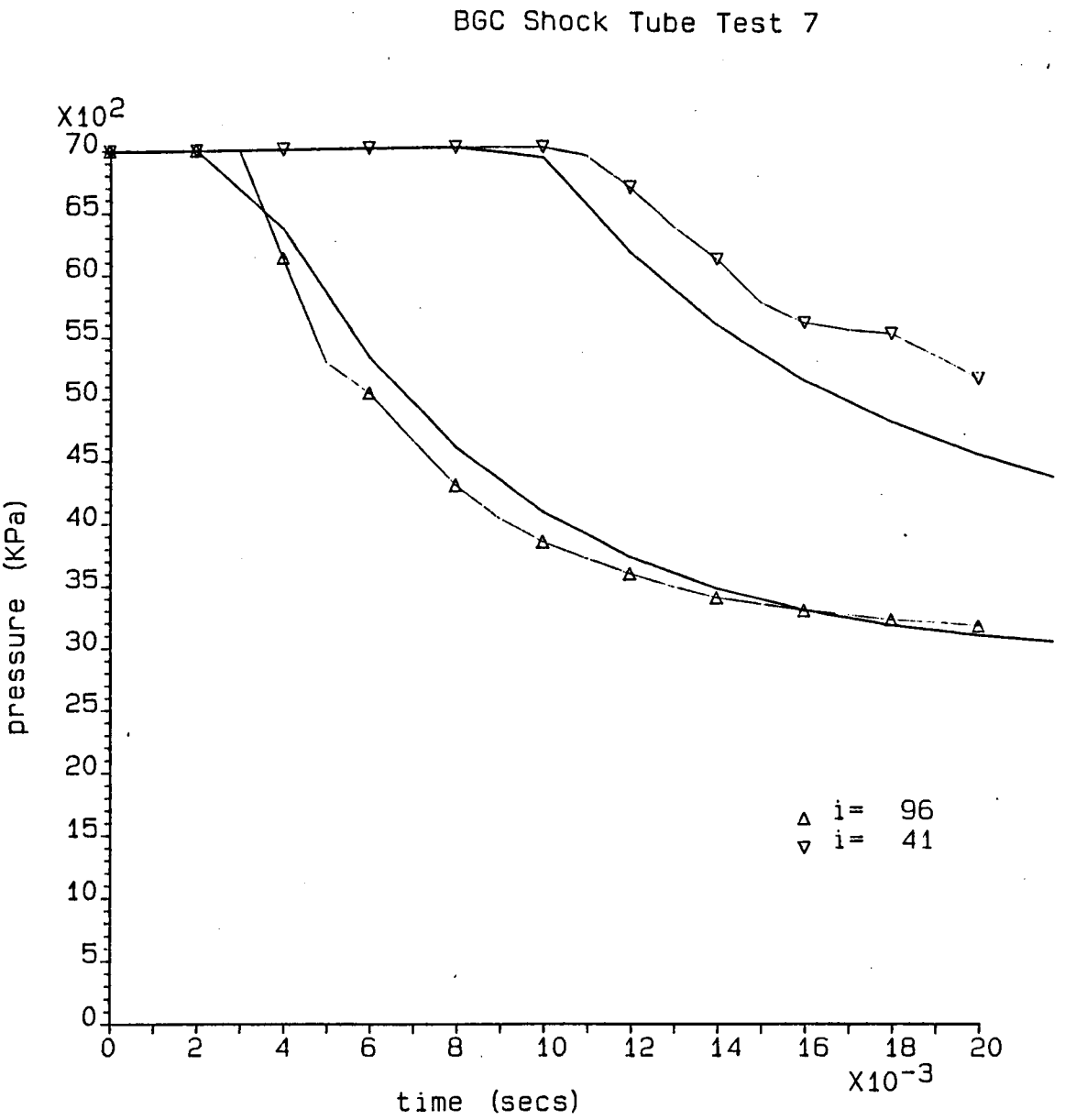
f = 0·018
St = 0·0027

Figure 7.6. P x t Graph for British Gas Shock Tube Test 4



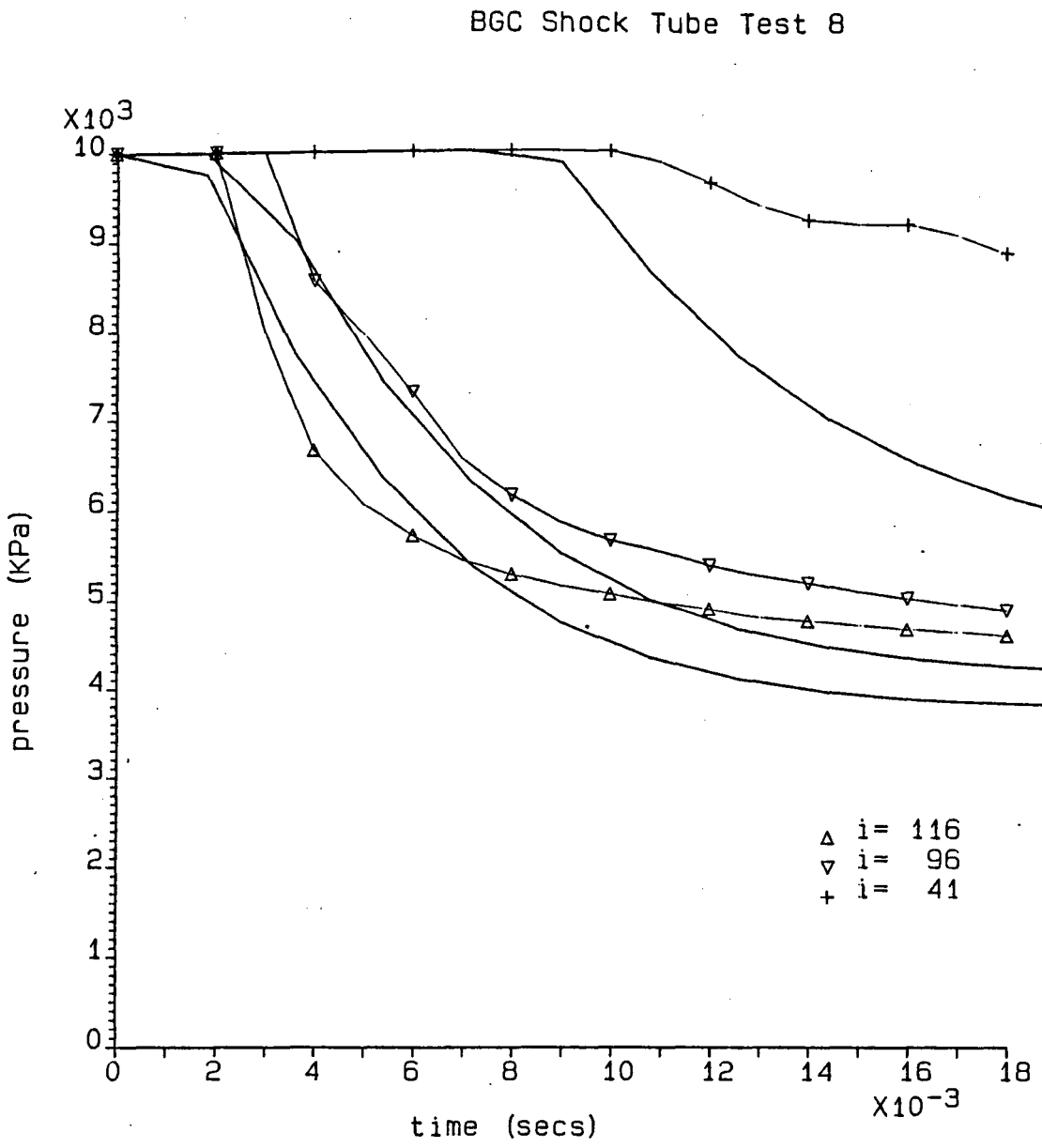
f = 0·018
St = 0·0027

Figure 7.7. P x t Graph for British Gas Shock Tube Test 5



f = 0·018
St = 0·0027

Figure 7.8. P x t Graph for British Gas Shock Tube Test 7



f = 0·018
St = 0·0027

Figure 7.9. P x t Graph for British Gas Shock Tube Test 8

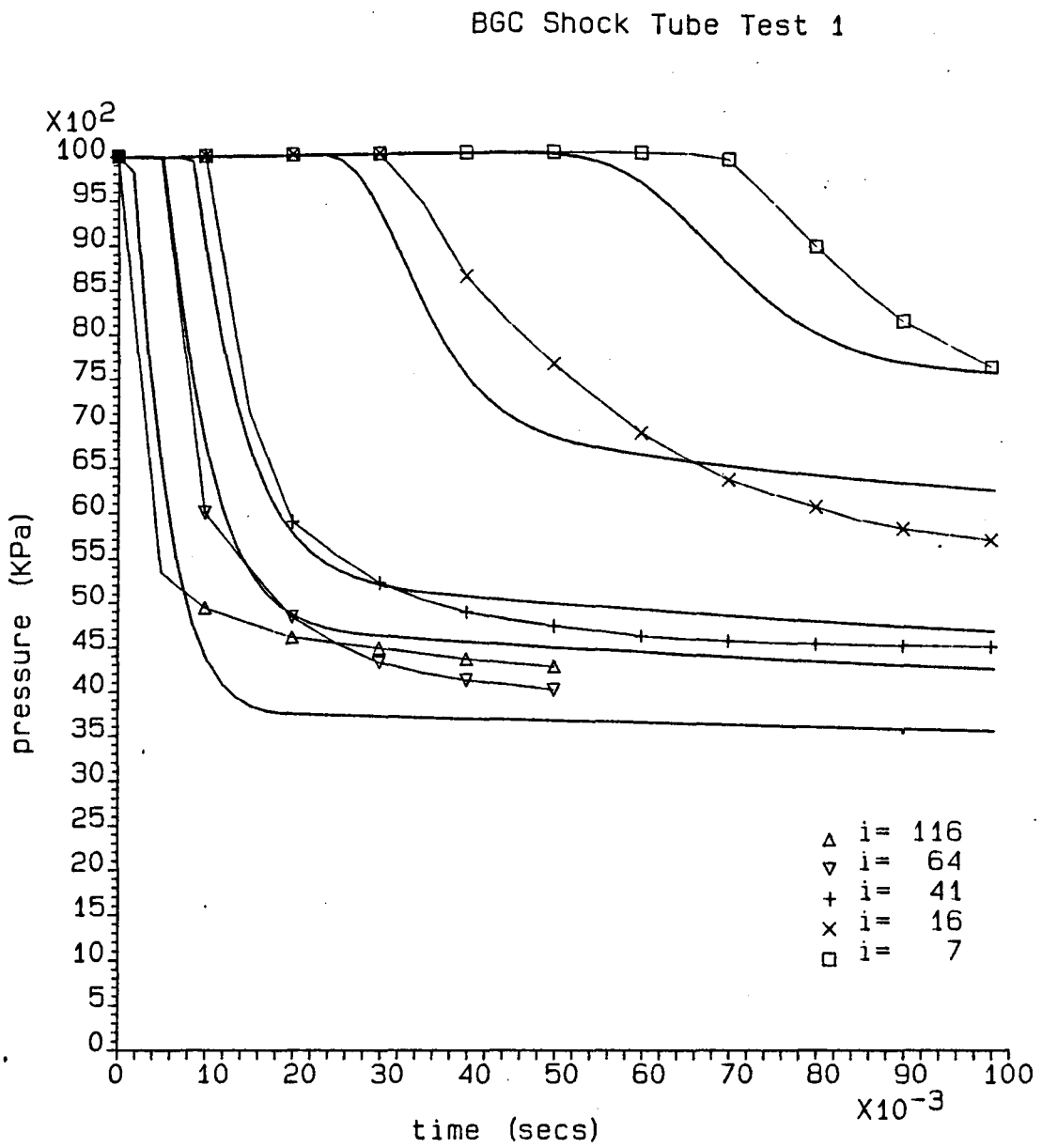
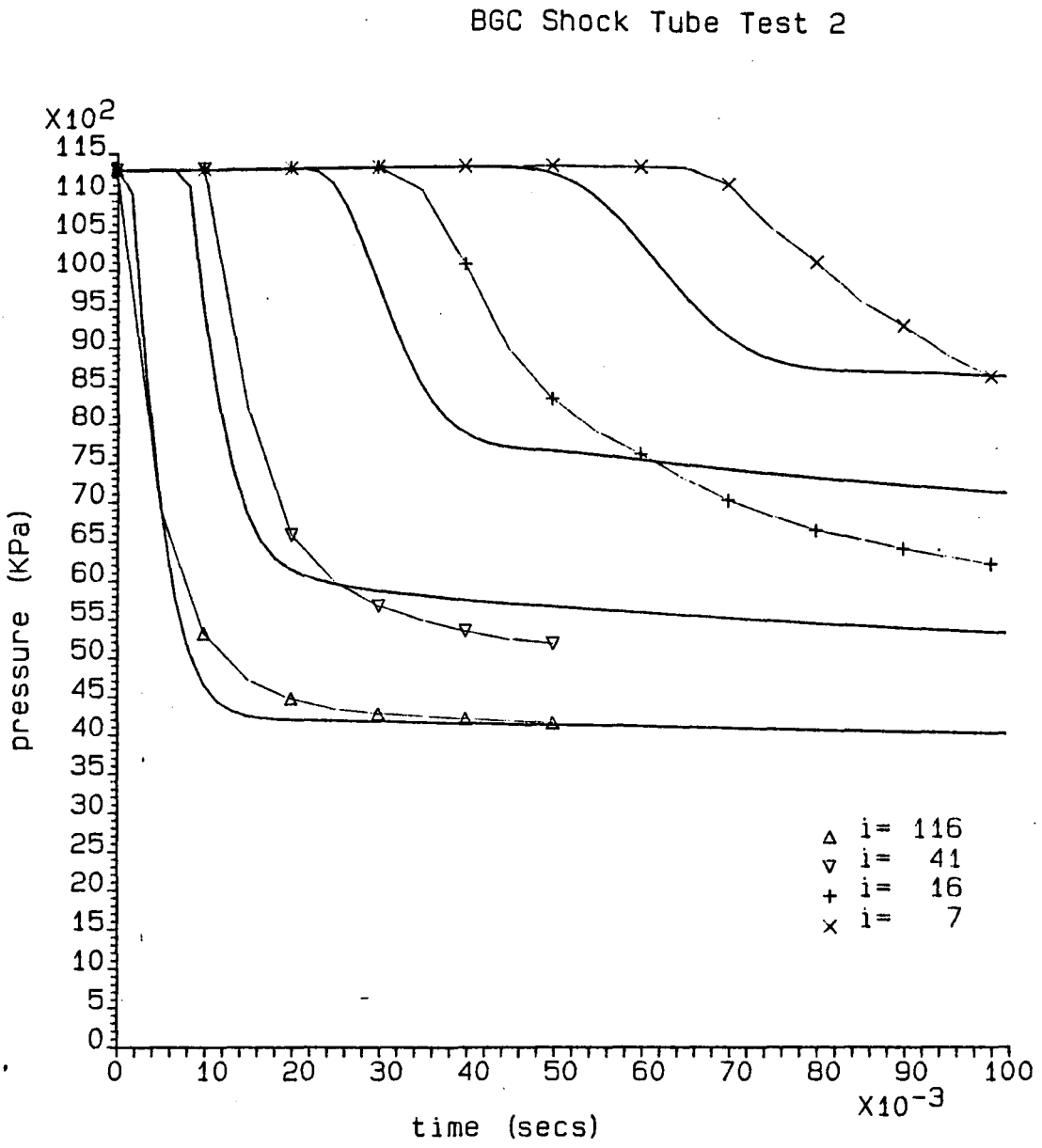
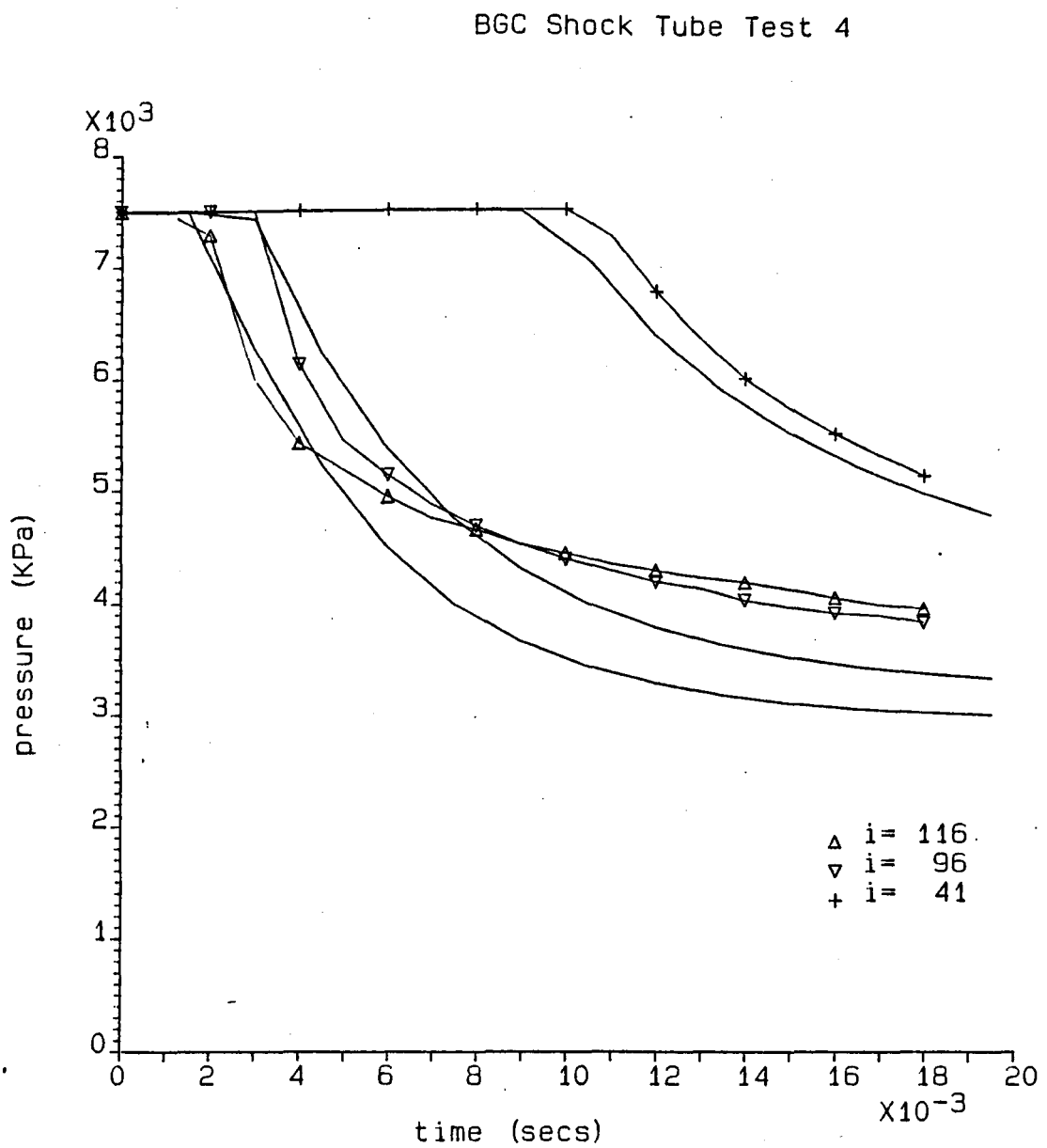


Figure 7.10. P x t Graph for British Gas Shock Tube Test 1



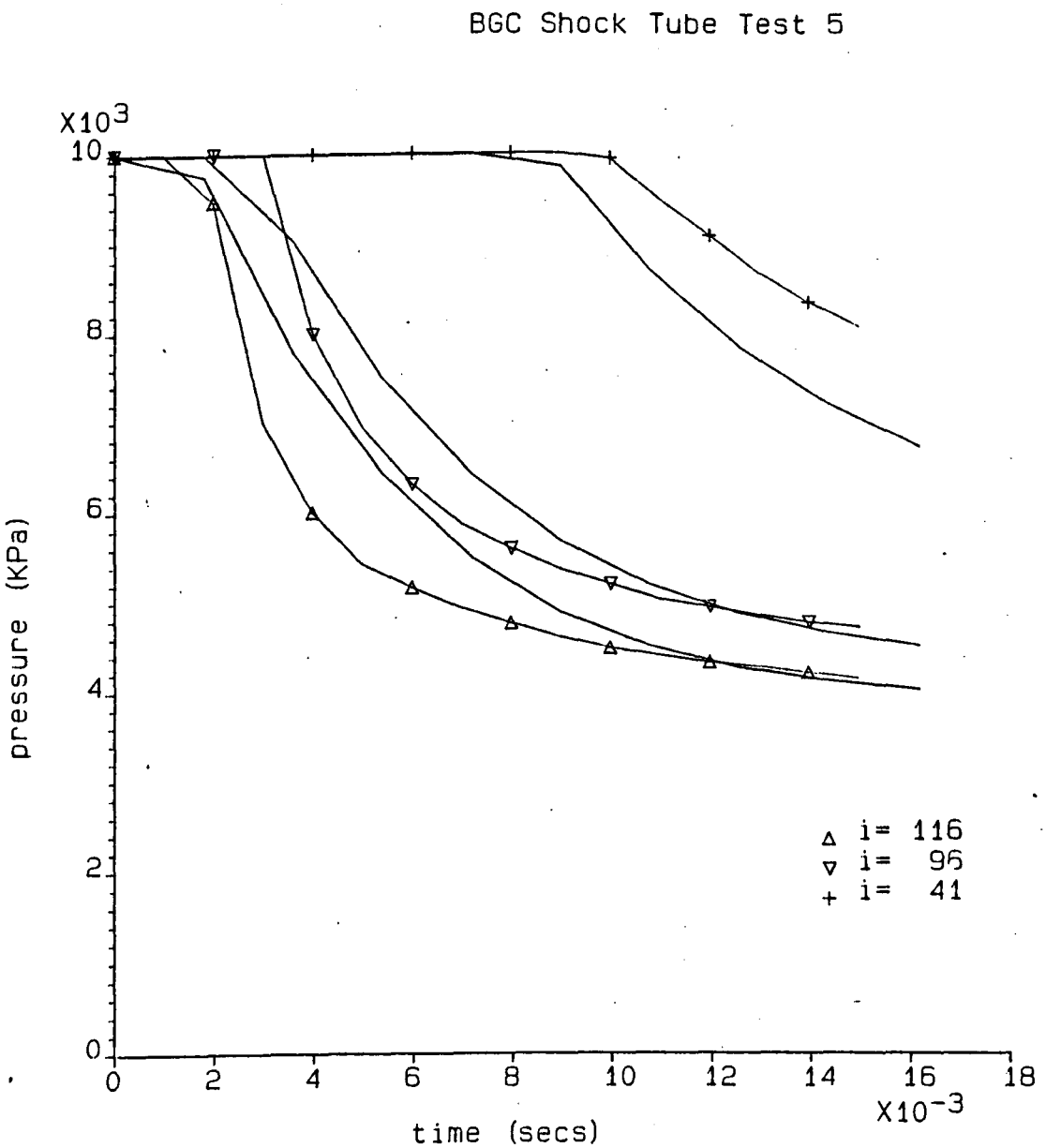
f = 0·01
St = 0·0027

Figure 7.11. P x t Graph for British Gas Shock Tube Test 2



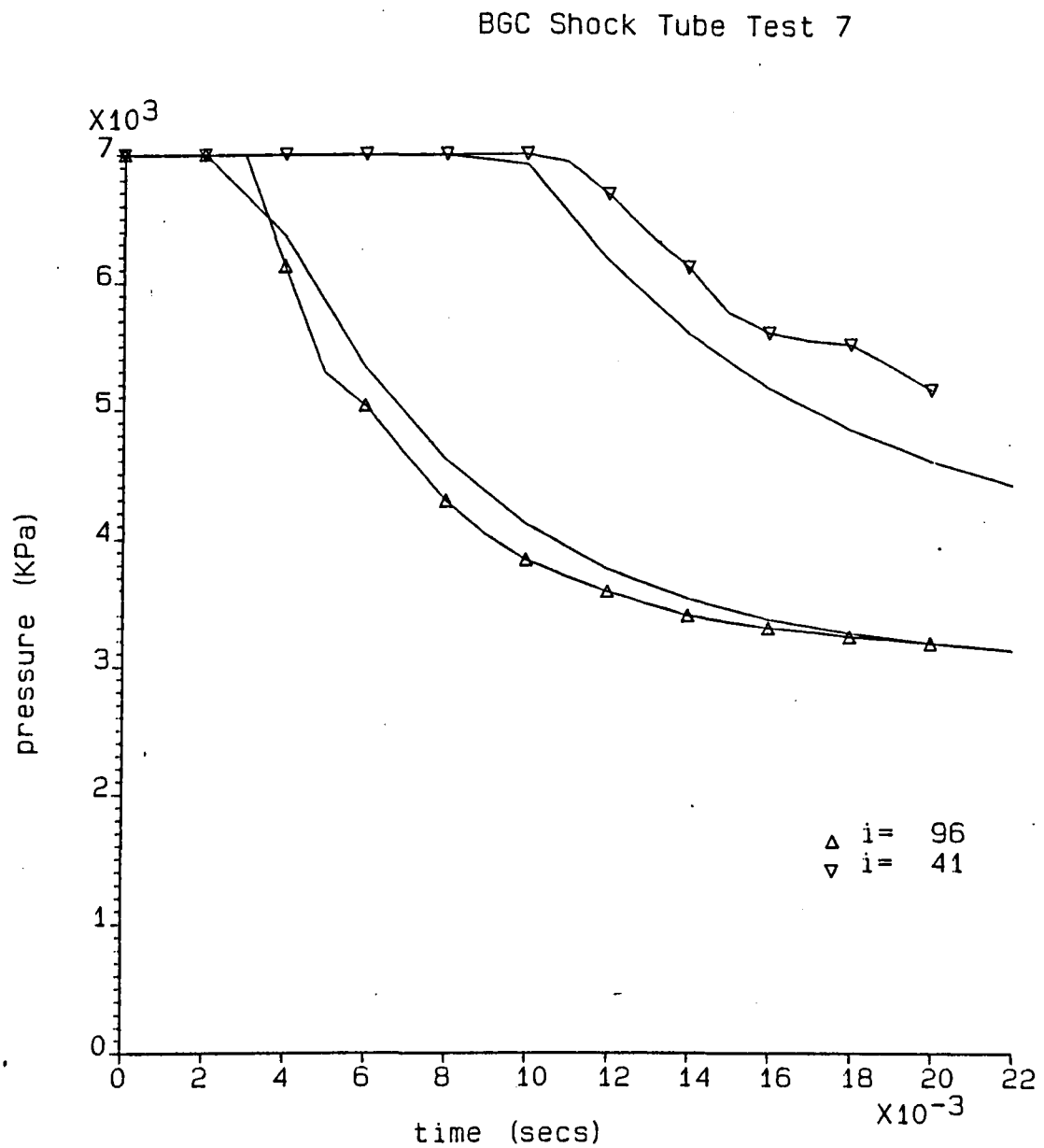
f = 0·03
St = 0·0027

Figure 7.12. P x t Graph for British Gas Shock Tube Test 4



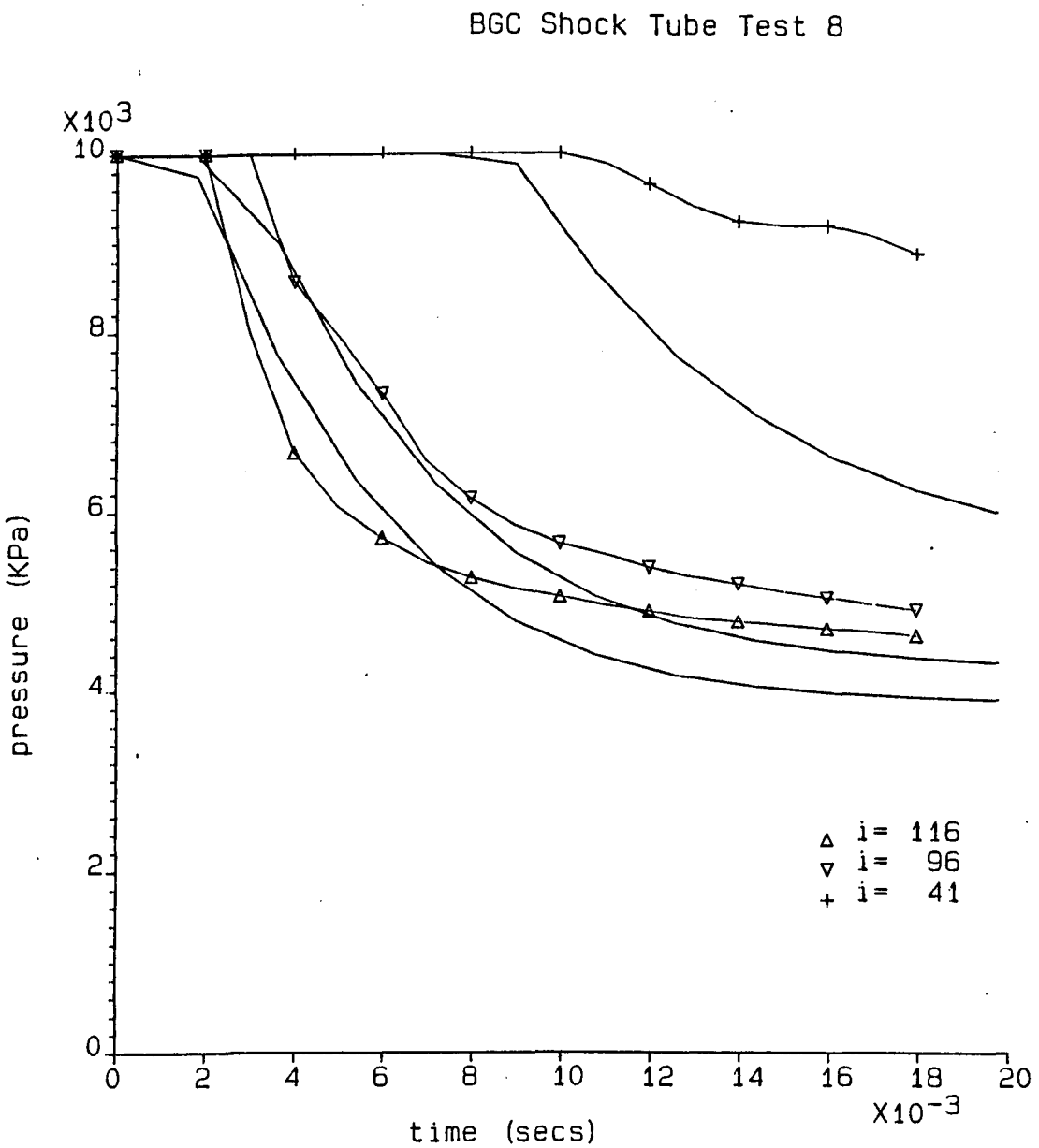
f = 0·03
St = 0·0027

Figure 7.13. P x t Graph for British Gas Shock Tube Test 5



f = 0·03
St = 0·0027

Figure 7.14. P x t Graph for British Gas Shock Tube Test 7



f = 0·03
St = 0·0027

Figure 7.15. P x t Graph for British Gas Shock Tube Test 8

7.4. FOOTHILLS PIPELINES (YUKON) FULL SIZE RESULTS

Stable solutions were obtained for each of the four tests for which experimental data was available (NABTF1, NABTF3, NABTF4, and NABTF5). A grid size of 0.01 m near the break was used with friction factor and Stanton Number of 0.018 and 0.0027 respectively. The break characteristics were the same as those used for the shock tube data except for NABTF3 when it was necessary to use 80 steps instead of 40 (for stability).

To obtain the theoretical curves, different lengths of time step were used for each test. For NABTF1, Δt = 0.74 s, for NABTF3, Δt = 0.72 s, for NABTF4, Δt = 0.65 s, and for NABTF5, Δt = 0.75 s.

For each test, pressure x time and pressure x wavespeed graphs were obtained for the pipe sections both east and west of the break. In order to distinguish between the two pipe sections, the i values corresponding to positions west of the break are preceded by a 1 and the i values corresponding to positions east of the break are preceded by a 2. The transducer positions corresponding to the values marked on the graphs (Figures 7.16 - 7.27) were as follows:

NABTF1

Position	i value
4.21 m west of break	1172
25.30 m west of break	1139
36.00 m west of break	1123
46.63 m west of break	1106
4.21 m east of break	2128
14.17 m east of break	2143
25.02 m east of break	2160
36.06 m east of break	2177
46.57 m east of break	2194

NABTF3

Position	i value
4.25 m west of break	1172
14.08 m west of break	1157
24.90 m west of break	1140
35.71 m west of break	1123
46.49 m west of break	1106
4.25 m east of break	2128
14.08 m east of break	2143
24.88 m east of break	2160
35.67 m east of break	2177
46.42 m east of break	2194

NABTF4

Position	i value
7.61 m west of break	1167
17.61 m west of break	1151
26.94 m west of break	1137
37.64 m west of break	1120
46.78 m west of break	1106
7.61 m east of break	2133
17.83 m east of break	2149
28.73 m east of break	2166
39.42 m east of break	2183
48.81 m east of break	2197

NABTF5

Position	i value
5.18 m west of break	1171
15.53 m west of break	1155
26.11 m west of break	1138
36.79 m west of break	1122
47.40 m west of break	1105
5.18 m east of break	2129
15.18 m east of break	2145
26.13 m east of break	2162
36.97 m east of break	2179
47.30 m east of break	2195

The experimental data for all the full size test graphs were obtained in the same manner as for the BGC shock tube tests, using the Foothills Pipelines experimental curves.

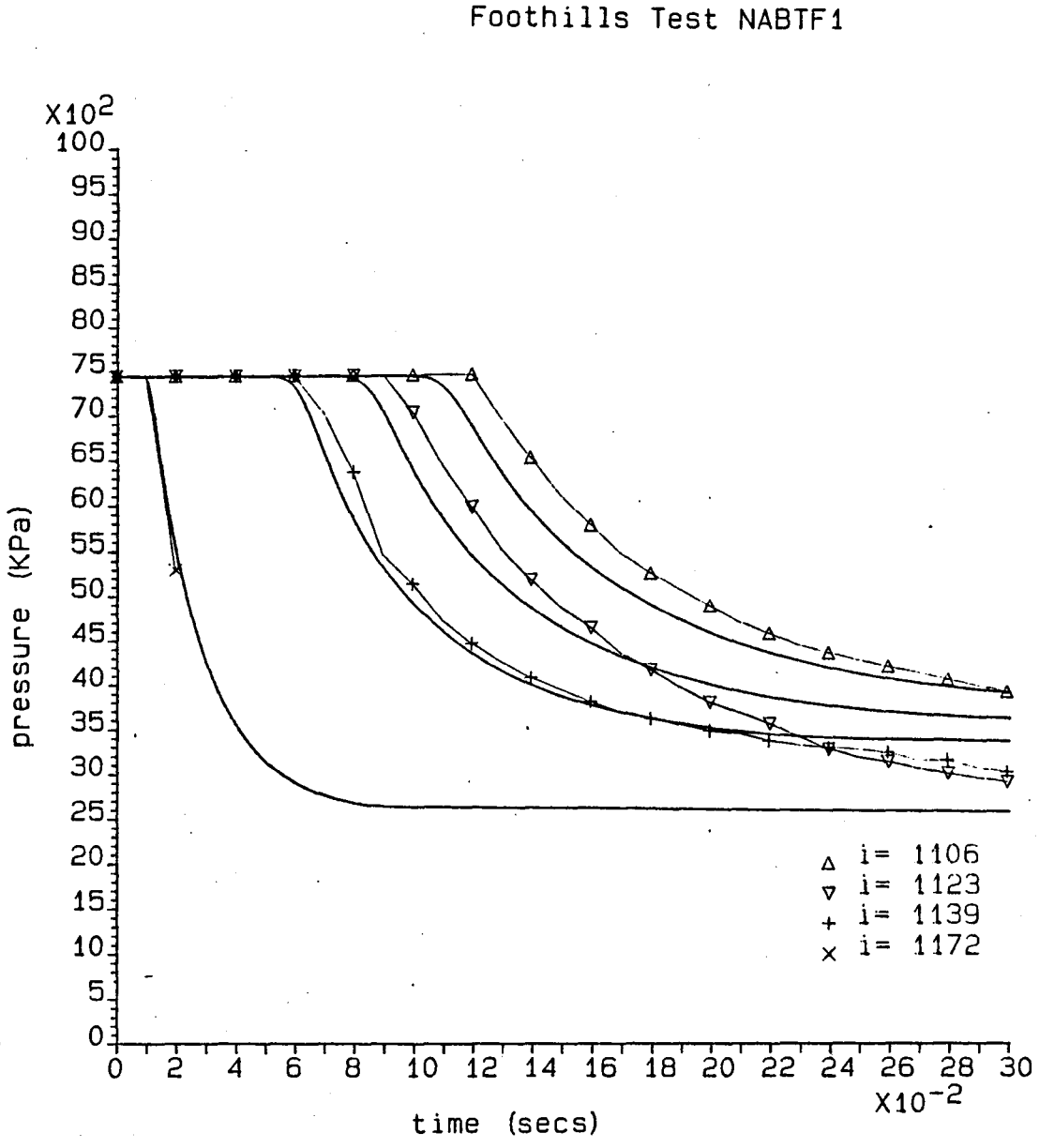


Figure 7.16. P x t Graph for Foothills Test Results NABTF1 (West)

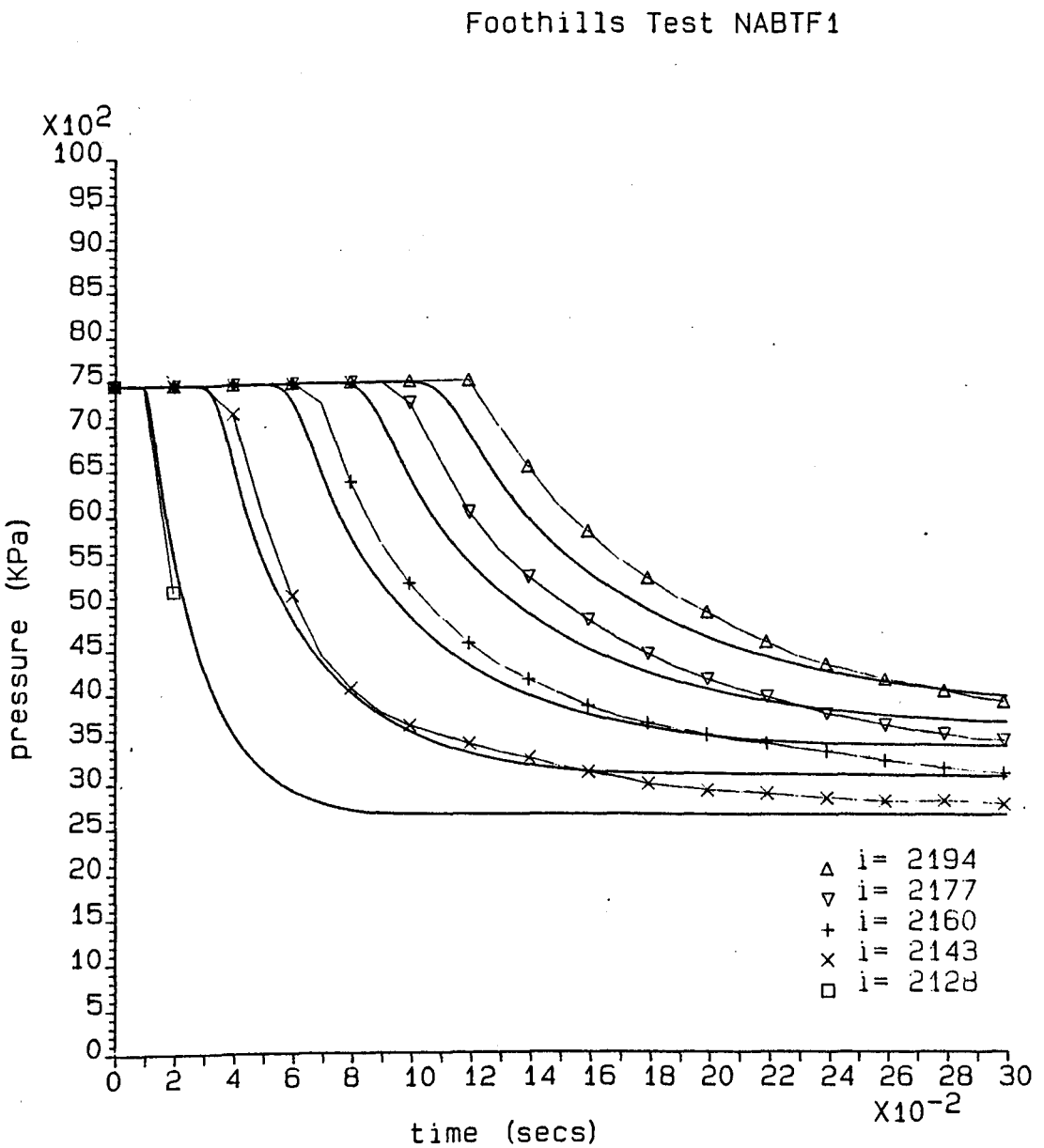


Figure 7.17. P x t Graph for Foothills Test Results NABTF1 (East)

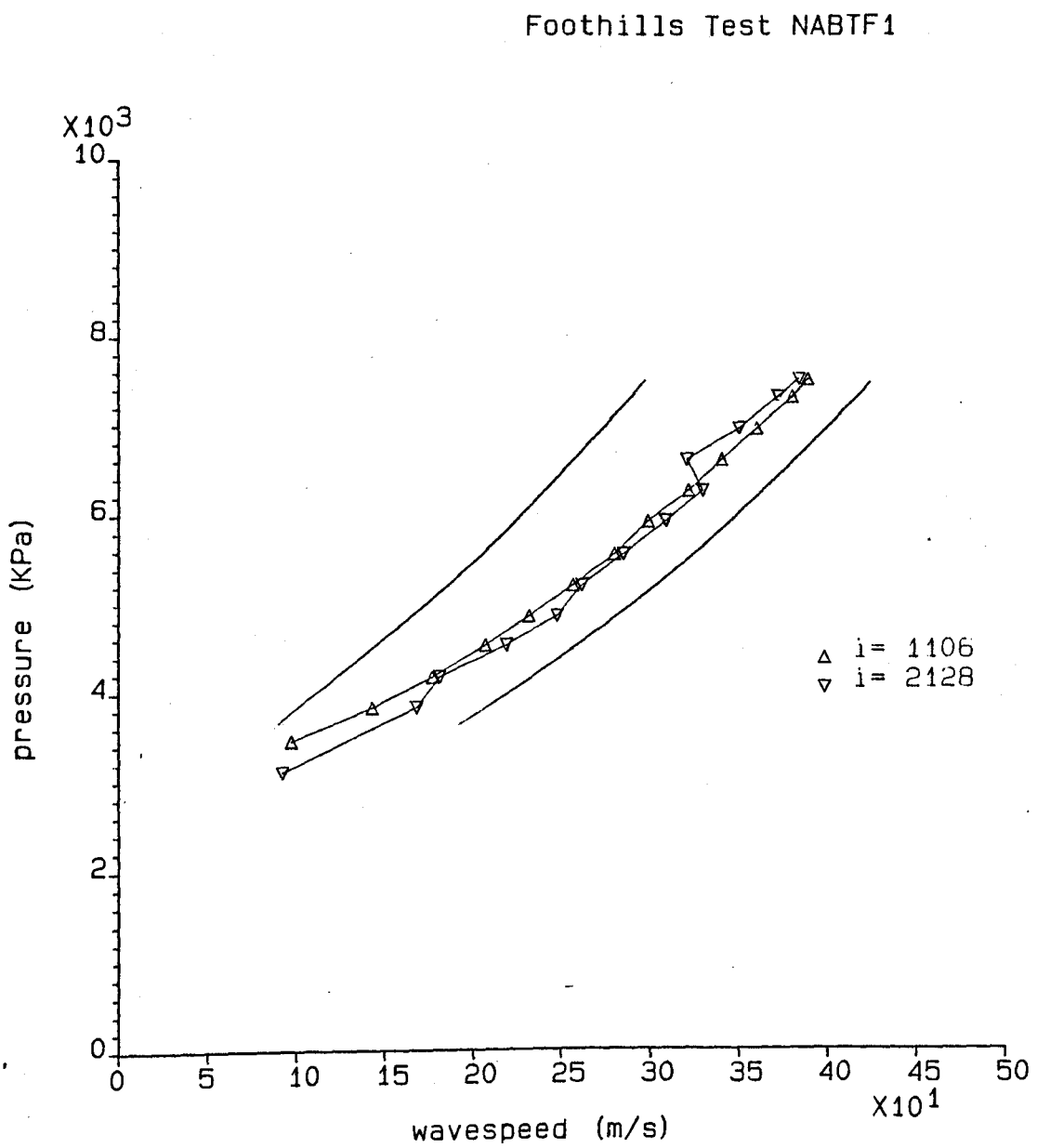


Figure 7.18. P x ω Graph for Foothills Test Results NABTF1

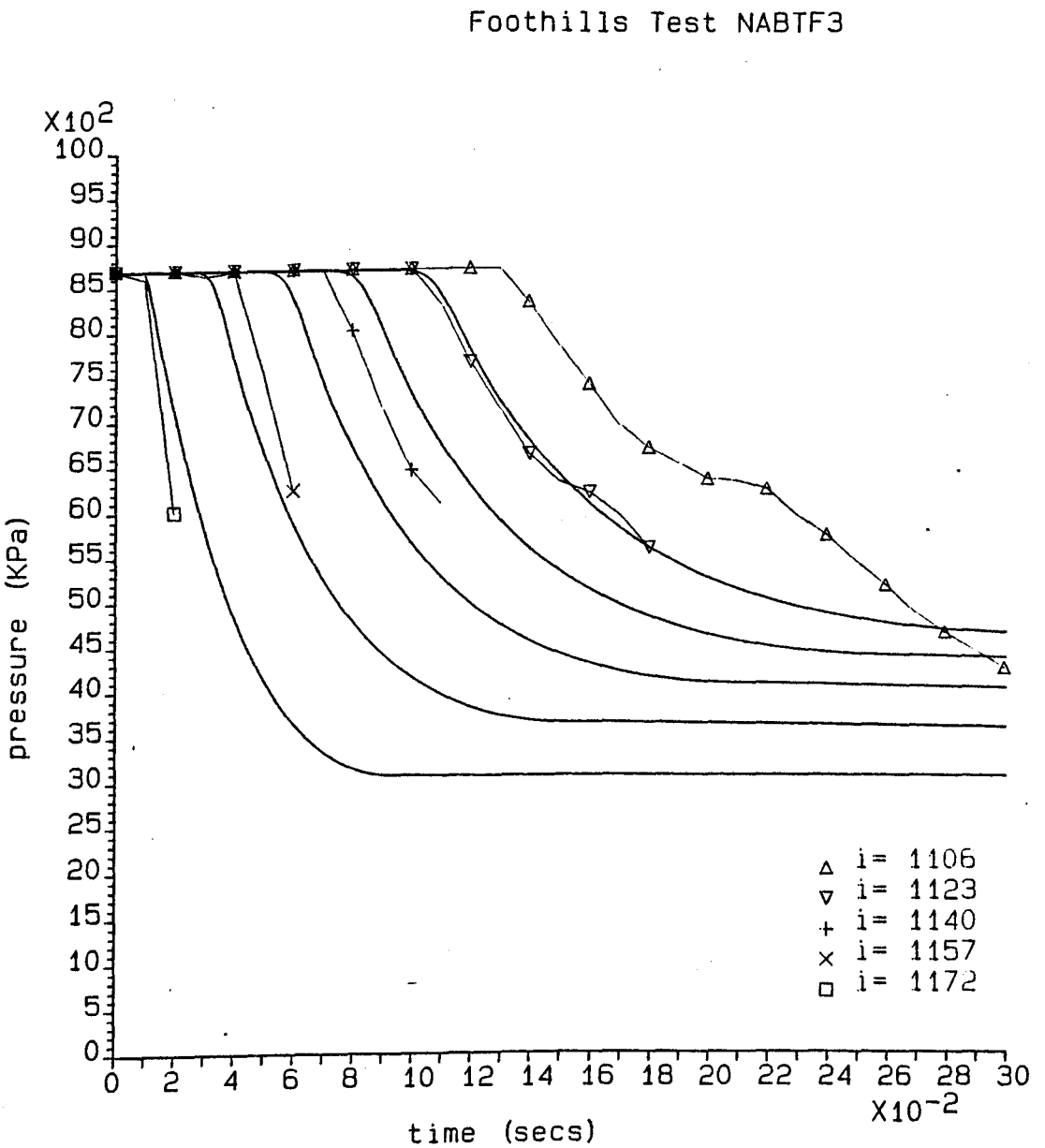


Figure 7.19. P x t Graph for Foothills Test Results NABTF3 (West)

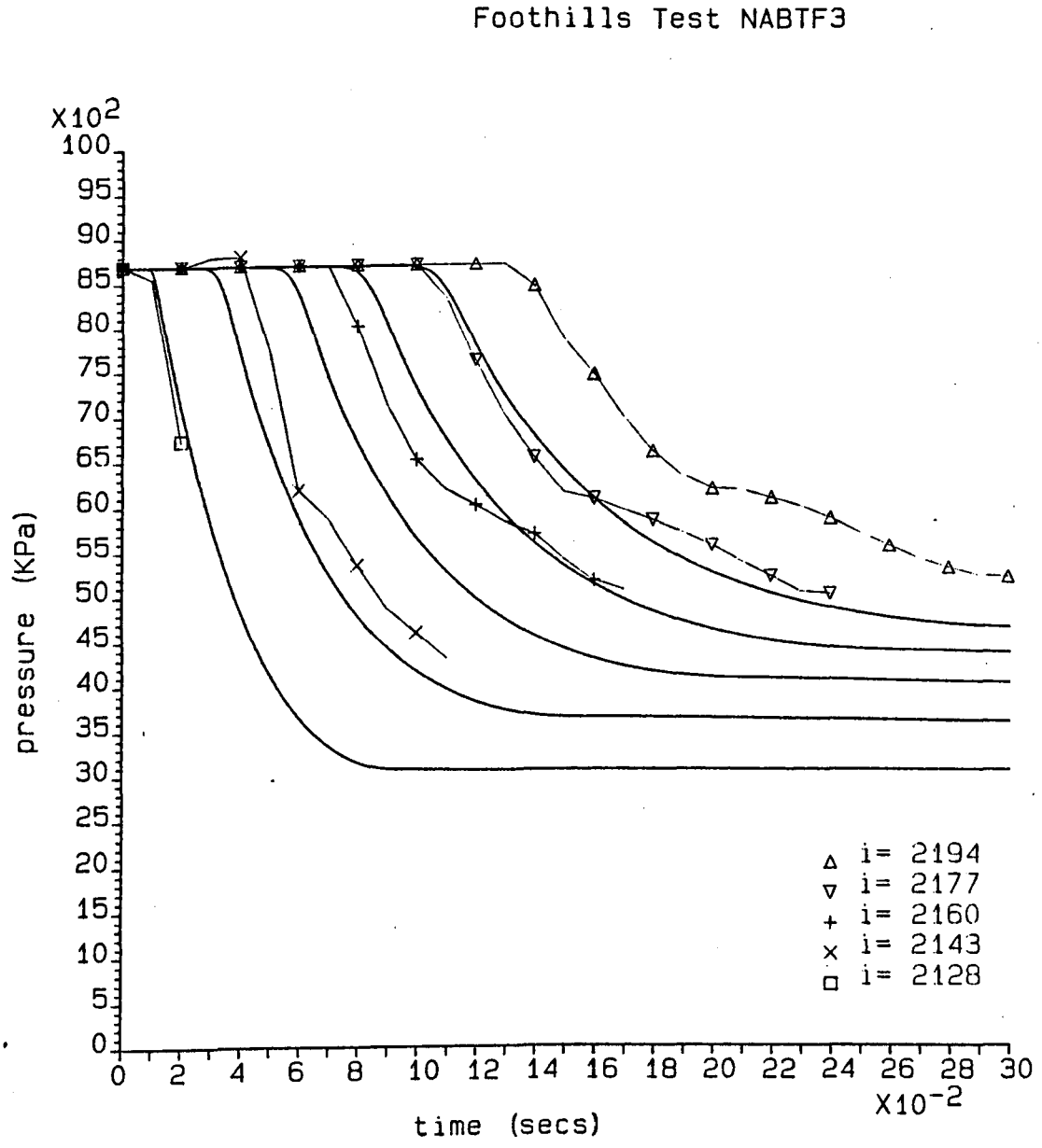


Figure 7.20. P x t Graph for Foothills Test Results NABTF3 (East)

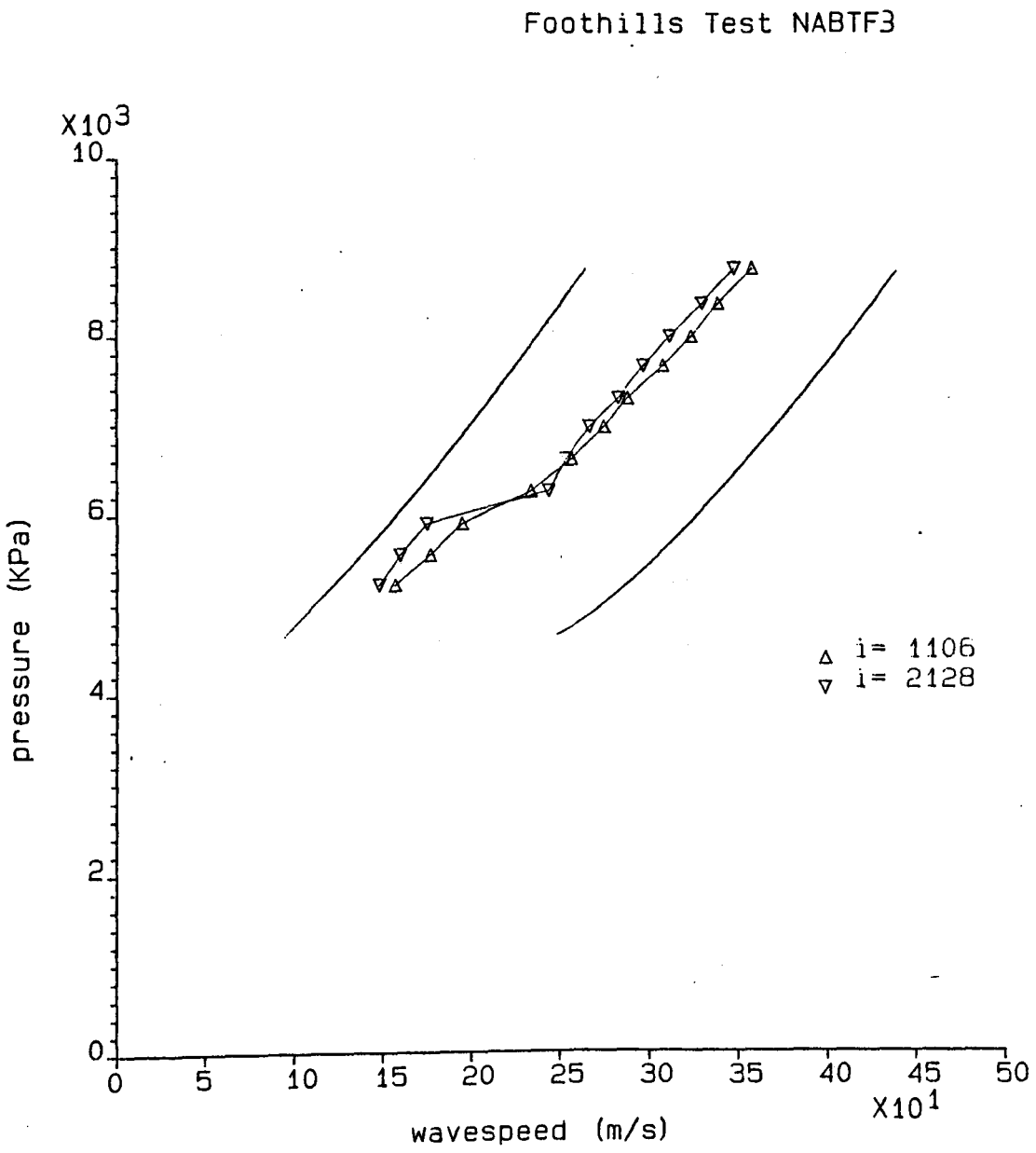


Figure 7.21. P x ω Graph for Foothills Test Results NABTF3

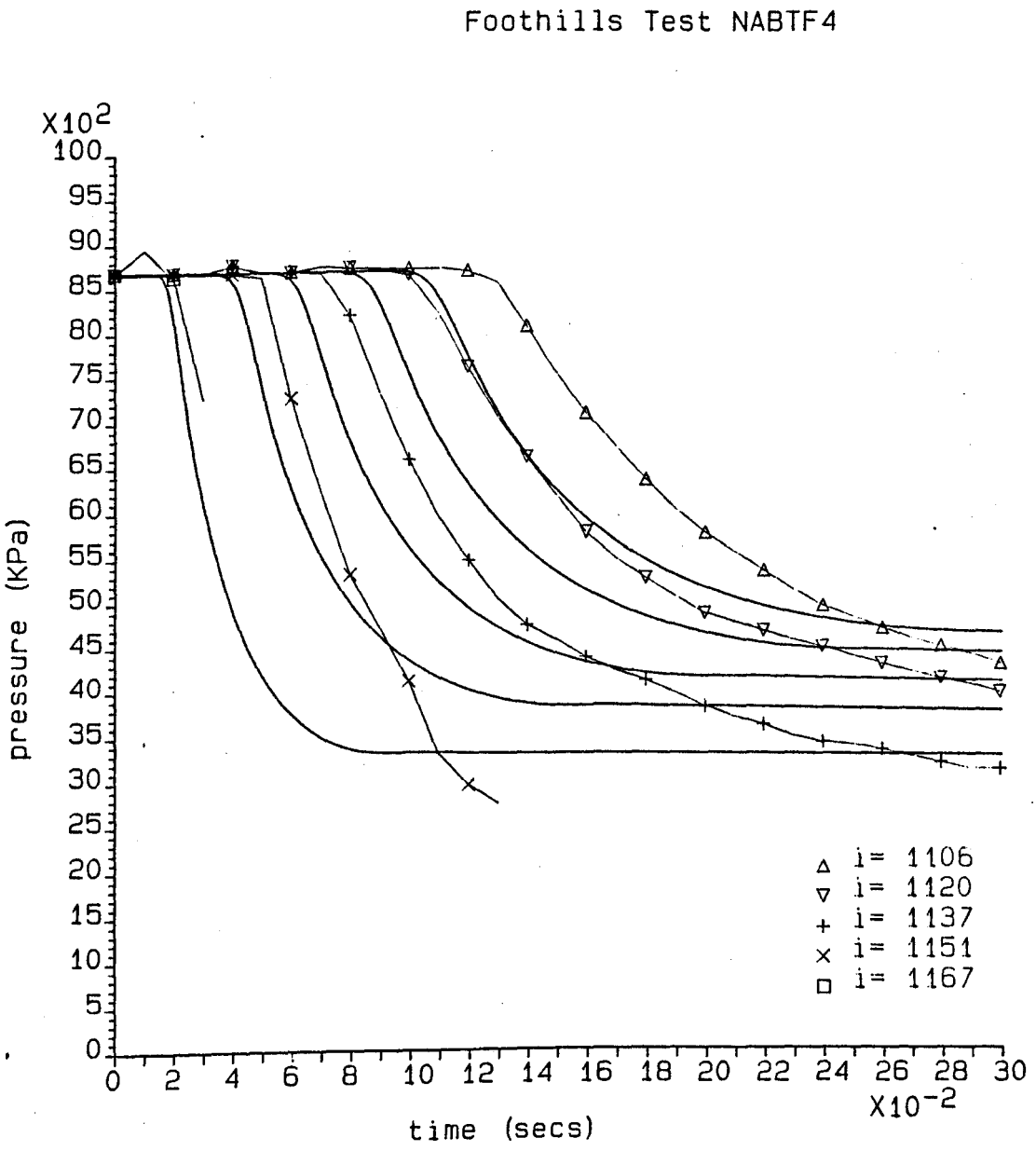


Figure 7.22. P x t Graph for Foothills Test Results NABTF4 (West)

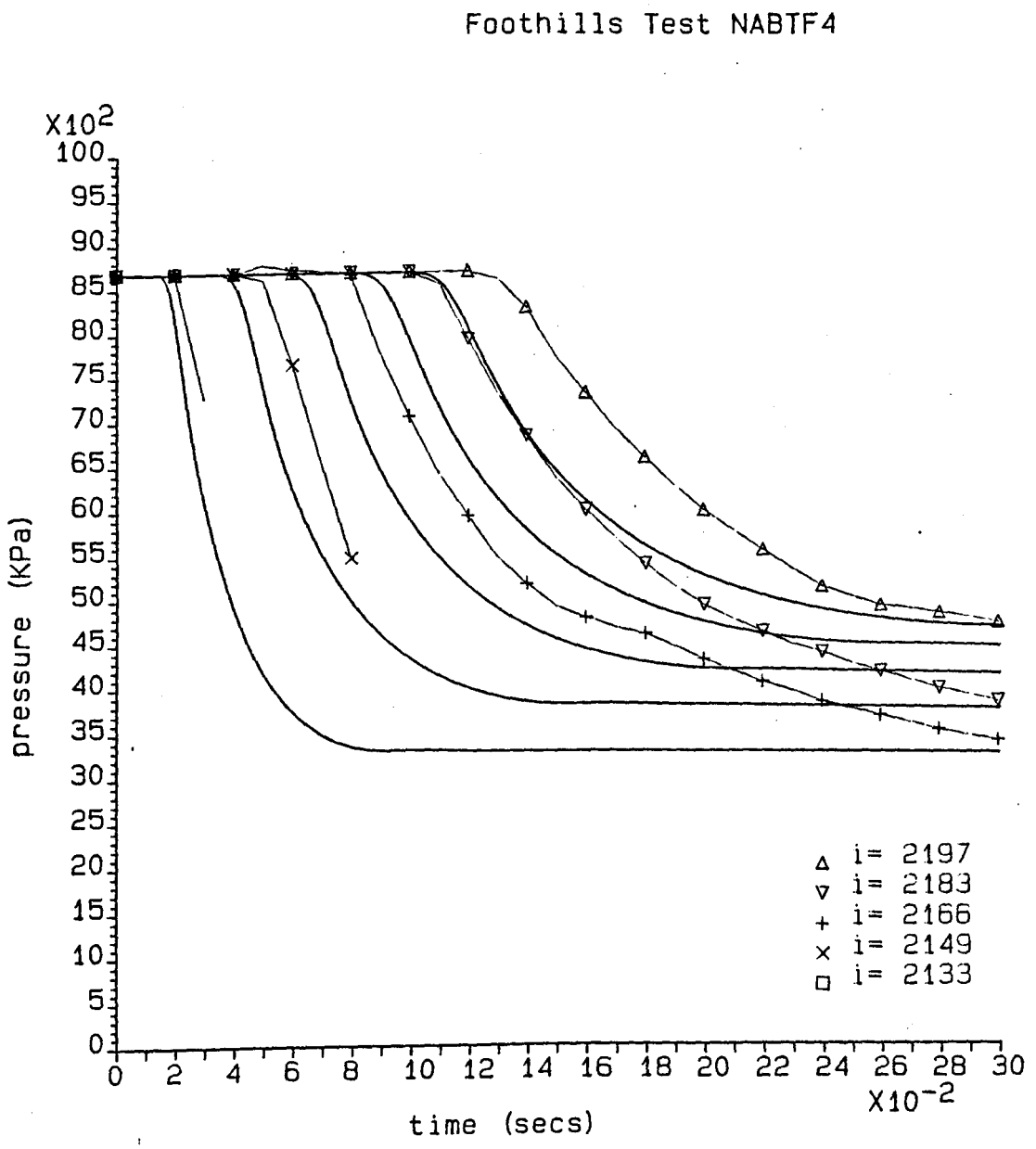


Figure 7.23. P x ω Graph for Foothills Test Results NABTF4 (East)

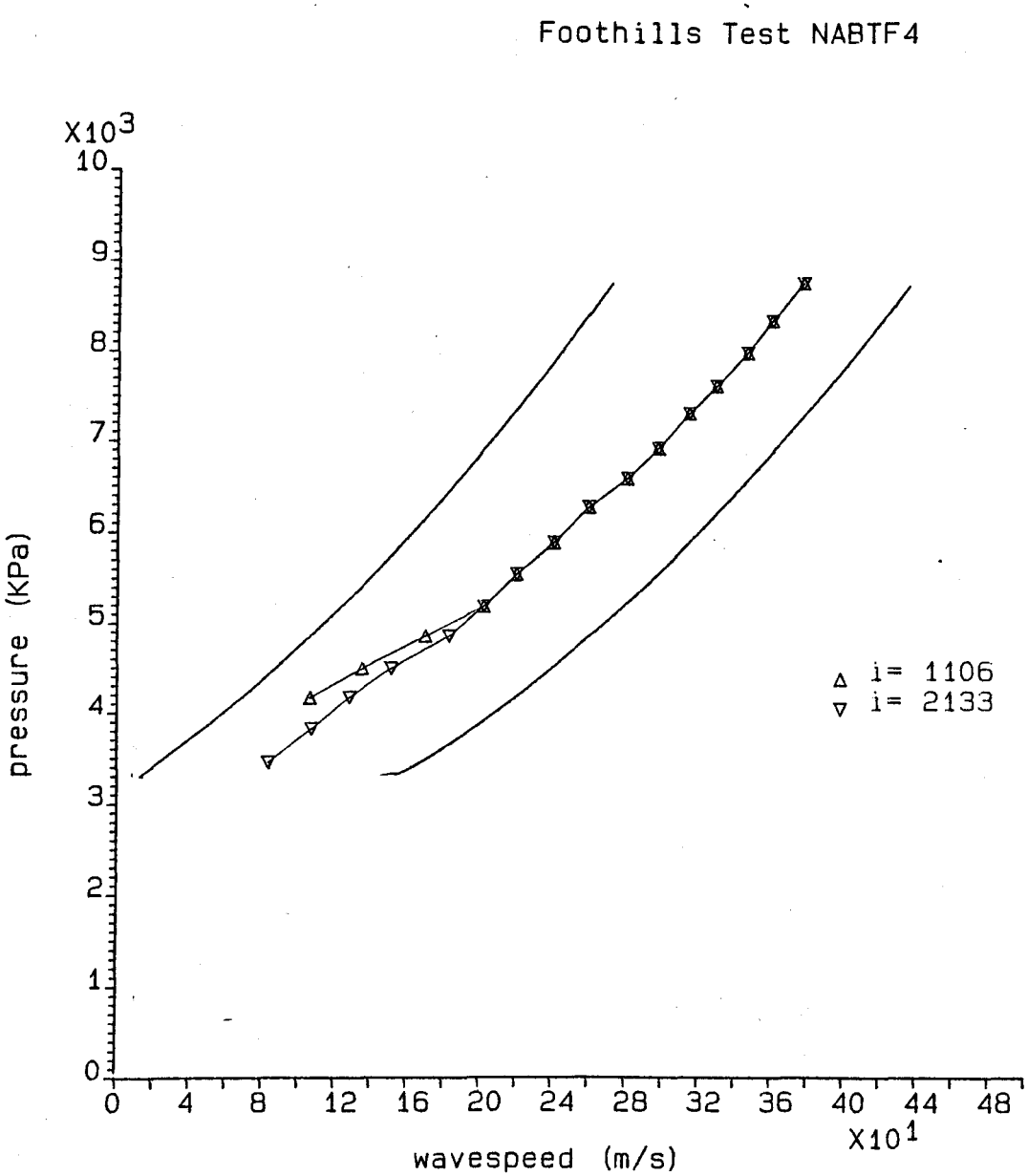


Figure 7.24. P x ω Graph for Foothills Test Results NABTF4

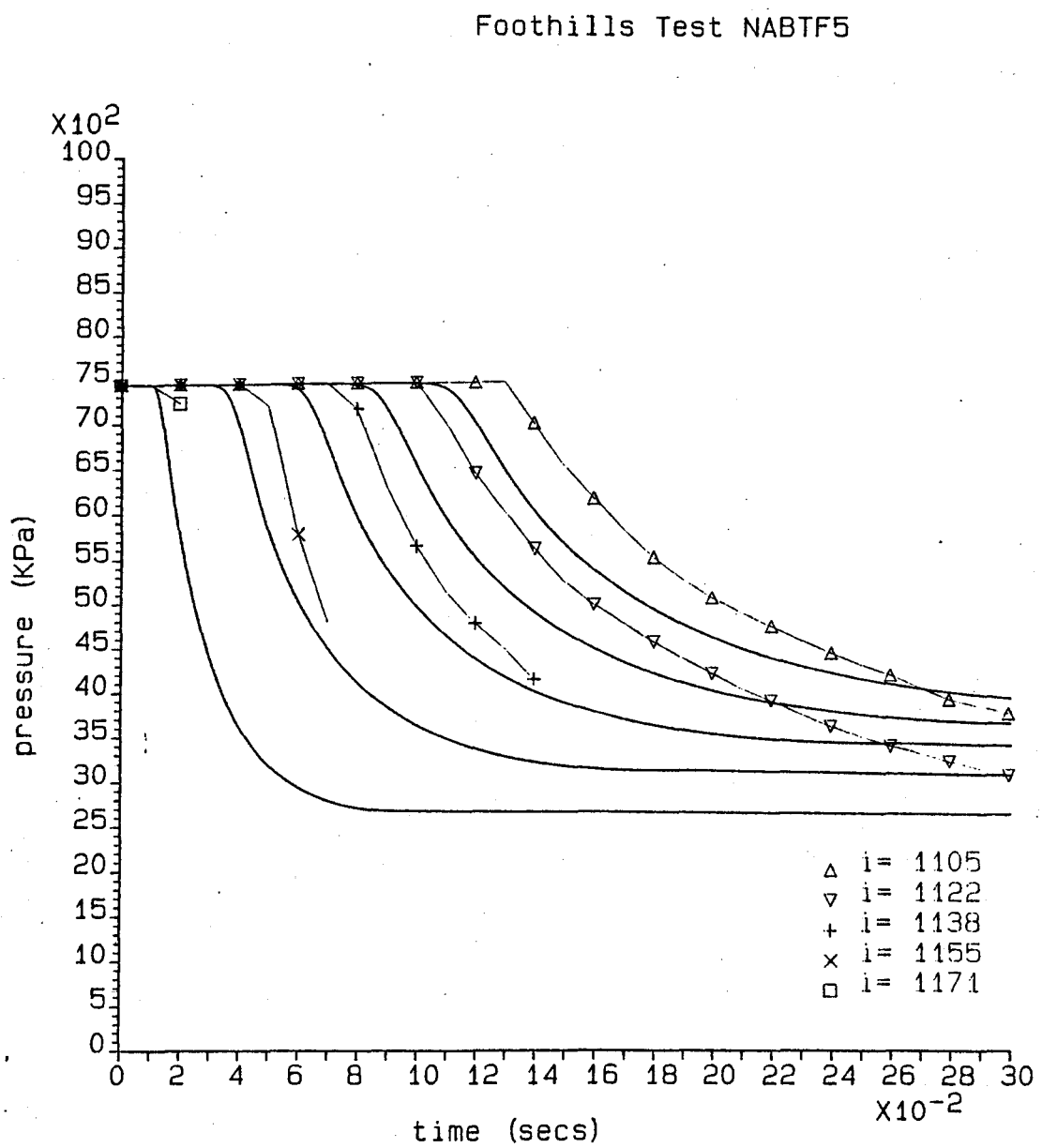


Figure 7.25. P x t Graph for Foothills Test Results NABTF5 (West)

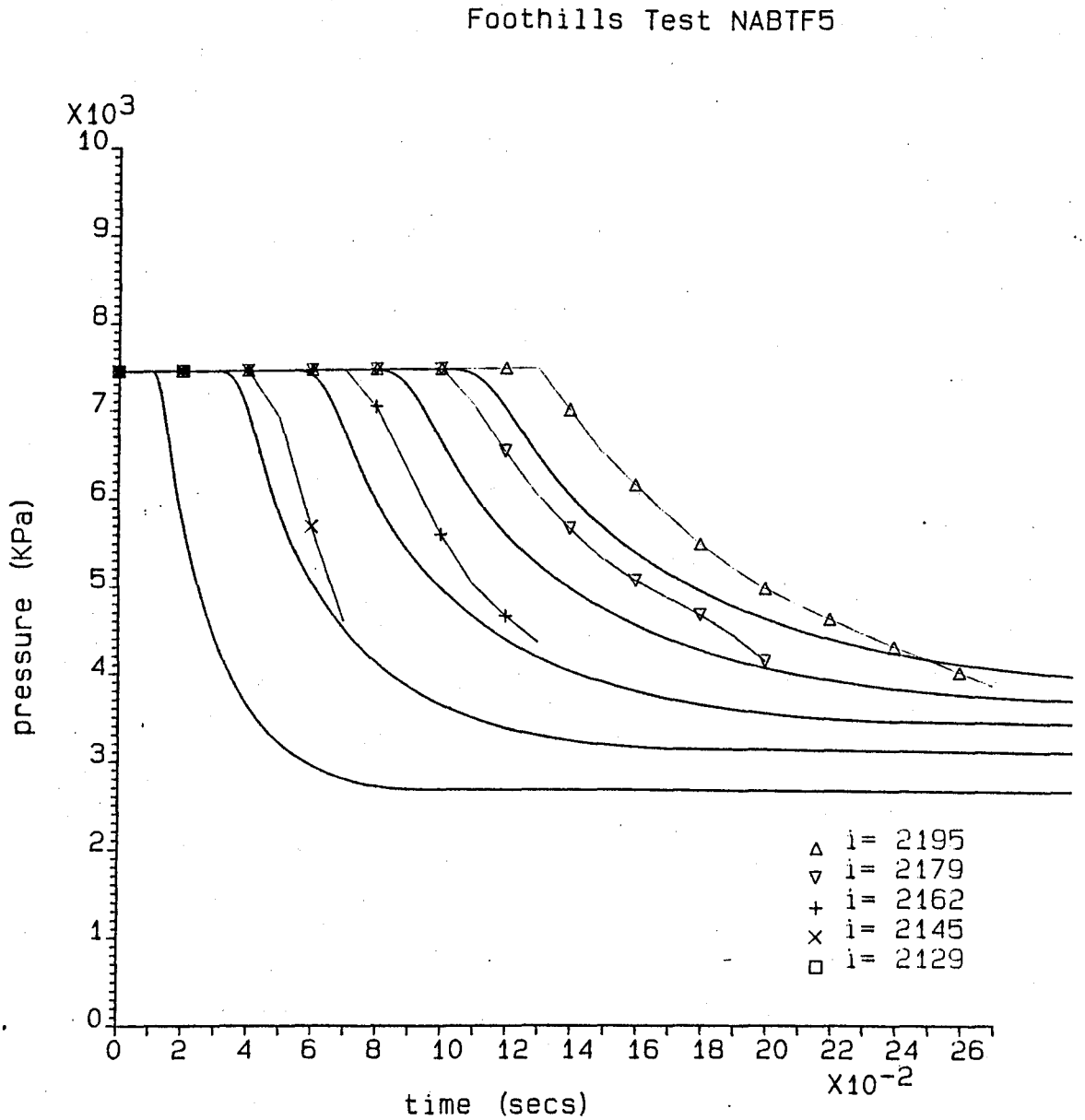


Figure 7.26. P x t Graph for Foothills Test Results NABTF5 (East

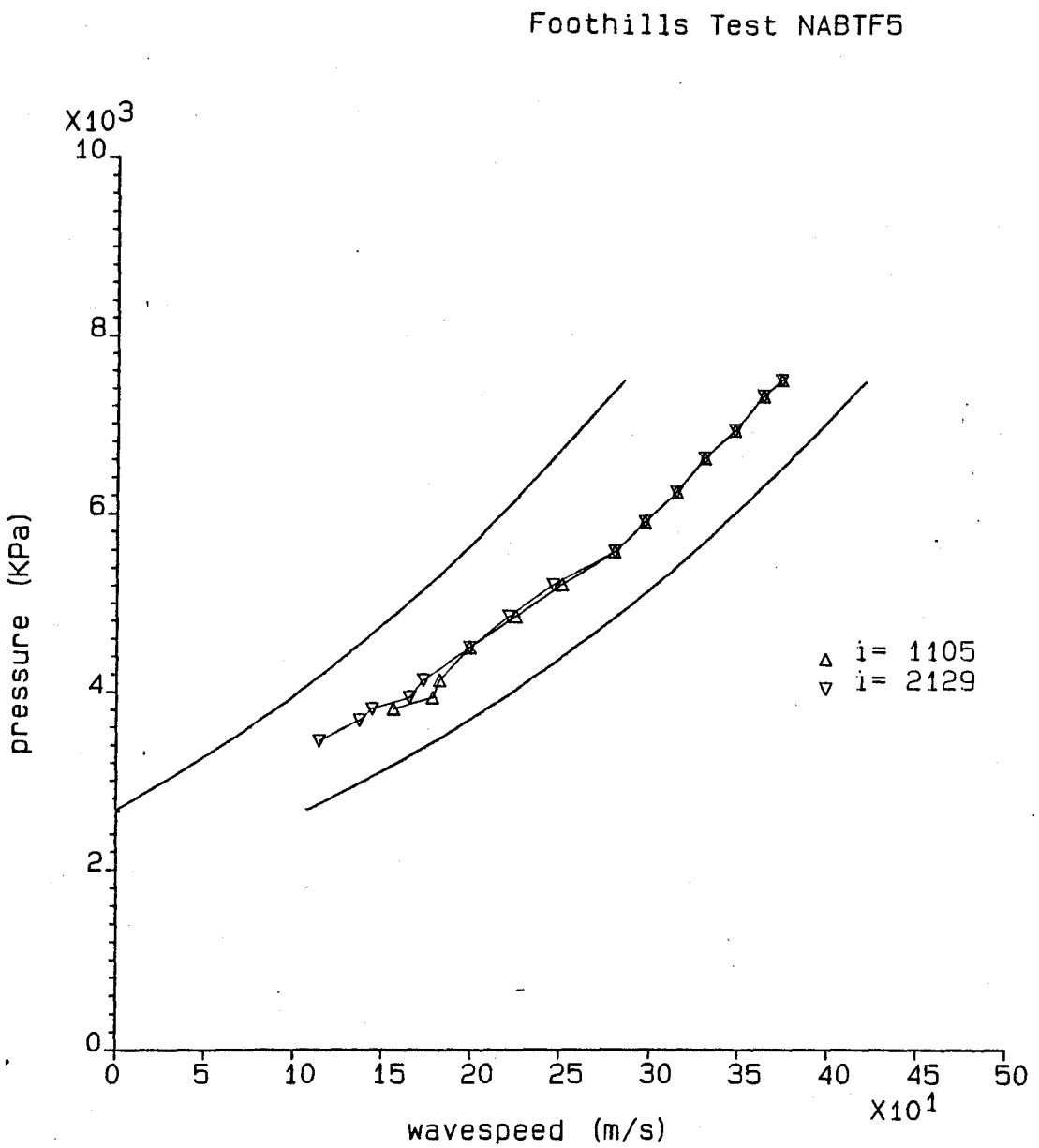


Figure 7.27. P x ω Graph for Foothills Test Results NABTF5

CHAPTER 8

DISCUSSION

8.1. GENERAL DISCUSSION OF RESULTS

The graphs obtained using the transient program in conjunction with the graphics program clearly showed some similarities to those obtained from experimental data. Due to the nature of the results it was decided that primarily a qualitative rather than quantitative assessment would be most suited. The graphs were examined and compared with each separate experimental data source and the following observations were made.

8.1.1. Groves' Shock Tube Results

Figures 7.1 to 7.3 illustrate how the computer program was successful in predicting the maximum (isentropic) and the minimum (isothermal) possible wavespeeds for any particular point along the pipe. In a real situation the actual wavespeeds will be between these two extremes as is confirmed by the experimental data points. Although these results do not give an unequivocal indication of the accuracy of the computer model, they do at least show that the calculations being performed by the program are of the right order.

With the pressure x wavespeed curves predicted for argon gas (Figure 7.2), it was noticed that the isentropic wavespeed curves started to vary at lower pressures for the different positions along the pipe. This trend is not clearly visible in the experimental data.

The natural gas isentropic curves (Figure 7.3) show a more distinct separation with the different grid points. In Figure 7.3 it is also possible to identify a similar separation pattern at the lowest wavespeeds region of the experimental curves. It is thought that this separation of curves is occurring when a maximum flow situation is happening in the pipe. In this situation the pressure upstream from the break will be greater than that at the break in order for the flow to overcome friction. Therefore the pressure will be higher for lower i values than for higher values. Figures 7.1 to 7.3 also confirm that these final pressures are dependent on the gas being used.

It might have been useful to have incorporated actual wavespeed calculations in the computer program so that a direct comparison could have been made between the theoretical and experimental readings. However, this would have necessitated calculating the actual speed of sound given by:

$$a^2 = \left[\frac{dP}{d\rho}\right]_{actual}$$

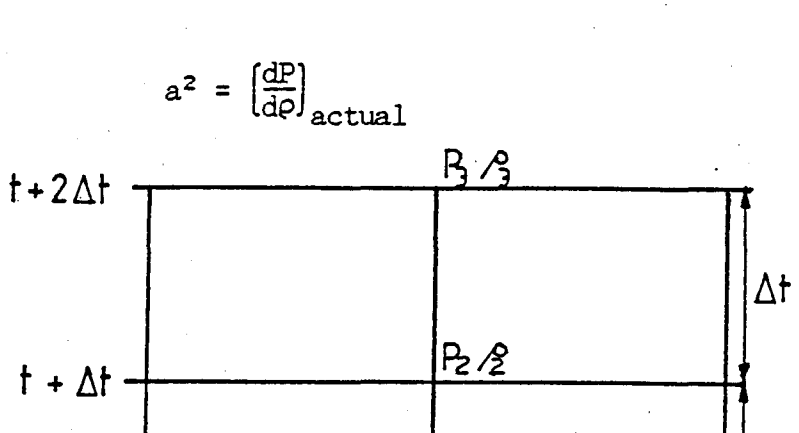


Figure 8.1. Approximation of dp/dρ on a finite grid.

With reference to Figure 8.1, $dP/d\rho$ between t and $t + \Delta t$ could be approximated by:

$$\frac{\Delta P}{\Delta \rho} = \frac{P_2 - P_1}{\rho_2 - \rho_1}$$

But, in order to obtain an estimate for the speed of sound at time $t + \Delta t$, a further approximation would be necessary of $dP/d\rho$ between $t + \Delta t$ and $t + 2\Delta t$ and the two values averaged.

This procedure would involve storing both pressure and density values for three consecutive time steps at each grid point. The required computer storage facilities would therefore be increased and extra calculations would be involved (hence increasing the program run time). It was thus decided that for the purposes of this project, actual wavespeed calculations were not justifiable.

8.1.2. British Gas Shock Tube Results

Despite the poor quality of these graphs, Figures 7.4 to 7.9 do show that both the theoretical and experimental results produce similar curves. The lack of smoothness and continuity in the experimental results arises from the difficulties experienced in obtaining sufficiently close experimental data points from British Gas graphs. The graphics program connected these data points with straight lines which produced some jaggedness when the points were not close enough together. This also occurred with the theoretical points in Figures 7.6 to 7.9 since the transient program was only producing output every 2 ms. Although these theoretical curves

could have been improved by decreasing the length of time step, this may have led to an increase in the accumulative round-off error in the results (as discussed further in section 8.2).

An alternative method of smoothing the curves (both experimental and theoretical) would be to employ a graphics program which connected the points using best-fit curves instead of straight lines. This was experimented with but it was found to produce an unrealistic apparent pressure rise just before the expansion wave reached each pressure transducer position. Hence it was thought that the straight line method of connecting the points was more suitable.

In the graphs for BGC tests 1 and 2, it was noticed that the theoretical curves tended to begin their pressure drop too early. This was more noticeable the further the transducer was from the break. Possible reasons for this were explored.

Firstly, it was thought that since no information was available to indicate how the pressure transducers were triggered, maybe the response times had not been accurately accounted for which was causing the delay. However, since the theoretical and experimental results for the pressure transducer 90 ft from the break differed by as much as 15-20 ms for the arrival of the expansion wave, it was deduced that there was probably another reason. The fact that British Gas's theoretical curves coincided better with their experimental results at the start of the expansion wave also suggested that there may be a problem with the theoretical results.

The theoretical wavespeeds at the initial pressures were therefore calculated from the pressure x time graphs and these were then compared with the isentropic and isothermal wavespeeds calculated in the program. The results from this investigation are shown below:

	calculated wavespeed (m/s)	isentropic wavespeed (m/s)	isothermal wavespeed (m/s)
BGC 1	543	457	264
BGC 2	583	496	252

Thus, there did appear to be a problem in the theoretical model regarding the wavespeed of the initial expansion wave since it should be less than the isentropic wavespeed. This falsely high wavespeed would account for the noticeable difference in positions of the pressure x time curves for the transducers at 45 ft and 90 ft from the break. Further investigation of this problem was therefore carried out using the Foothills Pipelines (Yukon) Ltd. results.

Another curious feature of the graphs obtained from British Gas data was that the final theoretical pressures reached in the shock tube are higher than those recorded experimentally in tests 1 and 2 but lower than those recorded in tests 4, 5, 7 and 8. This could be due to inaccuracies incurred in the calculation of the equalization pressure at the break. These would arise because the conditions are assumed to be isentropic and the gas is assumed to be a perfect gas at the break point. Therefore any change in state brought about by the rapid expansion of the gas would not be accounted for and this could affect the results.

The adjustment of the friction factor for the different gas mixtures (Figures 7.10 to 7.15) was found to have little effect. A slight improvement was noticed in each graph, but this improvement did not justify the changes made. The initial estimates used for friction factor and Stanton Number were therefore acceptable.

The accuracy of the experimental data points used for analysing these graphs could not be ascertained from contact with British Gas. However, on enquiring into the reasons for the experimental pressure transducer traces actually crossing one another in tests 1 and 4 (Figures 7.4 and 7.6), British Gas did report (Jones 1988) that they "believe (but could not conclusively confirm) that PT2 was malfunctioning". This implied that the transducer 8 ft from the break in test 1 was malfunctioning and the transducer 4 ft from the break in test 4 was malfunctioning. British Gas also indicated that they were not entirely satisfied with the results from any of their tests using the methane/propane mixtures (used in Figures 7.6, 7.7, 7.12 and 7.13) and so some caution should be exercised when using these experimental results.

8.1.3. Foothills Pipelines (Yukon) Ltd. Full Size Results

With the improved quality of these graphs (Figures 7.16 - 7.27) compared with that of the previous pressure x time graphs, the effect of the apparent difference in wavespeed between the theoretical and experimental curves was more clearly identifiable. With these graphs it was possible to take more accurate readings for calculating the actual wavespeeds at the initial pipe pressures. Only

the three transducers furthest from the break were examined in order to minimise the reading errors. The following results were obtained:

Test	Calculated Theoretical Wavespeeds (m/s)				Calculated Experimental Wavespeeds (m/s)			
	1	2	3	Mean	1	2	3	Mean
NABTF 1	472	471	466	470	410	398	389	399
NABTF 3	472	469	462	468	359	359	358	359
NABTF 4	491	487	486	488	364	388	397	383
NABTF 5	470	474	462	469	372	372	368	371

In all the graphs the calculated theoretical wavespeeds were greater than the experimental values. Comparing these wavespeeds with the initial isentropic and isothermal wavespeeds shown in Figures 7.18, 7.21, 7.24 and 7.27:

Test	Theoretical Wavespeed (m/s)	Experimental Wavespeeds		Isentropic Wavespeed (m/s)	Isothermal Wavespeed (m/s)
		From p x t graph	From p x w graph		
NABTF1	470	399	389	427	297
NABTF3	468	359	354	441	266
NABTF4	488	383	379	438	273
NABTF5	469	371	375	405	286

From the above table it can be seen that all the calculated theoretical wavespeeds are higher than the isentropic wavespeeds. This situation is not physically possible. One likely cause would be that a second-order numerical interpolation has been used in the

calculation of the theoretical wavespeeds. This can result in theoretical rapid pressure changes occurring prematurely. The problem may be resolved by using only linear interpolation in the calculation procedures.

Apart from the problems incurred in the wavespeed calculations of the program, the theoretical curves showed good agreement with those obtained experimentally. In most of the graphs it would appear that the final pressures calculated theoretically are slightly higher than those recorded. However, the theoretical model cannot account for the crack propagation along the pipe. Theoretically the break is modelled as a ruptured diaphragm but the rupturing of the pipe sections produces a lengthwise crack in the pipe which covers a finite length and opens up the pipe (as shown in Figure 8.2).

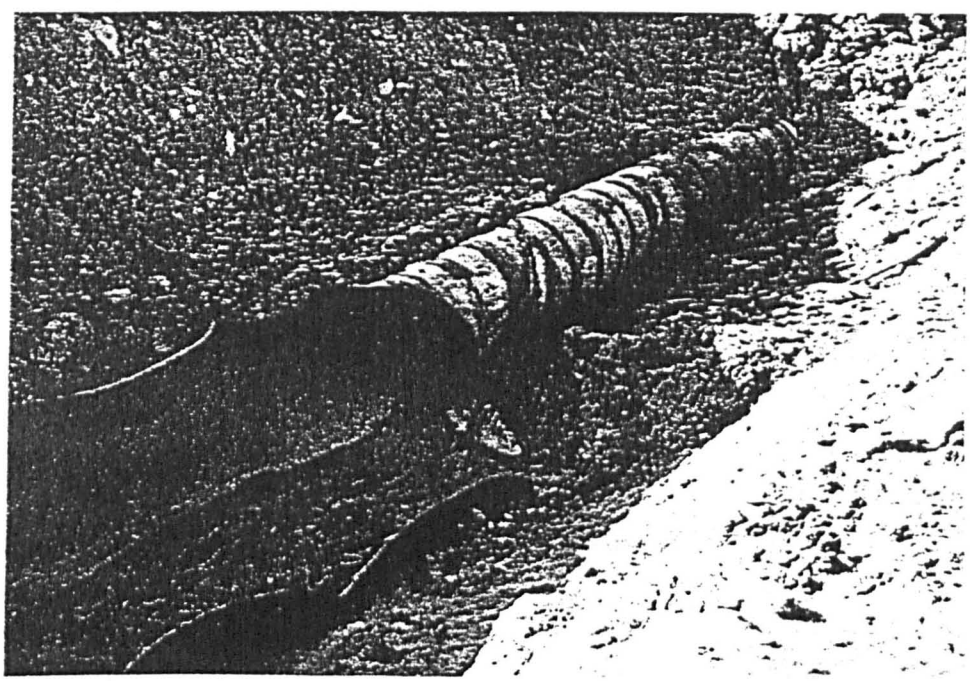


Figure 8.2. Pipe rupture in a full size pipe

This effectively reduces the pressure at the break point to atmospheric pressure and moves the point at which the equalization pressure would occur along the pipe. Hence the pressures at all points along the pipe would be proportionally reduced. This therefore accounts for the recorded pressures being lower than those predicted by the theoretical model. This crack propagation would also account for less levelling off in the experimental pressure data since the pressure at any transducer point would continue to fall as the crack tip approached.

Finally, in test NABTF1 it was noticed that, as with British Gas tests 1 and 4, the experimental pressure transducer traces actually crossed. In this case it was thought that the pressure transducer at i = 1123 (36 m west of the break) was most likely to be in error since its trace was out of line with the other traces. The cause of this error was reported to be a temperature-related zero drift during testing. This may also be the cause of the same problem encountered by British Gas.

8.2. DISCUSSION OF ERRORS

When modelling any real life situation, certain discrepancies between the theoretical prediction and the actual event are bound to arise. These discrepancies may be categorized into those arising from errors incurred in the calibration, measurement and recording of the experimental data, those due to necessary assumptions made in the theoretical modelling, and those due to errors inherent in the numerical modelling procedure.

The errors in the experimental data arise from two main sources; namely those errors involved in the measuring of the data and those that are incurred when the experimental data is transferred for use with this project. With Groves' data, no information was available detailing the accuracy of the measuring procedures so this could only be roughly estimated after examining the accuracies with which British Gas and Foothills Pipelines (Yukon) Ltd. could obtain their data. Also with Groves' data, assumptions had to be made regarding the shock tube material (so that a reasonable value for the friction factor could be estimated) and the effective rupture time of the diaphragm (so that the break boundary conditions could be decided upon). In none of the experimental data sources was there any indication of an effective rupture time, probably due to the considerable problems associated with measuring such a small finite time. Since, however, for the computer simulation the time taken for the pressure at the break point to fall to its equalization pressure must be estimated, this could be a possible error source.

Through private communication with J.E. Falcus at British Gas, it was established that the pressure measuring system (as a whole) that they used in their shock tube tests was believed to be accurate to within 5%. Foothills Pipelines (Yukon) Ltd. were confident that their initial pressures were within a ±5 kPa limit implying pressure transducer accuracy of 0.2%. They did not, however, estimate the overall accuracy of the transient pressure measuring system. It was therefore assumed that the accuracies of the pressure measuring systems used by Groves and by Foothills Pipelines (Yukon) Ltd. would be similar to the 5% estimated by British Gas.

In order to use this experimental data, it was necessary to transfer it from the graphical form provided to a numerical form for the computer. In each case this involved superimposing a grid onto the experimental data graph and reading off the co-ordinates. To reduce the possible inaccuracies involved with this process, the graphs of Groves et al. were enlarged by a factor of 2 prior to extracting co-ordinates. It was then possible to determine the pressure ratio to within ±0.01 and the wavespeed to within 5 m/s. Similarly, for the British Gas graphs, the estimated accuracy with which pressure could be determined was ±1 bar (after the initial sharp pressure drop); and the accuracy of the time scale was estimated to be ±2 ms for the methane/ethane curves and ±0.3 ms for the methane/propane and natural gas curves. In the graphs supplied by Foothills Pipelines (Yukon) Ltd. pressure could be determined to ±50 kPa, time to ± 2 ms, and wavespeed to ±3 m/s. Thus the maximum probable reading error involved in the transfer of data from any of these experimental graphs would be 5%.

- In the development of the theoretical model many assumptions were necessary in order to produce a viable computer program. Firstly, in the formation of the basic equations, one dimensional flow was assumed which could introduce slight errors when the rapid expansion occurs in the pipe. Other assumptions included were that the pipe wall was inelastic and that there were no localised frictional losses due to pipe joints, bends, etc. The use of these assumptions should not, however, cause any significant deviation from the actual recorded event in the experimental examples used.

A further source of error in the theoretical model could be due to the equation of state that was used. A relatively simple equation was chosen and since this was used for various gas mixtures over a considerable pressure - temperature range (incorporating in some instances a change of state) some discrepancies were expected. However, as with many of the consequences of the theoretical assumptions, it was not possible to quantify this error source.

The use of a constant specific heat also introduced errors since in practice it will vary over the considerable temperature and pressure range that is encountered following a pipe rupture. This variation is not a simple function of temperature and pressure as is shown by the data available for methane (Figure 8.3). There is no similar data for the gas mixtures used in the experimental tests and therefore the consequences of using a constant value term cannot be determined. Also, since the specific heats for the gas mixtures used were calculated using the method of mixtures (requiring values for the specific heat of each component), this may incorporate further inaccuracies.

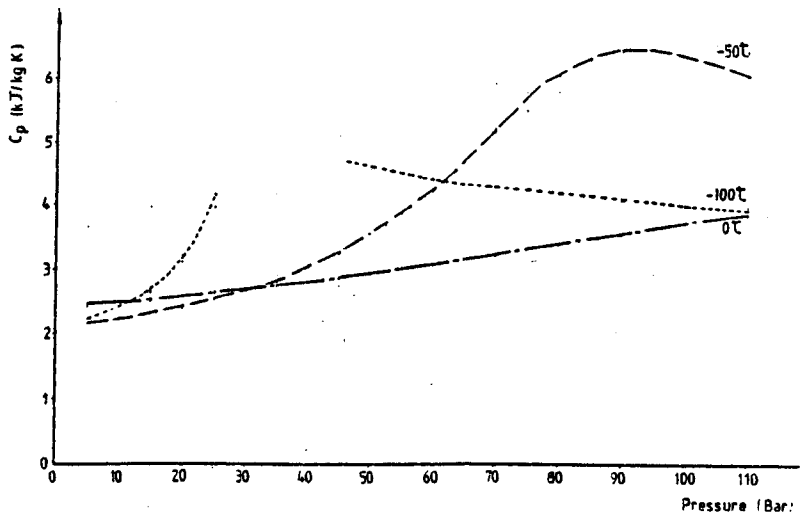


Figure 8.3. Variation of the Specific Heat of Methane

Values for the critical temperature and pressure of the gas mixture were required by the program in order that the compressibility factor could be calculated. Again, because of the complexity of some of the gas mixtures being used, these quantities had to be estimated. Any error involved in the estimation of the critical temperature (for example, due to the necessary exclusion of the nitrogen and oxygen components of natural gas in the calculations) would itself create an error of the same magnitude in the estimated value for the critical pressure. These would then both influence the calculated value for the compressibility factor.

The friction factor and, to a lesser extent, the Stanton Number may also be sources of error in the theoretical model since they were both estimated using steady flow formulae from various assumptions of the pipe wall material and condition. They were both assumed to be constant and uniform along the length of the pipe whereas in reality the values would be expected to increase in the vicinity of the break due to condensation occurring. The use of non-uniform values for the friction factor may slightly alter the shape of the pressure x time and pressure x wavespeed curves but the British Gas results (comparing different values for f) showed that this effect would be minimal.

One further problem that was inherent in some of the results from both British Gas and Foothills Pipelines (Yukon) Ltd. was that the positions of the various pressure transducers had to be

approximated to the nearest grid point in the theoretical model. This would cause slight errors in the positioning of the pressure x time curves. However, closer examination of this problem revealed that such errors were negligible.

Finally, the errors inherent in the numerical modelling of the system must be considered. In any situation where a continuous real life problem is replaced by a discrete model, a discretization error will arise in the solution. This can cause smearing when fixed grid methods are used. If the discrete equations are then solved iteratively rather than exactly, a further error is introduced called the round-off error. By reducing the grid size, the discretization error will be reduced since the discrete model would closer approximate the continuous problem. However, the round-off error would be increased since more iterative solutions would be required. Therefore, decreasing the grid size would not necessarily increase the overall accuracy.

In this project, where the grid size is varied according to its distance from the break, the discretization and round-off errors will vary along the length of the pipe. At points where the grid size changes there may also be local errors introduced.

Additional numerical problems in this project are due to the computer rounding error. Because of the complexity of the equations, in some instances it is necessary to add numbers differing

by a factor of 10^{12}. Even when working in double precision, the storage capabilities of the computer will obviously restrict the accuracy with which such calculations can be performed.

CHAPTER 9

CONCLUSIONS

The theoretical model that has been developed successfully simulated the rapid expansion of gas following a break in a high pressure gas pipeline. The reduced grid size in the vicinity of the break enabled close monitoring of the initial expansion wave in the shock tube and full size test runs, and the program's ability to simulate flow in both directions in the pipe was appreciated in the full size test runs where the break was not positioned at the end of the pipe. The method of representing the heat transfer and frictional losses in the pipe by using constant value Stanton Number and friction factor also appeared satisfactory.

The comparison of the theoretical results with available experimental data did, however, highlight the areas for concern. Firstly, it was found that despite calculating realistic isentropic and isothermal wavespeeds, the model over-estimated the actual wavespeed. This had the effect of displacing the pressure x time curves to the left of the experimental traces.

Secondly, there were found to be slight problems with the stability of the solution. For certain grid sizes and initial conditions the solution would become unstable at random points along the pipe. Although this type of instability could be controlled to an extent by varying the grid size and break boundary conditions, the problem may be totally alleviated by using an alternative numerical method for solving the theoretical equations.

If these teething problems with the program could be overcome, it is believed that excellent agreement with the experimental data would be achieved.

CHAPTER 10

FURTHER WORK

This project offers considerable scope for further work in a number of different areas, some of which are detailed below.

10.1. Investigation of the Wavespeed Error

The elimination of the error in the calculated wavespeed is essential if this theoretical model is to be developed further. A very detailed examination of the equations used and their numerical solution should reveal the source of the error. Initial investigations into this problem included an examination of the numerical relationship between the theoretical, isentropic, isothermal, and actual wavespeeds. With the Foothills Pipelines (Yukon) Ltd. data, it was found that by halving the difference between the theoretical and the isothermal wavespeeds, good agreement with experimental figures was obtained. This is shown in the table below, where the modified wavespeed is defined as:-

$$\text{Modified wavespeed} = \text{isothermal wavespeed} + \left(\text{theoretical wavespeed} - \text{isothermal wavespeed} \right) / 2$$

TEST	Isothermal wavespeed	Isentropic wavespeed	Theoretical wavespeed	Modified wavespeed	Actual wavespeed
NABTF 1	297	427	470	383.5	389
NABTF 3	266	441	468	367	354
NABTF 4	273	438	488	380.5	379
NABTF 5	286	405	469	377.5	375

(all wavespeeds are in m/s)

Although these figures are encouraging, further work is obviously necessary in order to pinpoint the error source.

10.2. Further Testing of the Present Model

This project has highlighted the need for further experimental data in order to validate conclusively the theoretical model. To date, there has been no accurate record made of the temperatures reached along the length of a pipe during and after a linebreak. Experimental tests are also required which will record the effects of a linebreak when the initial flow velocity in the pipe is non-zero.

10.3. Improvement of the Stability of the Solution

In order to improve the stability of the solution, it would be beneficial to experiment with other numerical methods for solving the ordinary differential equations produced by the Method of Characteristics. Improved computer run times may also be obtained with some methods, which would be advantageous if the program is to be extended to enable the modelling of branched systems.

10.4. Further Refinement of the Model

An extension to experimenting with alternative numerical methods for use with the Method of Characteristics, would be to examine the effectiveness of other methods of solution. For example, the use of flux difference splitting is now becoming more feasible due to new technology rapidly advancing the capacity of computers.

It would also be interesting to examine alternative models for the boundary conditions (both for the upstream and downstream pipe ends and for the break boundary). Ideally the need for operator adjustment in the setting of the break boundary condition could be overcome so that this possible error source would be eliminated. Similarly, to reduce any errors incurred in the calculation of the gas constants, C_p, R, T_c and P_c, alternative means of estimating these values ought to be examined.

One further possible refinement to the program would be to enable the computer to pre-determine the optimum time step length for each grid point. This could greatly reduce the numerical truncation errors incurred in the present model but would necessitate two-dimensional interpolation.

APPENDIX I. DERIVATION OF THE BASIC PARTIAL DIFFERENTIAL EQUATIONS, WITH PRESSURE, TEMPERATURE AND VELOCITY AS THE DEPENDENT VARIABLES.

The basic equations derived from first principles are:

$$\left[\frac{\partial \rho}{\partial t} + u \frac{\partial \rho}{\partial x}\right] + \rho \frac{\partial u}{\partial x} = 0 \qquad (1)$$

$$\rho \left[\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x}\right] + \frac{\partial P}{\partial x} = - \frac{W}{A} - \rho g \sin\theta \qquad (2)$$

$$\rho \left[\frac{\partial h}{\partial t} + u \frac{\partial h}{\partial x}\right] - \left[\frac{\partial P}{\partial t} + u \frac{\partial P}{\partial x}\right] = \frac{Wu + \Omega}{A} \qquad (3)$$

To obtain ρ in terms of P, z and T

From the equation of state

$$\rho = \frac{P}{RTz}$$

Therefore

$$\ell n\, \rho = \ell nP - \ell nR - \ell nT - \ell nz$$

Differentiating with respect to time

$$\frac{1}{\rho}\frac{d\rho}{dt} = \frac{1}{P}\frac{dP}{dt} - \frac{1}{R}\frac{dR}{dt} - \frac{1}{T}\frac{dT}{dt} - \frac{1}{z}\frac{dz}{dt}$$

But $z = z(T,P)$, therefore

$$dz = \left[\frac{\partial z}{\partial P}\right]_T dP + \left[\frac{\partial z}{\partial T}\right]_P dT$$

$$\frac{dz}{dt} = \left[\frac{\partial z}{\partial P}\right]_T \frac{dP}{dt} + \left[\frac{\partial z}{\partial T}\right]_P \frac{dT}{dt}$$

$$\therefore \frac{1}{\rho}\frac{d\rho}{dt} = \left\{\frac{1}{P} - \frac{1}{z}\left[\frac{\partial z}{\partial P}\right]_T\right\}\frac{dP}{dt} - \left\{\frac{1}{T} + \frac{1}{z}\left[\frac{\partial z}{\partial T}\right]_P\right\}\frac{dT}{dt}$$

Substituting this into equation (1):-

$$\left\{\frac{1}{P} - \frac{1}{z}\left[\frac{\partial z}{\partial P}\right]_T\right\}\frac{dP}{dt} - \left\{\frac{1}{T} + \frac{1}{z}\left[\frac{\partial z}{dT}\right]_P\right\}\frac{dT}{dt} + \frac{\partial u}{\partial x} = 0 \qquad (4)$$

To obtain h in terms of P, z and T

From Zemansky [1968]:

$$\frac{dh}{dt} = C_p\frac{dT}{dt} + \left\{\frac{T}{\rho}\left[\frac{\partial \rho}{\partial T}\right]_P + 1\right\}\frac{1}{\rho}\frac{dP}{dt}$$

Substituting this into equation (3):-

$$\rho C_p\frac{dT}{dt} + \left\{\frac{T}{\rho}\left[\frac{\partial \rho}{\partial T}\right]_P + 1 - 1\right\}\frac{dP}{dt} = \frac{\Omega + Wu}{A} \qquad (5)$$

Solving equations (4) and (5) simultaneously

$$\left.\begin{aligned}\left[\frac{1}{P} - \frac{1}{z}\left[\frac{\partial z}{\partial P}\right]_T\right]\frac{dP}{dt} - \left[\frac{1}{T} + \frac{1}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]\frac{dT}{dt} &= -\frac{\partial u}{\partial x}\\ \left[\frac{T}{\rho}\left[\frac{\partial \rho}{\partial T}\right]_P\right]\frac{dP}{dt} + \left[\rho\, C_p\right]\frac{dT}{dt} &= \frac{\Omega + Wu}{A}\end{aligned}\right\}$$

Solving for $\frac{dP}{dt}$:-

$$\rho C_p\left[\frac{1}{P} - \frac{1}{z}\left[\frac{\partial z}{\partial P}\right]_T\right]\frac{dP}{dt} + \left[\frac{1}{T} + \frac{1}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]\frac{T}{\rho}\left[\frac{\partial \rho}{\partial T}\right]_P\frac{dP}{dt}$$

$$= -\rho C_p\frac{\partial u}{\partial x} + \left[\frac{1}{T} + \frac{1}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]\frac{\Omega + Wu}{A} \qquad (6)$$

But from the equation of state:

$$\ell n\rho = \ell nP - \ell nR - \ell nT - \ell nz$$

Differentiating this with respect to temperature T, keeping pressure P constant:

$$\frac{1}{\rho}\left[\frac{\partial \rho}{\partial T}\right] = -\frac{1}{T}\left[\frac{\partial T}{\partial T}\right]_P - \frac{1}{z}\left[\frac{\partial z}{\partial T}\right]_P$$

$$= -\frac{1}{T}\left\{1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_P\right\}$$

Substituting this into equation (6):-

$$\left\{\rho C_p\left[\frac{1}{P} - \frac{1}{z}\left[\frac{\partial z}{\partial T}\right]_P\right] - \left[\frac{1}{T} + \frac{1}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]^2\right\}\frac{dP}{dt} + \rho C_p \frac{\partial u}{\partial x} = \left[\frac{1}{T} + \frac{1}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]\frac{\Omega + Wu}{A}$$

Dividing through by C_p:-

$$\frac{\rho}{P}\left\{\left[1 - \frac{P}{z}\left[\frac{\partial z}{\partial P}\right]_T\right] - \frac{PT}{\rho C_p T^2}\left[1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]^2\right\}\frac{dP}{dt} + \rho \frac{\partial u}{\partial x}$$

$$= \frac{1}{C_p T}\left[1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]\frac{\Omega + Wu}{A} \qquad (7)$$

Solving equations (4) and (5) for $\frac{dT}{dt}$:-

$$\rho C_p\left[\frac{1}{P} - \frac{1}{z}\left[\frac{\partial z}{\partial P}\right]_T\right]\frac{dT}{dt} + \left[\frac{1}{T} + \frac{1}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]\frac{T}{\rho}\left[\frac{\partial \rho}{\partial T}\right]_P \frac{dT}{dt}$$

$$= \left[\frac{1}{P} - \frac{1}{z}\left[\frac{\partial z}{\partial P}\right]_T\right]\frac{\Omega + Wu}{A} + \frac{T}{\rho}\left[\frac{\partial \rho}{\partial T}\right]_P \frac{\partial u}{\partial x}$$

Dividing through by C_p and substituting for $\frac{1}{\rho}\left[\frac{\partial \rho}{\partial T}\right]_P$ as before:-

$$\frac{\rho}{P}\left\{\left[1 - \frac{P}{z}\left[\frac{\partial z}{\partial P}\right]_T\right] - \frac{PT}{\rho C_p T^2}\left[1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]^2\right\}\frac{dT}{dt}$$

$$= \frac{1}{\rho C_p}\left[1 - \frac{P}{z}\left[\frac{\partial z}{\partial P}\right]_T\right]\frac{\Omega + Wu}{A} - \frac{1}{C_p}\left[1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]\frac{\partial u}{\partial x} \qquad (8)$$

Assume entropy 's' is a function of pressure and density, $s = s(P,\rho)$, then:-

$$ds = \left[\frac{\partial s}{\partial P}\right]_\rho dP + \left[\frac{\partial s}{\partial \rho}\right]_P d\rho$$

If the entropy is constant then:-

$$0 = \left[\frac{\partial s}{\partial P}\right]_\rho \cdot \left[\frac{\partial P}{\partial \rho}\right]_s + \left[\frac{\partial s}{\partial \rho}\right]_P$$

$$\therefore \left[\frac{\partial P}{\partial \rho}\right]_s = -\left[\frac{\partial s}{\partial \rho}\right]_P \Big/ \left[\frac{\partial s}{\partial P}\right]_\rho$$

$$= -\left[\frac{\partial s}{\partial T}\right]_P \left[\frac{\partial T}{\partial \rho}\right]_P \Big/ \left[\frac{\partial s}{\partial T}\right]_\rho \left[\frac{\partial T}{\partial P}\right]_\rho$$

Assuming temperature 'T' is a function of pressure and density, $T = T(P,\rho)$, then:-

$$dT = \left[\frac{\partial T}{\partial P}\right]_\rho dP + \left[\frac{\partial T}{\partial \rho}\right]_P d\rho$$

$$\Rightarrow \left[\frac{\partial T}{\partial \rho}\right]_P \Big/ \left[\frac{\partial T}{\partial P}\right]_\rho = -\frac{1}{\left[\frac{\partial \rho}{\partial P}\right]_T}$$

Therefore:-

$$\left[\frac{\partial P}{\partial \rho}\right]_s = \left[\left[\frac{\partial s}{\partial T}\right]_P \Big/ \left[\frac{\partial s}{\partial T}\right]_\rho\right] \cdot \frac{1}{\left[\frac{\partial \rho}{\partial P}\right]_T} \qquad (9)$$

But from Zemansky [1968] (page 288):

$$\left[\frac{\partial s}{\partial T}\right]_P = \frac{C_p}{T} \quad \text{and} \quad \left[\frac{\partial s}{\partial T}\right]_\rho = \frac{C_v}{T} = \frac{C_p}{T} - \left[\frac{\partial v}{\partial T}\right]_P \left[\frac{\partial P}{\partial T}\right]_v$$

Also,

$$\left[\frac{\partial P}{\partial T}\right]_v = -\left[\frac{\partial v}{\partial T}\right]_P \Big/ \left[\frac{\partial v}{\partial P}\right]_T$$

Therefore:

$$\left[\frac{\partial s}{\partial T}\right]_\rho = \frac{C_p}{T} + \left[\frac{\partial v}{\partial T}\right]_P^2 \Big/ \left[\frac{\partial v}{\partial P}\right]_T$$

$$= \frac{C_p}{T} + \frac{1}{\rho^4}\left[\frac{\partial \rho}{\partial T}\right]_P^2 \Big/ \frac{-1}{\rho^2}\left[\frac{\partial \rho}{\partial P}\right]_T$$

$$= \frac{C_p}{T} - \frac{1}{\rho^2}\left[\left[\frac{\partial\rho}{\partial T}\right]_P^2 / \left[\frac{\partial\rho}{\partial P}\right]_T\right]$$

$$\therefore \left[\frac{\partial s}{\partial T}\right]_\rho \cdot \left[\frac{\partial\rho}{\partial P}\right]_T = \frac{C_p}{T}\left[\frac{\partial\rho}{\partial P}\right]_T - \frac{1}{\rho^2}\left[\frac{\partial\rho}{\partial T}\right]_P^2 \qquad (10)$$

But it has already been proved that:-

$$\left[\frac{\partial\rho}{\partial T}\right]_P = -\frac{\rho}{T}\left[1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]$$

And from the equation of state:

$$\left[\frac{\partial\rho}{\partial P}\right]_T = \frac{\rho}{P}\left[1 - \frac{P}{z}\left[\frac{\partial z}{\partial P}\right]_T\right]$$

Substituting these identities into equation (9):-

$$\left[\frac{\partial P}{\partial\rho}\right]_s = \frac{C_p}{T}\left[\frac{C_p}{T}\left\{\frac{\rho}{P}\left[1 - \frac{P}{z}\left[\frac{\partial z}{\partial P}\right]_T\right] - \frac{1}{C_pT}\left[1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]^2\right\}\right]^{-1}$$

$$= \frac{P}{\rho}\left[1\,\frac{\rho}{P}\left\{\left[1 - \frac{P}{z}\left[\frac{\partial z}{\partial P}\right]_T - \frac{P}{\rho C_pT}\left[1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]^2\right\}\right]^{-1}$$

and $\left[\frac{\partial P}{\partial\rho}\right]_s^{\frac{1}{2}}$ can be defined as the ISENTROPIC WAVE SPEED 'a_s'.

Substituting this into equations (7) and (8):-

$$\frac{1}{a_s^2}\frac{dP}{dt} + \rho\frac{\partial u}{\partial x} = \frac{1}{C_pT}\left[1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]\frac{\Omega + Wu}{A}$$

$$\frac{1}{a_s^2}\frac{dT}{dt} + \frac{1}{C_p}\left[1 + \frac{T}{z}\left(\frac{\partial z}{\partial T}\right)_P\right]\frac{\partial u}{\partial x} = \frac{1}{C_pP}\left[1 - \frac{P}{z}\left(\frac{\partial z}{\partial P}\right)_T\right]\frac{\Omega + Wu}{A}$$

or alternatively, including equation (2):-

$$\frac{\partial P}{\partial t} + u\frac{\partial P}{\partial x} + \rho a_s^2\frac{\partial u}{\partial x} = \frac{a_s^2}{C_pT}\left[1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_P\right]\frac{\Omega + Wu}{A} \qquad (11)$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \frac{1}{\rho}\frac{\partial P}{\partial x} = -\frac{W}{\rho A} - g\sin\theta \qquad (12)$$

$$\frac{\partial T}{\partial t} + u\frac{\partial T}{\partial x} + \frac{a_s^2}{C_p}\left[1 + \frac{T}{z}\left[\frac{\partial z}{\partial T}\right]_p\right]\frac{\partial u}{\partial x} = \frac{a_s^2}{C_p P}\left[1 - \frac{P}{z}\left[\frac{\partial z}{\partial P}\right]_T\right]\frac{\Omega + \dot{W}u}{A} \qquad (13)$$

APPENDIX II. FRICTION FACTOR RELATIONSHIPS

Turbulent flow (such as usually occurs in gas transmission pipelines) may be categorised into two regimes:-

1. Fully developed turbulence - this is described by the Rough Pipe Law which assumes that the friction factor is solely dependent on the pipe roughness $\overset{(k)}{\wedge}$ and size. $\overset{(d)}{\wedge}$ The rough pipe law is of the form:

$$\frac{1}{\sqrt{f}} = A_1 \log \left[\frac{d}{k}\right] + B_1$$

2. Partially developed turbulence - this is described by either the Smooth Pipe Law or in Blasius form. Here it is assumed that the coefficient of friction is dependent on fluid properties and conduit size alone. The Smooth pipe law is of the form:

$$\frac{1}{\sqrt{f}} = A_2 \log(Re / \frac{1}{\sqrt{f}}) + B_2$$

and the Blasius form of the friction relationship is:

$$f = A_3 \, Re^{B_3}$$

A and B in each of these expressions are constants.

There is also a transition zone between the partially and fully developed turbulence which can be described by a combination of the above two laws.

Listed below are some of the relationships defining the friction factors that various research teams have used for the analysis of transient gas flows in pipes. Some of the relationships have been adapted so that they all apply to the definition of frictional force per unit length (W):-

$$W = \frac{A}{d} \rho f \frac{u|u|}{2}$$

1. Fully Developed Turbulence

In the mid 1950's Smith et al [1956] also developed a version of the rough pipe law:-

$$\frac{1}{\sqrt{f}} = 2 \log \left[\frac{3.7d}{k}\right] + 2.273$$

This version is almost identical to that developed empirically by Nikuradse [1933]. Nikuradse's formula was used by Weimann [1978] to dynamically model gas distribution networks because Schlichting [1965] maintained that it applied approximately to transient flow processes with slow vibrations of moderate amplitude. Taylor [1978] also used a friction factor defined by the rough pipe law. He assumed that because of the high velocities normally attained the flow would be fully turbulent.

2. Partially Developed Turbulence

Blasius [1911] proposed an equation for partially turbulent flow valid for Reynolds numbers below 10^5:-

$$f = 0.3160\ Re^{-0.25}$$

Since then, numerous modifications have been made to the equation by various researchers. For example, Chaudhry [1979] used a friction factor for flows in which

$Re > 2 \times 10^3$, defined by:-

$$f = 0.046\ Re^{-0.2}$$

Another equation that can be written in Blasius form is the Panhandle 'A' equation:-

$$\frac{1}{\sqrt{f}} = 3.39\ Re^{0.073}\ E$$

$$\Rightarrow f = 0.087 Re^{-0.146}\ \frac{1}{E^2}$$

where E is the efficiency of the system and is an adjustable parameter which allows for the effects of the minor losses and variations in pipe roughness. The Panhandle 'A' relationship is a popular equation for gas transmission calculations where partially developed turbulent flow is occurring.

Smith et al.[1956] developed a smooth pipe law which, for the Darcy friction factor, was defined as:-

$$\frac{1}{\sqrt{f}} = 2 \log \left[Re / \frac{2}{\sqrt{f}} \right] - 0.3$$

This version, when multiplied by a drag factor, F, to account for the effect of bends and fittings, was favoured by Uhl et al [1965] to represent partially developed turbulent flow:-

$$\frac{1}{\sqrt{f}} = -2F \log \left[\frac{4}{Re\sqrt{f}}\right]$$

3. Transition Zone

Colebrook [1938-9] proposed an equation for the transition zone between partially and fully developed flows:-

$$\frac{1}{\sqrt{f}} = 2 \log \frac{1}{k/3.7d + 2.5\ (1/\sqrt{f})/Re}$$

This was the first reasonably successful attempt to define a universal friction factor relationship for turbulent flow. It is an implicit semi-empirical formula which is represented graphically by the Moody Diagram and is the basis of the Colebrook-White or Prandtl-Colebrook equations:

$$\frac{1}{\sqrt{f}} = -2 \log \left[\frac{k}{3.7d} + \frac{2.53}{Re\sqrt{f}}\right]$$

Oliemans [1976] used a modified version of Colebrook's expression to model friction in two-phase flow:-

$$\frac{1}{\sqrt{f}} = -2 \log \left[\frac{2k}{d_{eff}} + \frac{18.7}{Re'\sqrt{f}}\right] + 1.74$$

where Re' is a two-phase Reynolds number

and d_{eff} is the effective diameter for two-phase mixture.

Over the years, numerous explicit approximations to Colebrook's equation have been developed, the first of which was by Moody [1947]:-

$$f = 0.0055 \left\{1 + \left[20000 \left[\frac{k}{d}\right] + \frac{10^6}{Re}\right]^{1/3}\right\}$$

This equation was relatively inaccurate showing an average error of 4.3% for the test cases of Zigrang and Sylvester [1982].

Swamee and Jain [1976] developed an explicit equation by curve fitting the Colebrook-White equation:-

$$\frac{1}{\sqrt{f}} = -2 \log \left[\frac{k}{3.7d} + \frac{5.74}{Re^{0.9}}\right]$$

This equation was found to have an accuracy to within 1% for steady flows where $5 \times 10^3 < Re < 10^8$ and $10^{-6} < k/d < 10^{-2}$.

Further explicit equations were obtained by substituting values for f into the right-hand side of Colebrook's equation. Zigrang and Sylvester [1982] used $f = 0.04$ and Shacham [1980] used $f = 0.03$.

Zigrang and Sylvester

$$\frac{1}{\sqrt{f}} = -2\log \left\{\frac{k/d}{3.7} + \frac{13}{Re}\right\}$$

Shacham

$$\frac{1}{\sqrt{f}} = -2\log \left\{\frac{k/d}{3.7} + \frac{14.5}{Re}\right\}$$

Haaland [1983] re-examined the basis for Colebrook's equation and developed the explicit relation:

$$\frac{1}{\sqrt{f}} = -1.8\log\left\{\frac{6.9}{Re} + \left(\frac{k/d}{3.7}\right)^{1.11}\right\}$$

Haaland then generalised this equation to:-

$$\frac{1}{\sqrt{f}} = \frac{-1.8}{n}\log\left\{\left(\frac{6.9}{Re}\right)^{n} + \left(\frac{k/d}{3.7}\right)^{1.11n}\right\}$$

and suggested that n = 3 yielded friction factors in agreement with those recommended for use in gas transmission lines.

The accuracy of these and more complicated explicit friction factor equations has been examined by Zigrang and Sylvester [1985].

APPENDIX III. IMPLEMENTATION OF TAYLOR'S THEOREM FOR VARIOUS GRID POINTS

Taylor's expansion around a point x is given by:-

$$u(x + h) = u(x) + hu'(x) + \frac{h^2}{2}u''(x) + \frac{h^3}{6}u'''(x) + \ldots\ldots$$

$$u(x - h) = u(x) - hu'(x) + \frac{h^2}{2}u''(x) - \frac{h^3}{6}u'''(x) + \ldots\ldots$$

For a standard internal point with equidistant adjacent points, using the notation shown in Figure 4.2(i), these equations may be written:-

$$u(i + 1) = u(i) + \Delta x.u'(i) + \frac{(\Delta x)^2}{2}u''(i) + \frac{(\Delta x)^3}{6}u'''(i) + \ldots.(1)$$

$$u(i - 1) = u(i) - \Delta x.u'(i) + \frac{(\Delta x)^2}{2}u''(i) - \frac{(\Delta x)^3}{6}u'''(i) + \ldots.(2)$$

Adding equations (1) and (2) produces:

$u(i + 1) + u(i - 1) = 2u(i) + 2\frac{(\Delta x)^2}{2}.u''(i)$ neglecting higher order terms

$$\Rightarrow \quad u''(i) = \frac{1}{\Delta x^2}\left\{u(i + 1) + (u - 1) - 2u(i)\right\} \qquad (3)$$

Subtracting equation (2) from equation (1) produces:

$u(i + 1) - u(i - 1) = 2\Delta x.u'(i)$ neglecting higher order terms

$$\Rightarrow \quad u'(i) = \frac{1}{2\Delta x}\left\{u(i + 1) - u(i - 1)\right\} \qquad (4)$$

With reference to point Q in Figure 4.2(i), by substituting equations (3) and (4) back into Taylor's expansion, the following expression for the property u at point Q may be derived:

$$u_Q = u(i - posQ)$$

$$= u(i) - \frac{posQ}{2\Delta x}\left\{u(i+1) - u(i-1)\right\} + \frac{posQ^2}{2\Delta x^2}\left\{u(i+1) + u(i-1) - 2u(i)\right\} \quad (5)$$

Similarly for point R:

$$u_R = u(i) - \frac{posR}{2\Delta x}\left\{u(i+1) - u(i-1)\right\} + \frac{posR^2}{2\Delta x^2}\left\{u(i+1) + u(i-1) - 2u(i)\right\} \quad (6)$$

And for point S:

$$u_S = u(i) + \frac{posS}{2\Delta x}\left\{u(i+1) - u(i-1)\right\} + \frac{posS^2}{2\Delta x^2}\left\{u(i+1) + u(i-1) - 2u(i)\right\} \quad (7)$$

Equations (5), (6) and (7) are used to define each of the variables P, u, T, z, $\partial z/\partial T$, ρ, a_s, Ω, and W at the bases of the characteristics.

For an internal point between two different grid sizes, two separate Taylor's expansions are necessary at the adjacent grid points. For such a point upstream of the break, as detailed in Figure 4.2(ii), the Taylor's expansions about point (i + 1) and point (i - 1) yield the following formulae:

$$u''(i + 1) = \frac{1}{(\Delta x)^2}\left\{u(i + 2) + u(i) - 2u(i + 1)\right\}$$

$$u'(i + 1) = \frac{1}{2\Delta x}\left\{u(i + 2) - u(i)\right\}$$

$$u''(i-1) = \frac{1}{4\Delta x^2}\left\{u(i) + u(i-2) - 2u(i-1)\right\}$$

$$u'(i-1) = \frac{1}{4\Delta x}\left\{u(i) - u(i-2)\right\}$$

Therefore,

$$u_Q = u((i-1) + (2\Delta x - posQ))$$

$$= u(i-1) + \frac{(2\Delta x - posQ)}{4\Delta x}\left\{u(i) - u(i-2)\right\}$$

$$+ \frac{(2\Delta x - posQ)^2}{8\Delta x^2}\left\{u(i) + u(i-2) - 2u(i-1)\right\} \quad (8)$$

For point R:

$$u_R = u((i-1) + (2\Delta x - posR))$$

$$= u(i-1) + \frac{(2\Delta x - posR)}{4\Delta x}\left\{u(i) - u(i-2)\right\}$$

$$+ \frac{(2\Delta x - posR)^2}{8\Delta x^2}\left\{u(i) + u(i-2) - 2u(i-1)\right\} \quad (9)$$

and for point S:

$$u_S = u((i+1) - (\Delta x - posS))$$

$$= u(i+1) - \frac{(\Delta x - posS)}{2\Delta x}\left\{u(i+2) - u(i)\right\}$$

$$+ \frac{(\Delta x - posS)^2}{2\Delta x^2}\left\{u(i+2) + u(i) - 2u(i+1)\right\} \quad (10)$$

By the same logic, for this type of point situated downstream of the break as shown in Figure 4.2(iv), the following equations may be derived:

$$u_Q = u((i-1) + (\Delta x - posQ))$$

$$= u(i-1) + \frac{(\Delta x - posQ)}{2\Delta x}\left\{u(i) - u(i-2)\right\}$$

$$+ \frac{(\Delta x - posQ)^2}{2x^2}\left\{u(i) + u(i-2) - 2u(i-1)\right\} \qquad (11)$$

$$u_R = u((i-1) + (\Delta x - posR))$$

$$= u(i-1) + \frac{(\Delta x - posR)}{2\Delta x}\left\{u(i) - u(i-2)\right\}$$

$$+ \frac{(\Delta x - posR)^2}{2\Delta x^2}\left\{u(i) + u(i-2) - 2u(i-1)\right\} \qquad (12)$$

$$u_S = u((i+1) - (2\Delta x - posS))$$

$$= u(i+1) - \frac{(2\Delta x - posS)}{4\Delta x}\left\{u(i+2) - u(i)\right\}$$

$$+ \frac{(2\Delta x - posS)^2}{8\Delta x^2}\left\{u(i) + u(i+2) - 2u(i+1)\right\} \qquad (13)$$

Equations (11), (12) and (13) are valid when the flow is in the positive x direction. However, if flow reversal occurs, equation (11) has to be replaced by:

$$u_Q = u((i+1) - (2\Delta x - posQ))$$

$$= u(i+1) - \frac{(2\Delta x - posQ)}{4\Delta x}\left\{u(i+2) - u(i)\right\}$$

$$+ \frac{(2\Delta x - posQ)^2}{8\Delta x^2}\left\{u(i+2) + u(i) - 2u(i+1)\right\} \qquad (14)$$

In the situation of a point linking two different grid size regions (as illustrated in Figure 4.4) two separate Taylor expansions are required at two different time levels. Using the notation of Figure 4.4, the following expressions may be derived:

For a point upstream of the break:-

$$u''(i+1) = \frac{1}{(\Delta x)^2}\left\{u(i+2) + u_Y - 2u(i+1)\right\}$$

$$u'(i+1) = \frac{1}{2\Delta x}\left\{u(i+2) - u_Y\right\}$$

$$u''(i-1) = \frac{1}{4\Delta x^2}\left\{u_X + u(i-2) - 2u(i-1)\right\}$$

$$u'(i-1) = \frac{1}{4\Delta x}\left\{u_X - u(i-2)\right\}$$

Therefore:

$$u_Q = u(i-1) + \frac{(2\Delta x - posQ)}{4\Delta x}\left\{u_X - u(i-2)\right\} + \frac{(2\Delta x - posQ)^2}{8\Delta x^2}\left\{u_X + u(i-2) - 2u(i-1)\right\} \quad (15)$$

$$u_R = u(i-1) + \frac{(2\Delta x - posR)}{4\Delta x}\left\{u_X - u(i-2)\right\} + \frac{(2\Delta x - posR)^2}{8\Delta x^2}\left\{u_X + u(i-2) - 2u(i-1)\right\} \quad (16)$$

$$u_S = u(i+1) - \frac{(\Delta x - posS)}{2\Delta x}\left\{u(i+2) - u_Y\right\} + \frac{(\Delta x - posS)^2}{2\Delta x^2}\left\{u(i+2) + u_Y - 2u(i+1)\right\} \quad (17)$$

Similarly, for this situation occurring downstream of the break, Figure 4.2(v):-

$$u_Q' = u(i-1) + \frac{(\Delta x - posQ)}{2\Delta x}\left\{u_X - u(i-2)\right\} + \frac{(\Delta x - posQ)^2}{2\Delta x^2}\left\{u_X + u(i-2) - 2u(i-1)\right\} \quad (18)$$

$$u_R = u(i-1) + \frac{(\Delta x - posR)}{2\Delta x}\left\{u_X - u(i-2)\right\} + \frac{(\Delta x - posR)^2}{2\Delta x^2}\left\{u_X + u(i-2) - 2u(i-1)\right\} \quad (19)$$

$$u_S = u(i+1) - \frac{(2\Delta x - posS)}{4\Delta x}\left\{u(i+2) - u_Y\right\}$$

$$+ \frac{(2\Delta x - posS)^2}{8\Delta x^2}\left\{u(i+2) + u_Y - 2u(i+1)\right\} \qquad (20)$$

and if flow reversal occurs, equation (18) is replaced by:

$$u_Q = u(i+1) - \frac{(2\Delta x - posQ)}{4\Delta x}\left\{u(i+2) - u_Y\right\}$$

$$+ \frac{(2\Delta x - posQ)^2}{8\Delta x^2}\left\{u(i+2) + u_Y - 2u(i+1)\right\} \qquad (21)$$

APPENDIX IV. DERIVATION OF THE PARTICLE VELOCITY OF A RAREFACTION WAVE

After making certain investigations into the properties of a sound wave transmitted through a small horizontal tube of uniform bore, Earnshaw [1860] deduced that for a system shown in Figure A.1:

$$\frac{dy}{dt} = F\left[\frac{dy}{dx}\right] \tag{1}$$

where F is a function of a form to be determined.

Differentiating equation (1) with respect to time produces:-

$$\frac{d^2y}{dt^2} = \left\{F'\left[\frac{dy}{dx}\right]\right\}^2 \frac{d^2y}{dx^2} \tag{2}$$

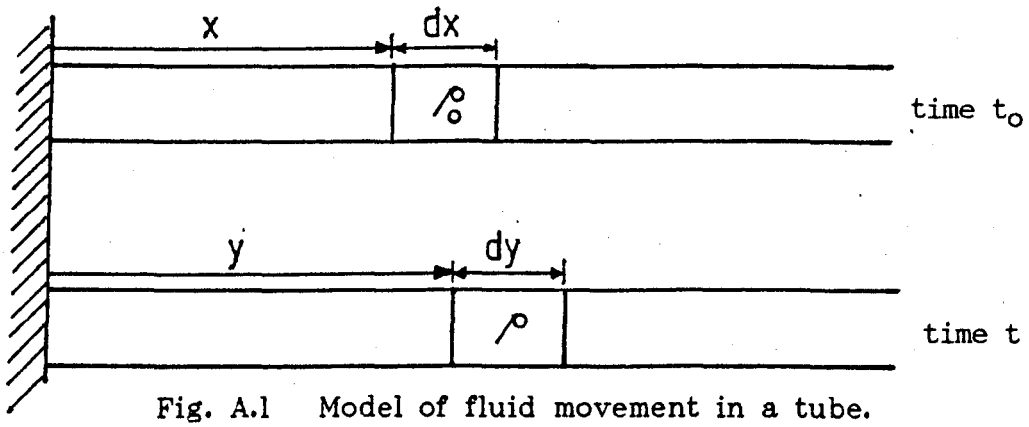


Fig. A.1 Model of fluid movement in a tube.

From the conservation of mass, using the notation shown in Figure A.1.

$$\rho_o \, dx = \rho \, dy$$

$$\Rightarrow \quad \rho = \rho_o \left[\frac{dy}{dx}\right]^{-1} \tag{3}$$

Similarly, from the conservation of linear momentum:

$$PA - \left[P + \frac{\partial P}{\partial x}\,dx\right]A = \rho_o\, A\, dx\, \frac{d^2y}{dt^2}$$

$$\Rightarrow \quad -\frac{\partial P}{\partial x} = \rho_o \frac{d^2y}{dt^2} \tag{4}$$

If the flow in the tube can be assumed to be isentropic, then:

$$P = \frac{P_o}{\rho_o^{\gamma}} \cdot \rho^{\gamma}$$

$$\therefore \quad \frac{\partial P}{\partial x} = \frac{P_o}{\rho_o^{\gamma}} \cdot \gamma \rho^{\gamma-1} \cdot \frac{\partial \rho}{\partial x} \tag{5}$$

But by differentiating equation (3) with respect to x, the following expression for $\partial\rho/\partial x$ is obtained:

$$\frac{\partial \rho}{\partial x} = -\rho_o \left[\frac{dy}{dx}\right]^{-2} \frac{d^2y}{dx^2} \tag{6}$$

Substituting equations (6) and (3) into equation (5) produces:

$$\frac{\partial P}{\partial x} = \frac{P_o}{\rho_o^{\gamma}} \cdot \gamma \rho_o^{\gamma-1} \left[\frac{dy}{dx}\right]^{-\gamma+1} \cdot (-\rho_o) \cdot \left[\frac{dy}{dx}\right]^{-2} \cdot \frac{d^2y}{dx^2}$$

$$= -P_o\, \gamma \left[\frac{dy}{dx}\right]^{-\gamma-1} \cdot \frac{d^2y}{dx^2} \tag{7}$$

Equation (4) can then be re-written using equation (7):

$$\frac{P_o}{\rho_o} \cdot \gamma \left[\frac{dy}{dx}\right]^{-\gamma-1} \cdot \frac{d^2y}{dx^2} = \frac{d^2y}{dt^2} \tag{8}$$

Comparing equations (2) and (8):

$$\left\{F'\left[\frac{dy}{dx}\right]\right\}^2 = \frac{\gamma P_o}{\rho_o} \Big/ \left[\frac{dy}{dx}\right]^{\gamma+1}$$

$$\therefore \quad F'\left[\frac{dy}{dx}\right] = \pm \sqrt{\frac{P_o}{\rho_o}} \Big/ \left[\frac{dy}{dx}\right]^{\frac{\gamma+1}{2}} \tag{9}$$

Integrating equation (9):

$$F\left[\frac{dy}{dx}\right] = \pm \sqrt{\frac{\gamma P_o}{\rho_o}} \cdot \frac{2}{-\gamma+1} \left[\frac{dy}{dx}\right]^{\frac{1-\gamma}{2}} + C$$

where C is a constant of integration.

Substituting for $F\left[\frac{dy}{dx}\right]$ and $\left[\frac{dy}{dx}\right]$ using equations (1) and (3) respectively:

$$\frac{dy}{dt} = C \pm \sqrt{\frac{\gamma P_o}{\rho_o}} \cdot \frac{2}{1-\gamma} \left[\frac{\rho_o}{\rho}\right]^{\frac{1-\gamma}{2}}$$

$$= C \mp \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_o}{\rho_o}} \left(\frac{\rho_o}{\rho}\right)^{\frac{1-\gamma}{2}} \tag{10}$$

Taking boundary conditions as:

$$u\left[= \frac{dy}{dt}\right] = 0 \text{ when } \rho = \rho_o$$

then from equation (10):

$$0 = C \mp \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_o}{\rho_o}}$$

$$\Rightarrow C = \pm \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_o}{\rho_o}}$$

Substituting this back into equation (10):

$$u = \pm \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_o}{\rho_o}} \mp \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_o}{\rho_o}} \left[\frac{\rho_o}{\rho}\right]^{\frac{1-\gamma}{2}}$$

$$= \mp \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_o}{\rho_o}} \left\{\left[\frac{\rho}{\rho_o}\right]^{\frac{\gamma-1}{2}} - 1\right\}$$

but since isentropic flow has been assumed:

$$\frac{\rho}{\rho_o} = \left[\frac{P}{P_o}\right]^{\frac{1}{\gamma}}$$

and

$$a_o = \sqrt{\frac{\gamma P_o}{\rho_o}}$$

Therefore:

$$u = -\frac{2}{\gamma-1} a_o \left\{\left[\frac{P}{P_o}\right]^{\frac{\gamma-1}{2\gamma}} - 1\right\}$$

$$= \frac{2}{\gamma-1} a_o \left\{1 - \left[\frac{P}{P_o}\right]^{\frac{\gamma-1}{2\gamma}}\right\} \qquad (11)$$

This expression has been used by several investigators (for example Bakhtar [1956], Jones and Gough [1981], Bannister and Mucklow [1948]) to determine the particle velocity of a rarefaction wave.

COMPONENT		SHOCK TUBE				FULL SIZE			
		GROVES	BGC (S/T)			FOOTHILLS (ALBERTA)			
			Test 7	Test 8	Test 9	NABTF1	NABTF3	NABTF4	NABTF5
OXYGEN	O_2	0.0008	0.022	0.017	0.018	0.016	1.56	0.013	2.212
NITROGEN	N_2	1.498	0.91	0.87	0.89	1.710		1.804	
CARBON DIOXIDE	CO_2	1.073	0.62	0.62	0.61	0.076		0.049	
METHANE	CH_4	83.266	82.4	82.3	82.8	86.59	85.36	85.36	84.70
ETHANE	C_2H_6	9.608	7.89	7.85	7.83	6.80	8.22	7.68	8.21
PROPANE	C_3H_8	3.597	5.2	5.3	5.1	4.03	4.34	4.46	4.38
ISO-BUTANE	iC_4H_{10}	0.3414	0.45	0.47	0.43	0.262	0.182	0.238	0.201
N-BUTANE	nC_4H_{10}	0.4581	1.21	1.28	1.15	0.421	0.278	0.331	0.235
ISO-PENTANE	iC_5H_{12}	0.0403	0.25	0.27	0.24	0.057	0.029	0.032	0.029
N-PENTANE	nC_5H_{12}	0.0342	0.35	0.37	0.33	0.034	0.028	0.032	0.030
N-HEXANE	nC_6H_{14}	0.0046	0.64	0.63	0.56	0.008	0.013	0.011	0.008
N-HEPTANE	nC_7H_{16}	0.0003	0.003	0.003	0.002	-	-	-	-
N-OCTANE	nC_8H_{18}	0.0001	-	-	-	-	-	-	-
N-NONANE	nC_9H_{20}	-	-	-	-	-	-	-	-

TABLE A1. NATURAL GAS MOLAR COMPOSITIONS (%)

COMPONENT		SHOCK TUBE				FULL SIZE			
		GROVES	BGC (S/T)			FOOTHILLS (ALBERTA)			
			Test 7	Test 8	Test 9	NABTF1	NABTF3	NABTF4	NABTF5
OXYGEN	O_2	-	0.03	0.03	0.03	0.03	2.32	0.02	3.28
NITROGEN	N_2	2.18	1.25	1.19	1.23	2.56		2.68	
CARBON DIOXIDE	CO_2	2.45	1.34	1.33	1.32	0.18		0.11	
METHANE	CH_4	69.40	64.89	64.57	65.66	74.28	72.70	72.50	72.04
ETHANE	C_2H_6	15.01	11.65	11.54	11.64	10.93	13.12	12.23	13.09
PROPANE	C_3H_8	8.24	11.26	11.43	11.12	9.50	10.16	10.41	10.24
ISO-BUTANE	iC_4H_{10}	1.03	1.28	1.34	1.23	0.81	0.56	0.73	0.62
N-BUTANE	nC_4H_{10}	1.38	3.45	3.64	3.30	1.31	0.86	1.02	0.72
ISO-PENTANE	iC_5H_{12}	0.15	0.89	0.95	0.86	0.22	0.11	0.12	0.11
N-PENTANE	nC_5H_{12}	0.13	1.24	1.30	1.18	0.13	0.11	0.12	0.11
N-HEXANE	nC_6H_{14}	0.02	2.71	2.65	2.38	0.04	0.06	0.05	0.04
N-HEPTANE	nC_7H_{16}	-	0.01	0.01	0.01	-	-	-	-
N-OCTANE	nC_8H_{18}	-	-	-	-	-	-	-	-
N-NONANE	nC_9H_{20}	-	-	-	-	-	-	-	-

TABLE A2. NATURAL GAS MASS COMPOSITIONS (%)

Component		Mol.wt. M	Boiling Pt. Temp. T_b (°K)	Critical Temp. T_c (°K)	Critical Pressure P_c (kPa)	Critical Spec.Vol. V_c (cm^3/mol)	Critical Comp.Factor Z_c	Spec.Gas Const.R (J/kg)	Spec.Heat C_p (J/kg)	Ratio of Spec.Heats γ
ARGON	Ar	39.948	87.3	150.8	4870	74.9	0.291	208.13	520	1.668
NITROGEN	N_2	28.013	77.4	126.2	3390	89.8	0.290	296.84	1037	1.401
CARBON DIOXIDE	CO_2	44.010	316.5	304.1	7380	93.9	0.274	188.92	819	1.300
METHANE	CH_4	16.043	111.6	190.4	4600	99.2	0.288	518.36	2174	1.313
ETHANE	C_2H_6	30.070	184.6	305.4	4880	148.3	0.285	276.51	1533	1.220
PROPANE	C_3H_8	44.094	231.1	369.8	4250	203.0	0.281	188.54	1639	1.130
ISO-BUTANE	iC_4H_{10}	58.124	261.4	408.2	3650	263	0.283	143.03	1633	1.096
N-BUTANE	nC_4H_{10}	58.124	272.7	425.2	3800	255	0.274	143.03	1633	1.096
ISO-PENTANE	iC_5H_{12}	72.151	282.6	433.8	3200	303	0.269	115.24	1455	1.086
N-PENTANE	nC_5H_{12}	72.151	309.2	469.7	3370	304	0.263	115.24	1455	1.086
N-HEXANE	nC_6H_{14}	86.178	341.9	507.5	3010	370	0.264	96.48	1302	1.080

TABLE A3. PROPERTIES OF THE MAIN CONSTITUENTS OF THE GAS MIXTURES

APPENDIX V PROGRAM LISTINGS

The programs and subroutines are listed in the following order:-

Transient Analysis Program

Subroutine STEAD1

Subroutine STEAD2

Subroutine SUB1

Subroutine SUB2

Subroutine SUB3

Subroutine SUB4

Subroutine SUB5

Subroutine SUB6

Subroutine BREAK1

Subroutine BREAK2

Subroutine BREAK3

Subroutine BREAK4

Subroutine SUBUP

Subroutine DOWN1

Subroutine GETFIL

Subroutine DMINV

Graphics Program

```
*                                                                      *
*      Program to evaluate pressure surge in gas pipelines             *
*                                                                      *
*      C. Tiley                                        February 1987   *
*                                                                      *
************************************************************************
c
c       notation:-
c
c       aat   isentropic wavespeed for atmosphere
c       ao    initial isentropic wavespeed at the break point
c       ar    cross-sectional area of the pipeline
c       as    isentropic wavespeed
c       cp    specific heat at constant pressure
c       d     diameter of pipeline
c       dt    variable time step used in subroutines
c       dt1   specified time step
c       dx    variable section length used in subroutines
c       dx1   specified elemental pipe section length
c       dd    grid length near the break (= dx/64)
c       e     flow rate / area
c       f     fricton factor
c       g     acceleration due to gravity
c       ga    ratio of specific heats
c       ht    heat transfer
c       l1    length of pipe before break
c       l2    length of pipe after break
c       m1    number of i values before break
c       m2    number of i values after break
c       n1    number of elemental sections of length dx before break
c       n2    number of elemental sections of length dx after break
c       p     pressure at a point
c       pat   atmospheric pressure
c       pc    critical pressure of the gas
c       pe    equalisation pressure
c       pec   critical equalisation pressure
c       pi    pi(=3.14159)
c       r     specific gas constant
c       ro    density at a point
c       st    stanton number
c       t     temperature at a point
c       tat   atmospheric temperature
c       tc    critical temperature of the gas
c       th    angle of inclination of the pipe
c       ti    time after break
c       tm    total run time
c       tw    pipe wall temperature
c       u     flow velocity at a point
c       w     frictional force
c       z     compressibility factor
c       zp    (dz/dp) at constant temperature
c       zt    (dz/dt) at constant pressure
c
      character*1 quest
      implicit double precision (a-h,o-z)
      double precision l1,l2
      dimension p1(300),t1(300),u1(300),z1(300),zp1(300),zt1(300),
     & ro1(300),as1(300),ht1(300),w1(300),
     & p2(300),t2(300),u2(300),z2(300),zp2(300),zt2(300),
     & ro2(300),as2(300),ht2(300),w2(300),
     & pp1(300),tt1(300),uu1(300),pp2(300),tt2(300),uu2(300),
     & pps1(300),pps2(300),tts1(300),tts2(300),zpx(6),zpy(6),
     & px(6),ux(6),tx(6),zx(6),wx(6),ztx(6),rox(6),asx(6),htx(6),
```

```
     & pz(6),uz(6),tz(6),zz(6),wz(6),ztz(6),roz(6),asz(6),htz(6),
     & wave1(300),wave2(300),at1(300),at2(300)
      pi=3.141592654
      g=9.81
  100 format(v)
      ti=0.0
c
c       Read in initial gas data
c
      print*,'is the gas data on file ? (y/n) '
      read*, quest
      if (quest.eq.'n'.or.quest.eq.'n') goto 1
      print*, 'enter name of gas '
      call getfil(10)
      read(10,*)cp,r,tc,pc
      goto 2
    1 print*, 'gas data required:-'
      print*, 'specific heat at constant pressure cp (kj/kg k)'
      read*, cp
      cp=cp*1000
      print*, 'specific gas constant r (kj/kg k)'
      read*, r
      r=r*1000
      print*, 'critical temperature tc (celcius)'
      read*, tc
      tc=tc+273.16
      print*, 'critical pressure pc (kpa)'
      read*, pc
      pc=pc*1000
c
c       Read in initial pipeline data
c
    2 print*, 'is the pipeline data on file ? (y/n) '
      read*, quest
      if (quest.eq.'n'.or.quest.eq.'n') goto 3
      print*, 'pipeline '
      call getfil(11)
      read(11,*)d,th,l1,l2,dd,f,st,tw
      goto 4
    3 print*, 'pipeline data required:-'
      print*, 'diameter of pipe d (m) ?'
      read*, d
      print*, 'angle of inclined pipe (degrees) ?'
      read*, th
      th = th * pi / 180.0
      print*, 'length of pipe upstream of the break point (m) ?'
      read*, l1
      print*, 'length of pipe downstream of the break point (m) ?'
      read*, l2
      print*, 'required grid size near the break (m) ?'
      if(l2.eq.0.0) goto 111
      dx2=min0(l1,l2)/576.0
      goto 112
  111 dx2=l1/576.0
  112 write (6,8)dx2
    8 format("(must be less than ",f8.3," m)")
      read*, dd
      print*, 'darcy friction factor ?'
      read*, f
      print*, 'stanton number ?'
      read*, st
      print*, 'wall temperature (celcius) ?'
      read*, tw
```

```
      print*, 'atmospheric temperature (celcius) ?'
      read*, tat
      tat=tat+273.16
      print*, 'atmospheric pressure (kpa) ?'
      read*, pat
      pat=pat*1000
c
c       Calculate initial isothermal flow conditions along the pipe
c
    4 dx1=dd*64
      ga=cp/(cp-r)
      aat=dsqrt(1.4*287*tat)
      n1=ifix(l1/dx1+0.5)
      n2=ifix(l2/dx1+0.5)
      m1=n1+121
      m2=n2+121
      ar=0.25*pi*d*d
      call stead1 (p1,t1,u1,z1,ro1,r,e,d,pc,tc,th,f,g,pi,dx1,m1,n1)
      ao=dsqrt(ga*p1(m1)/ro1(m1))
      if(l2.eq.0.0) goto 113
      p2(1)=p1(m1)
      t2(1)=t1(m1)
      u2(1)=u1(m1)
      z2(1)=z1(m1)
      ro2(1)=ro1(m1)
      call stead2 (p2,t2,u2,z2,ro2,r,e,d,pc,tc,th,f,g,dx1,m2,n2)
c
c       Calculate wavespeed at time to at i=1 in pipe 1
c
  113 zp1(1)=(z1(1)-1)/p1(1)
      zt1(1)=(81*tc**3/(64*t1(1)**4)-9*tc/(128*t1(1)**2))*p1(1)/pc
      as1(1)=((1+zt1(1)*t1(1)/z1(1))**2)*p1(1)/(ro1(1)*t1(1)*cp)
      as1(1)=(1-zp1(1)*p1(1)/z1(1)-as1(1))*ro1(1)/p1(1)
      as1(1)=1/dsqrt(dabs(as1(1)))
c
c       Read in the run data
c
    6 print*, 'length of time step required (msecs.) ?'
      grad=dx1*500./as1(1)
      write(6,7)grad
    7 format("time step must be less than",f8.3,"msecs")
      read*, dt1
      if(dt1.le.grad) goto 9
      print*,'time step exceeds the stability criterion'
      goto 6
    9 print*, 'total run time required (secs.) ?'
      read*, tm
      dt1=dt1/1000
c
      print*,'*************************************************************'
      print*,' '
      print*,'              Transient Analysis Results'
      print*,' '
      print*,'*************************************************************'
c
c       Set number of transducer points for printing out results at
c
      nint=12
      write(42,99)nint
   99 format(i4)
c
c       Transient calculations for pipe 1 (upstream of the break)
c
```

```
      kk=0
      jk=0
   53 do 70 j=1,64
      ti=ti+dt1/64
      do 5 i=1,m1
      z1(i)=9*tc/(128*t1(i))-27*tc**3/(64*t1(i)**3)
      z1(i)=z1(i)*p1(i)/pc+1
      zp1(i)=(z1(i)-1)/p1(i)
      zt1(i)=81*tc**3/(64*t1(i)**4)-9*tc/(128*t1(i)*t1(i))
      zt1(i)=zt1(i)*p1(i)/pc
      ro1(i)=p1(i)/(r*t1(i)*z1(i))
      w1(i)=dabs(ar*ro1(i)*f*u1(i)*u1(i))/(2*d)
      as1(i)=((1+zt1(i)*t1(i)/z1(i))**2)*p1(i)
      as1(i)=as1(i)/(ro1(i)*t1(i)*cp)+zp1(i)*p1(i)/z1(i)
      as1(i)=(1-as1(i))*ro1(i)/p1(i)
      as1(i)=1/dsqrt(dabs(as1(i)))
      ht1(i)=pi*cp*st*d*ro1(i)*u1(i)*(tw-t1(i))
      at1(i)=(1+t1(i)*zt1(i)/z1(i))**2
      at1(i)=at1(i)/(1-p1(i)*zp1(i)/z1(i))
      at1(i)=(1-at1(i)*z1(i)*r/cp)*as1(i)*as1(i)
      at1(i)=dsqrt(dabs(at1(i)))-dabs(u1(i))
    5 continue
      do 10 i=1,m2
      if(12.eq.0.0) goto 10
      z2(i)=9*tc/(128*t2(i))-27*tc**3/(64*t2(i)**3)
      z2(i)=z2(i)*p2(i)/pc+1
      zp2(i)=(z2(i)-1)/p2(i)
      zt2(i)=81*tc**3/(64*t2(i)**4)-9*tc/(128*t2(i)*t2(i))
      zt2(i)=zt2(i)*p2(i)/pc
      ro2(i)=p2(i)/(r*t2(i)*z2(i))
      w2(i)=dabs(ar*ro2(i)*f*u2(i)*u2(i))/(2*d)
      as2(i)=((1+zt2(i)*t2(i)/z2(i))**2)*p2(i)
      as2(i)=as2(i)/(ro2(i)*t2(i)*cp)+zp2(i)*p2(i)/z2(i)
      as2(i)=(1-as2(i))*ro2(i)/p2(i)
      as2(i)=1/dsqrt(dabs(as2(i)))
      ht2(i)=pi*cp*st*d*ro2(i)*u2(i)*(tw-t2(i))
      at2(i)=(1+t2(i)*zt2(i)/z2(i))**2
      at2(i)=at2(i)/(1-p2(i)*zp2(i)/z2(i))
      at2(i)=(1-at2(i)*z2(i)*r/cp)*as2(i)*as2(i)
      at2(i)=dsqrt(dabs(at2(i)))-dabs(u2(i))
   10 continue
      if(jk.eq.0) goto 116
      tis=ti-65*dt1/64
      write(6,56)tis
   56 format(/,f8.6,'secs. after break, the results in pipe 1 are:-')
      write(42,98)tis
   98 format(f8.6)
      do 52 i=1,m1
      pps1(i)=p1(i)/1000
      tts1(i)=t1(i)-273.16
      wave1(i)=as1(i)-u1(i)
      if(i.eq.299) goto 881
      if(i.eq.172) goto 881
      if(i.eq.157) goto 881
      if(i.eq.140) goto 881
      if(i.eq.123) goto 881
      if(i.eq.106) goto 881
      goto 52
  881 ii=i+1000
      write(42,55)ii,pps1(i),u1(i),tts1(i),wave1(i),at1(i)
   52 continue
      if(12.eq.0.0) goto 116
      do 46 i=1,m2
```

```
      tts2(i)=t2(i)-273.16
      wave2(i)=as2(i)-dabs(u2(i))
      if(i.eq.1) goto 882
      if(i.eq.128) goto 882
      if(i.eq.143) goto 882
      if(i.eq.160) goto 882
      if(i.eq.177) goto 882
      if(i.eq.194) goto 882
      goto 46
  882 ii=i+2000
      write(42,55)ii,pps2(i),u2(i),tts2(i),wave2(i),at2(i)
   46 continue
  116 jk=0
      dx=dx1/64
      dt=dt1/64
      do 37 i=1,m1
      pp1(i)=p1(i)
      tt1(i)=t1(i)
      uu1(i)=u1(i)
   37 continue
      do 40 i=1,m2
      if(12.eq.0.0) goto 40
      pp2(i)=p2(i)
      tt2(i)=t2(i)
      uu2(i)=u2(i)
   40 continue
      do 12 ii=1,m1
      i=ii
      if(i.lt.(m1-63)) goto 12
      if(i.gt.(m1-1)) goto 13
      call sub1(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      goto 12
   13 if(ti.gt.(81*dt1)) goto 58
      if(ti.gt.dt1) goto 92
      if(12.eq.0.0) goto 120
      call break1(p1,t1,u1,p2,t2,u2,pp1,tt1,uu1,z1,zt1,ro1,as1,ht1,
     & w1,dx,tc,pc,r,ar,f,d,cp,pi,st,tw,g,dt,i,th,z2,zt2,ro2,as2,
     & ht2,w2)
  120 if(j.ne.64) goto 12
      psave=pp1(m1)
      usave=uu1(m1)
      tsave=tt1(m1)
      pec=((2/(ga+1))**(2*ga/(ga-1)))*psave
      y1=(2/(ga-1))*as1(m1)*(1-(pec/pp1(m1))**((ga-1)/(2*ga)))
      y2=dsqrt(1.4*((2.4)*(pec/pat)+0.4)/2)
      y2=aat*((pec/pat)-1)/y2
      if(y1.gt.y2) goto 71
      peq=pec
      ueq=y1
      teq=2*tt1(m1)/(ga+1)
      goto 12
   71 pe=(50+42*y1*y1/(aat*aat))**2-100*(25-7*y1*y1/(aat*aat))
      pe=(50+42*y1*y1/(aat*aat)+dsqrt(pe))/50
      if(dabs(pe-pec).lt.pe/100) goto 72
      pec=pe
      y1=(2/(ga-1))*as1(m1)*(1-(pec/pp1(m1)))**((ga-1)/(2*ga))
      goto 71
   72 peq=pe
      ueq=y1
      teq=tt1(m1)*((pe/pp1(m1))**((ga-1)/(2*ga)))
      goto 12
   55 format(i4,1x,5f14.4)
```

```
      pp1(m1)=(psave-peq)*((1.0-kk/5120.0)**2)+peq
   58 call break3(p1,t1,u1,pp1,tt1,uu1,z1,zt1,ro1,as1,ht1,w1,
     & dx,tc,pc,r,ar,f,d,cp,pi,st,tw,g,dt,i,th,pat,tat)
   12 continue
      k=1
      j1=(j/2)*2
      if (j1.eq.j) goto 14
      i=m1-64
      px(1)=p1(m1-64)
      tx(1)=t1(m1-64)
      ux(1)=u1(m1-64)
      call sub2(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      py(1)=pp1(m1-64)
      ty(1)=tt1(m1-64)
      uy(1)=uu1(m1-64)
      goto 11
   14 j1=j1/2
      dx=dx*2
      dt=dt*2
      do 15 ii=1,(m1-64)
      i=ii
      if(i.lt.(m1-95)) goto 15
      if (i.ge.(m1-65)) goto 16
      call sub1(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      goto 15
   16 zx(k)=(9*tc/(128*tx(k))-27*tc**3/(64*tx(k)**3))*px(k)/pc+1
      zpx(k)=(zx(k)-1)/px(k)
      ztx(k)=(81*tc**3/(64*tx(k)**4)-9*tc/(128*tx(k)*tx(k)))*px(k)/pc
      rox(k)=px(k)/(r*tx(k)*zx(k))
      wx(k)=dabs(ar*rox(k)*f*ux(k)*ux(k))/(2*d)
      asx(k)=((1+tx(k)*ztx(k)/zx(k))**2)*px(k)/(rox(k)*tx(k)*cp)
      asx(k)=asx(k)+zpx(k)*px(k)/zx(k)-1
      asx(k)=1/dsqrt(dabs(asx(k)*rox(k)/px(k)))
      htx(k)=pi*cp*st*d*rox(k)*ux(k)*(tw-tx(k))
      if(i.gt.(m1-65)) goto 81
      p1(i+1)=px(k)
      u1(i+1)=ux(k)
      t1(i+1)=tx(k)
      z1(i+1)=zx(k)
      w1(i+1)=wx(k)
      zt1(i+1)=ztx(k)
      ro1(i+1)=rox(k)
      as1(i+1)=asx(k)
      ht1(i+1)=htx(k)
      call sub1(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      goto 15
   81 dt=dt/2
      dx=dx/2
      zy(k)=(9*tc/(128*ty(k))-27*tc**3/(64*ty(k)**3))*py(k)/pc+1
      zpy(k)=(zy(k)-1)/py(k)
      zty(k)=(81*tc**3/(64*ty(k)**4)-9*tc/(128*ty(k)*ty(k)))*py(k)/pc
      roy(k)=py(k)/(r*ty(k)*zy(k))
      wy(k)=dabs(ar*roy(k)*f*uy(k)*uy(k))/(2*d)
      asy(k)=((1+ty(k)*zty(k)/zy(k))**2)*py(k)/(roy(k)*ty(k)*cp)
      asy(k)=asy(k)+zpy(k)*py(k)/zy(k)-1
      asy(k)=1/dsqrt(dabs(asy(k)*roy(k)/py(k)))
      hty(k)=pi*cp*st*d*roy(k)*uy(k)*(tw-ty(k))
      call sub5(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th,px,ux,tx,wx,zx,
     & ztx,rox,asx,htx,py,uy,ty,wy,zy,zty,roy,asy,hty,k)
```

```
      dx=dx*2
   15 continue
      k=2
      j2=(j1/2)*2
      if (j2.eq.j1) goto 17
      i=(m1-96)
      px(2)=p1(m1-96)
      tx(2)=t1(m1-96)
      ux(2)=u1(m1-96)
      call sub2(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      py(2)=pp1(m1-96)
      ty(2)=tt1(m1-96)
      uy(2)=uu1(m1-96)
      goto 11
   17 j1=j1/2
      dx=dx*2
      dt=dt*2
      do 18 ii=1,(m1-96)
      i=ii
      if(i.lt.(m1-111)) goto 18
      if(i.ge.(m1-97)) goto 19
      call sub1(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      goto 18
   19 zx(k)=(9*tc/(128*tx(k))-27*tc**3/(64*tx(k)**3))*px(k)/pc+1
      zpx(k)=(zx(k)-1)/px(k)
      ztx(k)=(81*tc**3/(64*tx(k)**4)-9*tc/(128*tx(k)*tx(k)))*px(k)/pc
      rox(k)=px(k)/(r*tx(k)*zx(k))
      wx(k)=dabs(ar*rox(k)*f*ux(k)*ux(k))/(2*d)
      asx(k)=((1+tx(k)*ztx(k)/zx(k))**2)*px(k)/(rox(k)*tx(k)*cp)
      asx(k)=asx(k)+zpx(k)*px(k)/zx(k)-1
      asx(k)=1/dsqrt(dabs(asx(k)*rox(k)/px(k)))
      htx(k)=pi*cp*st*d*rox(k)*ux(k)*(tw-tx(k))
      if(i.gt.(m1-97)) goto 82
      p1(i+1)=px(k)
      u1(i+1)=ux(k)
      t1(i+1)=tx(k)
      z1(i+1)=zx(k)
      w1(i+1)=wx(k)
      zt1(i+1)=ztx(k)
      ro1(i+1)=rox(k)
      as1(i+1)=asx(k)
      ht1(i+1)=htx(k)
      call sub1(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      goto 18
   82 dt=dt/2
      dx=dx/2
      zy(k)=(9*tc/(128*ty(k))-27*tc**3/(64*ty(k)**3))*py(k)/pc+1
      zpy(k)=(zy(k)-1)/py(k)
      zty(k)=(81*tc**3/(64*ty(k)**4)-9*tc/(128*ty(k)*ty(k)))*py(k)/pc
      roy(k)=py(k)/(r*ty(k)*zy(k))
      wy(k)=dabs(ar*roy(k)*f*uy(k)*uy(k))/(2*d)
      asy(k)=((1+ty(k)*zty(k)/zy(k))**2)*py(k)/(roy(k)*ty(k)*cp)
      asy(k)=asy(k)+zpy(k)*py(k)/zy(k)-1
      asy(k)=1/dsqrt(dabs(asy(k)*roy(k)/py(k)))
      hty(k)=pi*cp*st*d*roy(k)*uy(k)*(tw-ty(k))
      call sub5(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th,px,ux,tx,wx,zx,
     & ztx,rox,asx,htx,py,uy,ty,wy,zy,zty,roy,asy,hty,k)
      dt=dt*2
      dx=dx*2
```

```
      k=3
      j2=(j1/2)*2
      if (j2.eq.j1) goto 20
      i=m1-112
      px(3)=p1(m1-112)
      tx(3)=t1(m1-112)
      ux(3)=u1(m1-112)
      call sub2(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      py(3)=pp1(m1-112)
      ty(3)=tt1(m1-112)
      uy(3)=uu1(m1-112)
      goto 11
20    j1=j1/2
      dx=dx*2
      dt=dt*2
      do 21 ii=1,m1-112
      i=ii
      if(i.lt.(m1-119)) goto 21
      if(i.ge.(m1-113)) goto 22
      call sub1(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      goto 21
22    zx(k)=(9*tc/(128*tx(k))-27*tc**3/(64*tx(k)**3))*px(k)/pc+1
      zpx(k)=(zx(k)-1)/px(k)
      ztx(k)=(81*tc**3/(64*tx(k)**4)-9*tc/(128*tx(k)*tx(k)))*px(k)/pc
      rox(k)=px(k)/(r*tx(k)*zx(k))
      wx(k)=dabs(ar*rox(k)*f*ux(k)*ux(k))/(2*d)
      asx(k)=((1+tx(k)*ztx(k)/zx(k))**2)*px(k)/(rox(k)*tx(k)*cp)
      asx(k)=asx(k)+zpx(k)*px(k)/zx(k)-1
      asx(k)=1/dsqrt(dabs(asx(k)*rox(k)/px(k)))
      htx(k)=pi*cp*st*d*rox(k)*ux(k)*(tw-tx(k))
      if(i.gt.(m1-113)) goto 83
      p1(i+1)=px(k)
      u1(i+1)=ux(k)
      t1(i+1)=tx(k)
      z1(i+1)=zx(k)
      w1(i+1)=wx(k)
      zt1(i+1)=ztx(k)
      ro1(i+1)=rox(k)
      as1(i+1)=asx(k)
      ht1(i+1)=htx(k)
      call sub1(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      goto 21
83    dt=dt/2
      dx=dx/2
      zy(k)=(9*tc/(128*ty(k))-27*tc**3/(64*ty(k)**3))*py(k)/pc+1
      zpy(k)=(zy(k)-1)/py(k)
      zty(k)=(81*tc**3/(64*ty(k)**4)-9*tc/(128*ty(k)*ty(k)))*py(k)/pc
      roy(k)=py(k)/(r*ty(k)*zy(k))
      wy(k)=dabs(ar*roy(k)*f*uy(k)*uy(k))/(2*d)
      asy(k)=((1+ty(k)*zty(k)/zy(k))**2)*py(k)/(roy(k)*ty(k)*cp)
      asy(k)=asy(k)+zpy(k)*py(k)/zy(k)-1
      asy(k)=1/dsqrt(dabs(asy(k)*roy(k)/py(k)))
      hty(k)=pi*cp*st*d*roy(k)*uy(k)*(tw-ty(k))
      call sub5(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th,px,ux,tx,wx,zx,
     & ztx,rox,asx,htx,py,uy,ty,wy,zy,zty,roy,asy,hty,k)
      dt=dt*2
      dx=dx*2
21    continue
      k=4
```

```
      if (j2.eq.j1) goto 23
      i=m1-120
      px(4)=p1(m1-120)
      tx(4)=t1(m1-120)
      ux(4)=u1(m1-120)
      call sub2(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      py(4)=pp1(m1-120)
      ty(4)=tt1(m1-120)
      uy(4)=uu1(m1-120)
      goto 11
23    j1=j1/2
      dx=dx*2
      dt=dt*2
      do 24 ii=1,m1-120
      i=ii
      if(i.lt.(m1-123)) goto 24
      if(i.ge.(m1-121)) goto 25
      call sub1(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      goto 24
25    zx(k)=(9*tc/(128*tx(k))-27*tc**3/(64*tx(k)**3))*px(k)/pc+1
      zpx(k)=(zx(k)-1)/px(k)
      ztx(k)=(81*tc**3/(64*tx(k)**4)-9*tc/(128*tx(k)*tx(k)))*px(k)/pc
      rox(k)=px(k)/(r*tx(k)*zx(k))
      wx(k)=dabs(ar*rox(k)*f*ux(k)*ux(k))/(2*d)
      asx(k)=((1+tx(k)*ztx(k)/zx(k))**2)*px(k)/(rox(k)*tx(k)*cp)
      asx(k)=asx(k)+zpx(k)*px(k)/zx(k)-1
      asx(k)=1/dsqrt(dabs(asx(k)*rox(k)/px(k)))
      htx(k)=pi*cp*st*d*rox(k)*ux(k)*(tw-tx(k))
      if(i.gt.(m1-121)) goto 84
      p1(i+1)=px(k)
      u1(i+1)=ux(k)
      t1(i+1)=tx(k)
      z1(i+1)=zx(k)
      w1(i+1)=wx(k)
      zt1(i+1)=ztx(k)
      ro1(i+1)=rox(k)
      as1(i+1)=asx(k)
      ht1(i+1)=htx(k)
      call sub1(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      goto 24
84    dt=dt/2
      dx=dx/2
      zy(k)=(9*tc/(128*ty(k))-27*tc**3/(64*ty(k)**3))*py(k)/pc+1
      zpy(k)=(zy(k)-1)/py(k)
      zty(k)=(81*tc**3/(64*ty(k)**4)-9*tc/(128*ty(k)*ty(k)))*py(k)/pc
      roy(k)=py(k)/(r*ty(k)*zy(k))
      wy(k)=dabs(ar*roy(k)*f*uy(k)*uy(k))/(2*d)
      asy(k)=((1+ty(k)*zty(k)/zy(k))**2)*py(k)/(roy(k)*ty(k)*cp)
      asy(k)=asy(k)+zpy(k)*py(k)/zy(k)-1
      asy(k)=1/dsqrt(dabs(asy(k)*roy(k)/py(k)))
      hty(k)=pi*cp*st*d*roy(k)*uy(k)*(tw-ty(k))
      call sub5(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th,px,ux,tx,wx,zx,
     & ztx,rox,asx,htx,py,uy,ty,wy,zy,zty,roy,asy,hty,k)
      dt=dt*2
      dx=dx*2
24    continue
      k=5
      j2=(j1/2)*2
      if (j2.eq.j1) goto 26
```

```
      px(5)=p1(m1-124)
      tx(5)=t1(m1-124)
      ux(5)=u1(m1-124)
      call sub2(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      py(5)=pp1(m1-124)
      ty(5)=tt1(m1-124)
      uy(5)=uu1(m1-124)
      goto 11
   26 j1=j1/2
      dx=dx*2
      dt=dt*2
      do 27 ii=1,m1-124
      i=ii
      if(i.lt.(m1-125)) goto 27
      zx(k)=(9*tc/(128*tx(k))-27*tc**3/(64*tx(k)**3))*px(k)/pc+1
      zpx(k)=(zx(k)-1)/px(k)
      ztx(k)=(81*tc**3/(64*tx(k)**4)-9*tc/(128*tx(k)*tx(k)))*px(k)/pc
      rox(k)=px(k)/(r*tx(k)*zx(k))
      wx(k)=dabs(ar*rox(k)*f*ux(k)*ux(k))/(2*d)
      asx(k)=((1+tx(k)*ztx(k)/zx(k))**2)*px(k)/(rox(k)*tx(k)*cp)
      asx(k)=asx(k)+zpx(k)*px(k)/zx(k)-1
      asx(k)=1/dsqrt(dabs(asx(k)*rox(k)/px(k)))
      htx(k)=pi*cp*st*d*rox(k)*ux(k)*(tw-tx(k))
      if(i.gt.(m1-125)) goto 85
      p1(i+1)=px(k)
      u1(i+1)=ux(k)
      t1(i+1)=tx(k)
      z1(i+1)=zx(k)
      w1(i+1)=wx(k)
      zt1(i+1)=ztx(k)
      ro1(i+1)=rox(k)
      as1(i+1)=asx(k)
      ht1(i+1)=htx(k)
      call sub1(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      goto 27
   85 dt=dt/2
      dx=dx/2
      zy(k)=(9*tc/(128*ty(k))-27*tc**3/(64*ty(k)**3))*py(k)/pc+1
      zpy(k)=(zy(k)-1)/py(k)
      zty(k)=(81*tc**3/(64*ty(k)**4)-9*tc/(128*ty(k)*ty(k)))*py(k)/pc
      roy(k)=py(k)/(r*ty(k)*zy(k))
      wy(k)=dabs(ar*roy(k)*f*uy(k)*uy(k))/(2*d)
      asy(k)=((1+ty(k)*zty(k)/zy(k))**2)*py(k)/(roy(k)*ty(k)*cp)
      asy(k)=asy(k)+zpy(k)*py(k)/zy(k)-1
      asy(k)=1/dsqrt(dabs(asy(k)*roy(k)/py(k)))
      hty(k)=pi*cp*st*d*roy(k)*uy(k)*(tw-ty(k))
      call sub5(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th,px,ux,tx,wx,zx,
     & ztx,rox,asx,htx,py,uy,ty,wy,zy,zty,roy,asy,hty,k)
      dt=dt*2
      dx=dx*2
   27 continue
      k=6
      j2=(j1/2)*2
      if (j2.eq.j1) goto 29
      i=m1-126
      px(6)=p1(m1-126)
      tx(6)=t1(m1-126)
      ux(6)=u1(m1-126)
      p1(i+2)=px(5)
      u1(i+2)=ux(5)
```

```
      z1(i+2)=zx(5)
      w1(i+2)=wx(5)
      zt1(i+2)=ztx(5)
      ro1(i+2)=rox(5)
      as1(i+2)=asx(5)
      ht1(i+2)=htx(5)
      call sub2(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      py(6)=pp1(m1-126)
      ty(6)=tt1(m1-126)
      uy(6)=uu1(m1-126)
      goto 11
   29 j1=j1/2
      dx=dx*2
      dt=dt*2
      do 30 ii=1,(m1-126)
      i=ii
      if(i.gt.1) goto 31
      call subup(p1,t1,u1,dx,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th,pp1,tt1,uu1)
      goto 30
   31 if(i.lt.(m1-127)) goto 90
      zx(k)=(9*tc/(128*tx(k))-27*tc**3/(64*tx(k)**3))*px(k)/pc+1
      zpx(k)=(zx(k)-1)/px(k)
      ztx(k)=(81*tc**3/(64*tx(k)**4)-9*tc/(128*tx(k)*tx(k)))*px(k)/pc
      rox(k)=px(k)/(r*tx(k)*zx(k))
      wx(k)=dabs(ar*rox(k)*f*ux(k)*ux(k))/(2*d)
      asx(k)=((1+tx(k)*ztx(k)/zx(k))**2)*px(k)/(rox(k)*tx(k)*cp)
      asx(k)=asx(k)+zpx(k)*px(k)/zx(k)-1
      asx(k)=1/dsqrt(dabs(asx(k)*rox(k)/px(k)))
      htx(k)=pi*cp*st*d*rox(k)*ux(k)*(tw-tx(k))
      if(i.gt.(m1-127)) goto 86
      p1(i+1)=px(k)
      u1(i+1)=ux(k)
      t1(i+1)=tx(k)
      z1(i+1)=zx(k)
      w1(i+1)=wx(k)
      zt1(i+1)=ztx(k)
      ro1(i+1)=rox(k)
      as1(i+1)=asx(k)
      ht1(i+1)=htx(k)
   90 call sub1(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th)
      goto 30
   86 dt=dt/2
      dx=dx/2
      zy(k)=(9*tc/(128*ty(k))-27*tc**3/(64*ty(k)**3))*py(k)/pc+1
      zpy(k)=(zy(k)-1)/py(k)
      zty(k)=(81*tc**3/(64*ty(k)**4)-9*tc/(128*ty(k)*ty(k)))*py(k)/pc
      roy(k)=py(k)/(r*ty(k)*zy(k))
      wy(k)=dabs(ar*roy(k)*f*uy(k)*uy(k))/(2*d)
      asy(k)=((1+ty(k)*zty(k)/zy(k))**2)*py(k)/(roy(k)*ty(k)*cp)
      asy(k)=asy(k)+zpy(k)*py(k)/zy(k)-1
      asy(k)=1/dsqrt(dabs(asy(k)*roy(k)/py(k)))
      hty(k)=pi*cp*st*d*roy(k)*uy(k)*(tw-ty(k))
      call sub5(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,ar,f,d,cp,pi,st,
     & tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th,px,ux,tx,wx,zx,
     & ztx,rox,asx,htx,py,uy,ty,wy,zy,zty,roy,asy,hty,k)
      dt=dt*2
      dx=dx*2
   30 continue
c
c       Transient calculations for pipe 2 (downstream of the break)
```

```
 11 dx=dx1/64
    dt=dt1/64
    if(12.eq.0.0) goto 32
    do 33 ii=1,64
    i=ii
    if(i.gt.1) goto 34
    if(ti.gt.dt1) goto 95
    call break2(pp1,tt1,uu1,pp2,tt2,uu2,m1,i)
    goto 33
 95 pp2(1)=pp1(m1)
    call break4(p2,t2,u2,pp2,tt2,uu2,z2,zt2,ro2,as2,ht2,w2,dx,tc,pc,
   & r,ar,f,d,cp,pi,st,tw,g,dt,i,th,pat,tt1,m1)
    goto 33
 34 call sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
   & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
 33 continue
    k=1
    j1=(j/2)*2
    if(j1.eq.j) goto 35
    i=65
    pz(1)=p2(i)
    uz(1)=u2(i)
    tz(1)=t2(i)
    zz(1)=z2(i)
    ztz(1)=zt2(i)
    roz(1)=ro2(i)
    asz(1)=as2(i)
    htz(1)=ht2(i)
    wz(1)=w2(i)
    call sub3(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
   & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
    goto 32
 35 j1=j1/2
    dx=dx*2
    dt=dt*2
    do 36 ii=1,96
    i=ii
    if(i.le.66) goto 36
    call sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
   & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
 36 continue
    i=65
    dx=dx/2
    dt=dt/2
    call sub6(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
   & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th,pz,tz,uz,zz,
   & ztz,roz,asz,htz,wz,k)
    dx=dx*2
    dt=dt*2
    i=66
    p2(i-1)=pz(1)
    t2(i-1)=tz(1)
    u2(i-1)=uz(1)
    z2(i-1)=zz(1)
    w2(i-1)=wz(1)
    zt2(i-1)=ztz(1)
    ro2(i-1)=roz(1)
    as2(i-1)=asz(1)
    ht2(i-1)=htz(1)
    call sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
   & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
    k=2
    j2=(j1/2)*2
```

```
      i=97
      pz(2)=p2(i)
      uz(2)=u2(i)
      tz(2)=t2(i)
      zz(2)=z2(i)
      ztz(2)=zt2(i)
      roz(2)=ro2(i)
      asz(2)=as2(i)
      htz(2)=ht2(i)
      wz(2)=w2(i)
      call sub3(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
      goto 32
   38 j1=j1/2
      dx=dx*2
      dt=dt*2
      do 39 ii=1,112
      i=ii
      if(i.le.98) goto 39
      call sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
   39 continue
      i=97
      dx=dx/2
      dt=dt/2
      call sub6(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th,pz,tz,uz,zz,
     & ztz,roz,asz,htz,wz,k)
      dx=dx*2
      dt=dt*2
      i=98
      p2(i-1)=pz(2)
      t2(i-1)=tz(2)
      u2(i-1)=uz(2)
      z2(i-1)=zz(2)
      w2(i-1)=wz(2)
      zt2(i-1)=ztz(2)
      ro2(i-1)=roz(2)
      as2(i-1)=asz(2)
      ht2(i-1)=htz(2)
      call sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
      k=3
      j2=(j1/2)*2
      if(j2.eq.j1) goto 41
      i=113
      pz(3)=p2(i)
      uz(3)=u2(i)
      tz(3)=t2(i)
      zz(3)=z2(i)
      ztz(3)=zt2(i)
      roz(3)=ro2(i)
      asz(3)=as2(i)
      htz(3)=ht2(i)
      wz(3)=w2(i)
      call sub3(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
      goto 32
   41 j1=j1/2
      dx=dx*2
      dt=dt*2
      do 42 ii=1,120
      i=ii
```

```
      call sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
   42 continue
      i=113
      dx=dx/2
      dt=dt/2
      call sub6(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th,pz,tz,uz,zz,
     & ztz,roz,asz,htz,wz,k)
      dx=dx*2
      dt=dt*2
      i=114
      p2(i-1)=pz(3)
      t2(i-1)=tz(3)
      u2(i-1)=uz(3)
      z2(i-1)=zz(3)
      w2(i-1)=wz(3)
      zt2(i-1)=ztz(3)
      ro2(i-1)=roz(3)
      as2(i-1)=asz(3)
      ht2(i-1)=htz(3)
      call sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
      k=4
      j2=(j1/2)*2
      if(j2.eq.j1) goto 44
      i=121
      pz(4)=p2(i)
      uz(4)=u2(i)
      tz(4)=t2(i)
      zz(4)=z2(i)
      ztz(4)=zt2(i)
      roz(4)=ro2(i)
      asz(4)=as2(i)
      htz(4)=ht2(i)
      wz(4)=w2(i)
      call sub3(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
      goto 32
   44 j1=j1/2
      dx=dx*2
      dt=dt*2
      do 45 ii=1,124
      i=ii
      if(i.le.122) goto 45
      call sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
   45 continue
      i=121
      dx=dx/2
      dt=dt/2
      call sub6(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th,pz,tz,uz,zz,
     & ztz,roz,asz,htz,wz,k)
      dx=dx*2
      dt=dt*2
      i=122
      p2(i-1)=pz(4)
      t2(i-1)=tz(4)
      u2(i-1)=uz(4)
      z2(i-1)=zz(4)
      w2(i-1)=wz(4)
      zt2(i-1)=ztz(4)
```

```
      as2(i-1)=asz(4)
      ht2(i-1)=htz(4)
      call sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
      k=5
      j2=(j1/2)*2
      if(j2.eq.j1) goto 47
      i=125
      pz(5)=p2(i)
      uz(5)=u2(i)
      tz(5)=t2(i)
      zz(5)=z2(i)
      ztz(5)=zt2(i)
      roz(5)=ro2(i)
      asz(5)=as2(i)
      htz(5)=ht2(i)
      wz(5)=w2(i)
      call sub3(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
      goto 32
47    j1=j1/2
      i=125
      call sub6(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th,pz,tz,uz,zz,
     & ztz,roz,asz,htz,wz,k)
      dx=dx*2
      dt=dt*2
      i=126
      p2(i-1)=pz(5)
      t2(i-1)=tz(5)
      u2(i-1)=uz(5)
      z2(i-1)=zz(5)
      w2(i-1)=wz(5)
      zt2(i-1)=ztz(5)
      ro2(i-1)=roz(5)
      as2(i-1)=asz(5)
      ht2(i-1)=htz(5)
      call sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
      k=6
      j2=(j1/2)*2
      if(j2.eq.j1) goto 50
      i=127
      pz(6)=p2(i)
      uz(6)=u2(i)
      tz(6)=t2(i)
      zz(6,)=z2(i)
      ztz(6)=zt2(i)
      roz(6)=ro2(i)
      asz(6)=as2(i)
      htz(6)=ht2(i)
      wz(6)=w2(i)
      call sub3(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
      goto 32
50    j1=j1/2
      dx=dx*2
      dt=dt*2
      do 51 ii=1,m2-1
      i=ii
      if(i.le.128) goto 51
      call sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
```

```
      i=127
      dx=dx/2
      dt=dt/2
      call sub6(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th,pz,tz,uz,zz,
     & ztz,roz,asz,htz,wz,k)
      dx=dx*2
      dt=dt*2
      i=128
      p2(i-1)=pz(6)
      t2(i-1)=tz(6)
      u2(i-1)=uz(6)
      z2(i-1)=zz(6)
      w2(i-1)=wz(6)
      zt2(i-1)=ztz(6)
      ro2(i-1)=roz(6)
      as2(i-1)=asz(6)
      ht2(i-1)=asz(6)
      call sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2,th)
      i=m2
      call down1(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,th,f,d,cp,pi,
     & st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,ht2)
32    do 60 i=1,m1
      p1(i)=pp1(i)
      t1(i)=tt1(i)
      u1(i)=uu1(i)
60    continue
      do 59 i=1,m2
      if(l2.eq.0.0) goto 59
      p2(i)=pp2(i)
      t2(i)=tt2(i)
      u2(i)=uu2(i)
59    continue
      if(ti.lt.(6*dt1)) goto 70
70    continue
      jk=1
      if(tis.le.tm) goto 53
      print*,' '
      print*,' '
      print*,'                          End of run'
      stop
      end
```

```
c
      subroutine steadi (p1,t1,u1,z1,ro1,r,e,d,pc,tc,th,f,g,
     & pi,dx1,m1,n1)
c
c       This subroutine calculates steady isothermal flow
c       along pipe 1.  (upstream of the break)
c
      implicit double precision  (a-h,o-z)
      dimension p1(300),u1(300),ro1(300),z1(300),t1(300)
c
      print*,'initial conditions required:-'
      print*,'initial temperature along the pipe (celcius)'
      read*, t1(1)
      t1(1)=t1(1)+273.16
      print*,'initial pressure at upstream end of pipe (kpa)'
      read*, p1(1)
      p1(1)=p1(1)*1000
      print*,'mass flow rate through the pipe (kg/s)'
      read*, flow
c
      z1(1)=9*tc/(128*t1(1))-27*tc**3/(64*t1(1)**3)
      z1(1)=z1(1)*p1(1)/pc +1
      ro1(1)=p1(1)/(z1(1)*r*t1(1))
      e=flow/(.25*pi*d*d)
      u1(1)=e/ro1(1)
c
      do 2 i=1,m1-1
      if (i.ge.(m1-126)) goto 3
      dx=dx1
    1 if(flow.eq.0.0) goto 9
      aa=f*e*dx/(4.0d0*d)+e
      bb=f*e*dx*u1(i)/(4.0d0*d)+dx*ro1(i)*g*dsin(th)/2.0d0
      bb=bb-e*u1(i)-p1(i)
      cc=e*dx*g*dsin(th)/2.0d0+e*r*t1(i)*z1(i)
      u1(i+1)=(-bb-dsqrt(bb*bb-4*aa*cc))/(2*aa)
      ro1(i+1)=e/u1(i+1)
      p1(i+1)=ro1(i+1)*z1(i)*r*t1(i)
      z1(i+1)=9*tc/(128*t1(i))-27*tc**3/(64*t1(i)**3)
      z1(i+1)=z1(i+1)*p1(i+1)/pc +1
      t1(i+1)=t1(i)
      goto 2
c
    9 u1(i+1)=0.0
      p1(i+1)=p1(i)-dx1*dsin(th)
      t1(i+1)=t1(i)
      z1(i+1)=9*tc/(128*t1(i))-27*tc**3/(64*t1(i)**3)
      z1(i+1)=z1(i+1)*p1(i+1)/pc+1
      ro1(i+1)=p1(i+1)/(z1(i+1)*r*t1(i+1))
      goto 2
c
    3 if (i.ge.(m1-124)) goto 4
      dx=dx1/2.
      goto 1
    4 if (i.ge.(m1-120)) goto 5
      dx=dx1/4.
      goto 1
    5 if (i.ge.(m1-112)) goto 6
      dx=dx1/8.
      goto 1
    6 if (i.ge.(m1-96)) goto 7
      dx=dx1/16.
      goto 1
    7 if (i.ge.(m1-64)) goto 8
```

```
      goto 1
    8 dx=dx1/64.
      goto 1
    2 continue
c
      return
      end
```

```
c
      subroutine stead2 (p2, t2, u2, z2, ro2, r, e, d, pc, tc, th, f, g,
     &  dx1, m2, n2)
c
c       This subroutine calculates steady isothermal flow
c       along pipe 2.  (downstream of the break)
c
      implicit double precision  (a-h, o-z)
      dimension p2(300), u2(300), ro2(300), z2(300), t2(300)
c
      do 2 i=1, m2-1
      if (i.lt.127) goto 3
      dx=dx1
    1 if(u2(1).eq.0.0) goto 9
      aa=f*e*dx/(4*d)+e
      bb=f*e*dx*u2(i)/(4*d)+dx*ro2(i)*g*dsin(th)/2
      bb=bb-e*u2(i)-p2(i)
      cc=e*dx*g*dsin(th)/2+e*r*t2(i)*z2(i)
      u2(i+1)=(-bb-dsqrt(bb**2-4*aa*cc))/(2*aa)
      ro2(i+1)=e/u2(i+1)
      p2(i+1)=ro2(i+1)*z2(i)*r*t2(i)
      z2(i+1)=9*tc/(128*t2(i))-27*tc**3/(64*t2(i)**3)
      z2(i+1)=z2(i+1)*p2(i+1)/pc +1
      t2(i+1)=t2(i)
      goto 2
c
    9 u2(i+1)=0.0
      p2(i+1)=p2(i)-dx1*dsin(th)
      t2(i+1)=t2(i)
      z2(i+1)=9*tc/(128*t2(i))-27*tc**3/(64*t2(i)**3)
      z2(i+1)=z2(i+1)*p2(i+1)/pc+1
      ro2(i+1)=p2(i+1)/(z2(i+1)*r*t2(i))
      goto 2
c
    3 if (i.lt.125) goto 4
      dx=dx1/2
      goto 1
    4 if (i.lt.121) goto 5
      dx=dx1/4
      goto 1
    5 if (i.lt.113) goto 6
      dx=dx1/8
      goto 1
    6 if (i.lt.97) goto 7
      dx=dx1/16
      goto 1
    7 if (i.lt.65) goto 8
      dx=dx1/32
      goto 1
    8 dx=dx1/64
      goto 1
    2 continue
c
      continue
      return
      end
```

```
c
      subroutine sub1(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,
     & ar,f,d,cp,pi,st,tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,
     & ht1,th)
c
c       This subroutine calculates p,t and u at normal
c       internal points upstream of the break.
c
      implicit double precision (a-h,o-z)
      parameter(ni = 300)
      dimension p1(ni),t1(ni),u1(ni),z1(ni),
     & zt1(ni),w1(ni),as1(ni),ht1(ni),pp1(ni),
     & tt1(ni),uu1(ni),zz1(ni),zzp1(ni),zzt1(ni),
     & rro1(ni),ww1(ni),aas1(ni),hht1(ni),
     & a(9),b(3),ro1(ni),zp1(ni)
      integer ll(3),mm(3),count
c
c       First order approximation
c
      if(u1(i).eq.0.0) goto 20
      if(u1(i-1).eq.0.0) goto 20
      posq=2*dt/(1/u1(i)+1/u1(i-1))
      goto 22
   20 posq=dt*(u1(i)+u1(i-1))/2
   22 posr=dt*2/(1/(u1(i)+as1(i))+1/(u1(i-1)+as1(i-1)))
      poss=dt*2/(1/(as1(i)-u1(i))+1/(as1(i+1)-u1(i+1)))
c
      pq=posq/dx*p1(i-1)+(1-posq/dx)*p1(i)
      tq=posq/dx*t1(i-1)+(1-posq/dx)*t1(i)
      uq=posq/dx*u1(i-1)+(1-posq/dx)*u1(i)
      zq=posq/dx*z1(i-1)+(1-posq/dx)*z1(i)
      ztq=posq/dx*zt1(i-1)+(1-posq/dx)*zt1(i)
      roq=posq/dx*ro1(i-1)+(1-posq/dx)*ro1(i)
      asq=posq/dx*as1(i-1)+(1-posq/dx)*as1(i)
      htq=posq/dx*ht1(i-1)+(1-posq/dx)*ht1(i)
      wq=posq/dx*w1(i-1)+(1-posq/dx)*w1(i)
      pr=posr/dx*p1(i-1)+(1-posr/dx)*p1(i)
      tr=posr/dx*t1(i-1)+(1-posr/dx)*t1(i)
      ur=posr/dx*u1(i-1)+(1-posr/dx)*u1(i)
      zr=posr/dx*z1(i-1)+(1-posr/dx)*z1(i)
      ztr=posr/dx*zt1(i-1)+(1-posr/dx)*zt1(i)
      ror=posr/dx*ro1(i-1)+(1-posr/dx)*ro1(i)
      asr=posr/dx*as1(i-1)+(1-posr/dx)*as1(i)
      htr=posr/dx*ht1(i-1)+(1-posr/dx)*ht1(i)
      wr=posr/dx*w1(i-1)+(1-posr/dx)*w1(i)
      ps=poss/dx*p1(i+1)+(1-poss/dx)*p1(i)
      ts=poss/dx*t1(i+1)+(1-poss/dx)*t1(i)
      us=poss/dx*u1(i+1)+(1-poss/dx)*u1(i)
      zs=poss/dx*z1(i+1)+(1-poss/dx)*z1(i)
      zts=poss/dx*zt1(i+1)+(1-poss/dx)*zt1(i)
      ros=poss/dx*ro1(i+1)+(1-poss/dx)*ro1(i)
      ass=poss/dx*as1(i+1)+(1-poss/dx)*as1(i)
      hts=poss/dx*ht1(i+1)+(1-poss/dx)*ht1(i)
      ws=poss/dx*w1(i+1)+(1-poss/dx)*w1(i)
c
      x1=asr*dt*(1+tr*ztr/zr)/(ror*cp*tr*ar)
      x2=ass*dt*(1+ts*zts/zs)/(ros*cp*ts*ar)
      a(1)=-(1+tq*ztq/zq)/(roq*cp)-wq*uq*dt/(2*roq*cp*ar*pq)
      a(2)=1+wq*uq*dt/(2*roq*cp*ar*tq)
      a(3)=-wq*dt/(roq*cp*ar)
      a(4)=1/(ror*asr)-wr*ur*x1/(2*pr)+wr*dt/(2*ar*ror*pr)
      a(5)=wr*ur*x1/(2*tr)-wr*dt/(2*ar*ror*tr)
      a(7)=-1/(ros*ass)+ws*us*x2/(2*ps)+ws*dt/(2*ar*ros*ps)
```

```
      if(ur.eq.0.0)goto 2
      if(us.eq.0.0) goto 2
      a(9)=1+ws*x2+ws*dt/(ar*ros*us)
      a(6)=1-wr*x1+wr*dt/(ar*ror*ur)
      goto 1
    2 a(6)=1
      a(9)=1
c
    1 b(1)=htq*dt/(roq*cp*ar)+tq-pq*(1+tq*ztq/zq)/(roq*cp)
      b(2)=htr*x1+ur+pr/(ror*asr)-g*dt*dsin(th)
      b(3)=-hts*x2+us-ps/(ros*ass)-g*dt*dsin(th)
c
      call dminv(a,3,det,11,mm)
      if(det.ne.0.0)goto 9
      write(6,30) i
   30 format('pipe1 i=',i3)
      stop "no inverse"
    9 pp1(i)=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tt1(i)=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uu1(i)=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      count=0
      psave=pp1(i)
      tsave=tt1(i)
      usave=uu1(i)
      pdif=pp1(i)*1000
      tdif=tt1(i)*1000
   10 zz1(i)=9*tc/(128*tt1(i))-27*tc**3/(64*tt1(i)**3)
      zz1(i)=zz1(i)*pp1(i)/pc+1
      zzp1(i)=(zz1(i)-1)/pp1(i)
      zzt1(i)=81*tc**3/(64*tt1(i)**4)-9*tc/(128*tt1(i)*tt1(i))
      zzt1(i)=zzt1(i)*pp1(i)/pc
      rro1(i)=pp1(i)/(r*tt1(i)*zz1(i))
      ww1(i)=dabs(ar*rro1(i)*f*uu1(i)*uu1(i))/(2*d)
      aas1(i)=((1+zzt1(i)*tt1(i)/zz1(i))**2)*pp1(i)
      aas1(i)=aas1(i)/(rro1(i)*tt1(i)*cp)+zzp1(i)*pp1(i)/zz1(i)
      aas1(i)=(1-aas1(i))*rro1(i)/pp1(i)
      aas1(i)=1/dsqrt(dabs(aas1(i)))
      hht1(i)=pi*cp*st*d*rro1(i)*uu1(i)*(tw-tt1(i))
      count=count + 1
c
c      Second order procedure
c
      if(uq.eq.0.0) goto 21
      if(uu1(i).eq.0.0) goto 21
      posq=2*dt/(1/uq+1/uu1(i))
      goto 23
   21 posq=dt*(uq+uu1(i))/2
   23 posr=2*dt/(1/(ur+asr)+1/(uu1(i)+aas1(i)))
      poss=2*dt/(1/(ass-us)+1/(aas1(i)-uu1(i)))
c
      pq=p1(i)-posq*(p1(i+1)-p1(i-1))/(2*dx)
      tq=t1(i)-posq*(t1(i+1)-t1(i-1))/(2*dx)
      uq=u1(i)-posq*(u1(i+1)-u1(i-1))/(2*dx)
      zq=z1(i)-posq*(z1(i+1)-z1(i-1))/(2*dx)
      ztq=zt1(i)-posq*(zt1(i+1)-zt1(i-1))/(2*dx)
      roq=ro1(i)-posq*(ro1(i+1)-ro1(i-1))/(2*dx)
      asq=as1(i)-posq*(as1(i+1)-as1(i-1))/(2*dx)
      htq=ht1(i)-posq*(ht1(i+1)-ht1(i-1))/(2*dx)
      wq=w1(i)-posq*(w1(i+1)-w1(i-1))/(2*dx)
      pr=p1(i)-posr*(p1(i+1)-p1(i-1))/(2*dx)
      tr=t1(i)-posr*(t1(i+1)-t1(i-1))/(2*dx)
      ur=u1(i)-posr*(u1(i+1)-u1(i-1))/(2*dx)
      zr=z1(i)-posr*(z1(i+1)-z1(i-1))/(2*dx)
```

```
      ror=ro1(i)-posr*(ro1(i+1)-ro1(i-1))/(2*dx)
      asr=as1(i)-posr*(as1(i+1)-as1(i-1))/(2*dx)
      htr=ht1(i)-posr*(ht1(i+1)-ht1(i-1))/(2*dx)
      wr=w1(i)-posr*(w1(i+1)-w1(i-1))/(2*dx)
      ps=p1(i)+poss*(p1(i+1)-p1(i-1))/(2*dx)
      ts=t1(i)+poss*(t1(i+1)-t1(i-1))/(2*dx)
      us=u1(i)+poss*(u1(i+1)-u1(i-1))/(2*dx)
      zs=z1(i)+poss*(z1(i+1)-z1(i-1))/(2*dx)
      zts=zt1(i)+poss*(zt1(i+1)-zt1(i-1))/(2*dx)
      ros=ro1(i)+poss*(ro1(i+1)-ro1(i-1))/(2*dx)
      ass=as1(i)+poss*(as1(i+1)-as1(i-1))/(2*dx)
      hts=ht1(i)+poss*(ht1(i+1)-ht1(i-1))/(2*dx)
      ws=w1(i)+poss*(w1(i+1)-w1(i-1))/(2*dx)
      pq=pq+posq*posq*(p1(i+1)+p1(i-1)-2*p1(i))/(2*dx*dx)
      tq=tq+posq*posq*(t1(i+1)+t1(i-1)-2*t1(i))/(2*dx*dx)
      uq=uq+posq*posq*(u1(i+1)+u1(i-1)-2*u1(i))/(2*dx*dx)
      zq=zq+posq*posq*(z1(i+1)+z1(i-1)-2*z1(i))/(2*dx*dx)
      ztq=ztq+posq*posq*(zt1(i+1)+zt1(i-1)-2*zt1(i))/(2*dx*dx)
      roq=roq+posq*posq*(ro1(i+1)+ro1(i-1)-2*ro1(i))/(2*dx*dx)
      asq=asq+posq*posq*(as1(i+1)+as1(i-1)-2*as1(i))/(2*dx*dx)
      htq=htq+posq*posq*(ht1(i+1)+ht1(i-1)-2*ht1(i))/(2*dx*dx)
      wq=wq+posq*posq*(w1(i+1)+w1(i-1)-2*w1(i))/(2*dx*dx)
      pr=pr+posr*posr*(p1(i+1)+p1(i-1)-2*p1(i))/(2*dx*dx)
      tr=tr+posr*posr*(t1(i+1)+t1(i-1)-2*t1(i))/(2*dx*dx)
      ur=ur+posr*posr*(u1(i+1)+u1(i-1)-2*u1(i))/(2*dx*dx)
      zr=zr+posr*posr*(z1(i+1)+z1(i-1)-2*z1(i))/(2*dx*dx)
      ztr=ztr+posr*posr*(zt1(i+1)+zt1(i-1)-2*zt1(i))/(2*dx*dx)
      ror=ror+posr*posr*(ro1(i+1)+ro1(i-1)-2*ro1(i))/(2*dx*dx)
      asr=asr+posr*posr*(as1(i+1)+as1(i-1)-2*as1(i))/(2*dx*dx)
      htr=htr+posr*posr*(ht1(i+1)+ht1(i-1)-2*ht1(i))/(2*dx*dx)
      wr=wr+posr*posr*(w1(i+1)+w1(i-1)-2*w1(i))/(2*dx*dx)
      ps=ps+poss*poss*(p1(i+1)+p1(i-1)-2*p1(i))/(2*dx*dx)
      ts=ts+poss*poss*(t1(i+1)+t1(i-1)-2*t1(i))/(2*dx*dx)
      us=us+poss*poss*(u1(i+1)+u1(i-1)-2*u1(i))/(2*dx*dx)
      zs=zs+poss*poss*(z1(i+1)+z1(i-1)-2*z1(i))/(2*dx*dx)
      zts=zts+poss*poss*(zt1(i+1)+zt1(i-1)-2*zt1(i))/(2*dx*dx)
      ros=ros+poss*poss*(ro1(i+1)+ro1(i-1)-2*ro1(i))/(2*dx*dx)
      ass=ass+poss*poss*(as1(i+1)+as1(i-1)-2*as1(i))/(2*dx*dx)
      hts=hts+poss*poss*(ht1(i+1)+ht1(i-1)-2*ht1(i))/(2*dx*dx)
      ws=ws+poss*poss*(w1(i+1)+w1(i-1)-2*w1(i))/(2*dx*dx)
C
      a(1)=-(1+tt1(i)*zzt1(i)/zz1(i))/rro1(i)
      a(1)=(a(1)-(1+tq*ztq/zq)/roq)/(2*cp)
      a(2)=1.0
      a(3)=0.0
      a(4)=(1/(ror*asr)+1/(rro1(i)*aas1(i)))/2
      a(5)=0.0
      a(6)=1.0
      a(7)=(-1/(ros*ass)-1/(rro1(i)*aas1(i)))/2
      a(8)=0.0
      a(9)=1.0
C
      b(1)=(htq+wq*uq)/roq+(hht1(i)+ww1(i)*uu1(i))/rro1(i)
      b(1)=b(1)*dt/(2*cp*ar)+a(1)*pq+tq
      b(2)=(1+tt1(i)*zzt1(i)/zz1(i))*(hht1(i)+ww1(i)*uu1(i))
      b(2)=b(2)*aas1(i)/(rro1(i)*tt1(i))
      b(2)=(b(2)+asr*(1+tr*ztr/zr)*(htr+wr*ur)/(ror*tr))/cp
      b(2)=(b(2)-(wr/ror+ww1(i)/rro1(i)))*dt/(2*ar)
      b(2)=b(2)-g*dt*dsin(th)+a(4)*pr+ur
      b(3)=(1+tt1(i)*zzt1(i)/zz1(i))*(hht1(i)+ww1(i)*uu1(i))
      b(3)=b(3)*aas1(i)/(rro1(i)*tt1(i))
      b(3)=(-b(3)-ass*(1+ts*zts/zs)*(hts+ws*us)/(ros*ts))/cp
      b(3)=(b(3)-(ws/ros+ww1(i)/rro1(i)))*dt/(2*ar)
```

```
c
      call dminv(a,3,det,11,mm)
      if(det.ne.0.0) goto 12
      write(6,31) i
   31 format('pipe 1 (2nd order) i=',i3)
      stop "no inverse"
   12 pit=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tit=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uit=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      difp=dabs(pp1(i)-pit)/pit
      dift=dabs(tt1(i)-tit)/tit
      if(count.gt.200) goto 15
      if(difp.gt.0.01) goto 13
      if(dift.lt.0.01) goto 14
   13 pp1(i)=pit
      tt1(i)=tit
      uu1(i)=uit
      pdif=difp
      tdif=dift
      goto 10
c
   15 write(6,16)i
   16 format('sub1 - no iteration for i=',i4,' in pipe 1')
   14 pp1(i)=pit
      tt1(i)=tit
      uu1(i)=uit
      return
      end
```

```
      subroutine sub2(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,
     & ar,f,d,cp,pi,st,tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,
     & ht1,th)
c
c       this subroutine calculates p,t and u at internal
c       boundary points between different grid sizes.
c
      implicit double precision (a-h,o-z)
      dimension p1(300),t1(300),u1(300),z1(300),
     & zt1(300),w1(300),as1(300),ht1(300),pp1(300),
     & tt1(300),uu1(300),zz1(300),zzp1(300),zzt1(300),
     & rro1(300),ww1(300),aas1(300),hht1(300),
     & a(9),b(3),ro1(300),zp1(300)
      integer l1(3),mm(3),count
c
c       first order approximation
c
      if(u1(i).eq.0.0) goto 20
      if(u1(i-1).eq.0.0) goto 20
      posq=2*dt/(1/u1(i)+1/u1(i-1))
      goto 22
   20 posq=dt*(u1(i)+u1(i-1))/2
   22 posr=dt*2/(1/(u1(i)+as1(i))+1/(u1(i-1)+as1(i-1)))
      poss=dt*2/(1/(as1(i)-u1(i))+1/(as1(i+1)-u1(i+1)))
c
      pq=posq/(2*dx)*p1(i-1)+(1-posq/(2*dx))*p1(i)
      tq=posq/(2*dx)*t1(i-1)+(1-posq/(2*dx))*t1(i)
      uq=posq/(2*dx)*u1(i-1)+(1-posq/(2*dx))*u1(i)
      zq=posq/(2*dx)*z1(i-1)+(1-posq/(2*dx))*z1(i)
      ztq=posq/(2*dx)*zt1(i-1)+(1-posq/(2*dx))*zt1(i)
      roq=posq/(2*dx)*ro1(i-1)+(1-posq/(2*dx))*ro1(i)
      asq=posq/(2*dx)*as1(i-1)+(1-posq/(2*dx))*as1(i)
      htq=posq/(2*dx)*ht1(i-1)+(1-posq/(2*dx))*ht1(i)
      wq=posq/(2*dx)*w1(i-1)+(1-posq/(2*dx))*w1(i)
      pr=posr/(2*dx)*p1(i-1)+(1-posr/(2*dx))*p1(i)
      tr=posr/(2*dx)*t1(i-1)+(1-posr/(2*dx))*t1(i)
      ur=posr/(2*dx)*u1(i-1)+(1-posr/(2*dx))*u1(i)
      zr=posr/(2*dx)*z1(i-1)+(1-posr/(2*dx))*z1(i)
      ztr=posr/(2*dx)*zt1(i-1)+(1-posr/(2*dx))*zt1(i)
      ror=posr/(2*dx)*ro1(i-1)+(1-posr/(2*dx))*ro1(i)
      asr=posr/(2*dx)*as1(i-1)+(1-posr/(2*dx))*as1(i)
      htr=posr/(2*dx)*ht1(i-1)+(1-posr/(2*dx))*ht1(i)
      wr=posr/(2*dx)*w1(i-1)+(1-posr/(2*dx))*w1(i)
      ps=poss/dx*p1(i+1)+(1-poss/dx)*p1(i)
      ts=poss/dx*t1(i+1)+(1-poss/dx)*t1(i)
      us=poss/dx*u1(i+1)+(1-poss/dx)*u1(i)
      zs=poss/dx*z1(i+1)+(1-poss/dx)*z1(i)
      zts=poss/dx*zt1(i+1)+(1-poss/dx)*zt1(i)
      ros=poss/dx*ro1(i+1)+(1-poss/dx)*ro1(i)
      ass=poss/dx*as1(i+1)+(1-poss/dx)*as1(i)
      hts=poss/dx*ht1(i+1)+(1-poss/dx)*ht1(i)
      ws=poss/dx*w1(i+1)+(1-poss/dx)*w1(i)
c
      x1=asr*dt*(1+tr*ztr/zr)/(ror*cp*tr*ar)
      x2=ass*dt*(1+ts*zts/zs)/(ros*cp*ts*ar)
      a(1)=-(1+tq*ztq/zq)/(roq*cp)-wq*uq*dt/(2*roq*cp*ar*pq)
      a(2)=1+wq*uq*dt/(2*roq*cp*ar*tq)
      a(3)=-wq*dt/(roq*cp*ar)
      a(4)=1/(ror*asr)-wr*ur*x1/(2*pr)+wr*dt/(2*ar*ror*pr)
      a(5)=wr*ur*x1/(2*tr)-wr*dt/(2*ar*ror*tr)
      a(7)=-1/(ros*ass)+ws*us*x2/(2*ps)+ws*dt/(2*ar*ros*ps)
      a(8)=-ws*us*x2/(2*ts)-ws*dt/(2*ar*ros*ts)
```

```
      if(us.eq.0.0) goto 2
      a(6)=1-wr*x1+wr*dt/(ar*ror*ur)
      a(9)=1+ws*x2+ws*dt/(ar*ros*us)
      goto 1
    2 a(6)=1
      a(9)=1
c
    1 b(1)=htq*dt/(roq*cp*ar)+tq-pq*(1+tq*ztq/zq)/(roq*cp)
      b(2)=htr*x1+ur+pr/(ror*asr)-g*dt*dsin(th)
      b(3)=-hts*x2+us-ps/(ros*ass)-g*dt*dsin(th)
c
    5 call dminv(a,3,det,ll,mm)
      if(det.ne.0.0)goto 9
      write(6,30) i
   30 format('pipe1 i=',i3)
      stop "no inverse"
    9 pp1(i)=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tt1(i)=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uu1(i)=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      count=0
      psave=pp1(i)
      tsave=tt1(i)
      usave=uu1(i)
      pdif=pp1(i)*1000
      tdif=tt1(i)*1000
   10 zz1(i)=9*tc/(128*tt1(i))-27*tc**3/(64*tt1(i)**3)
      zz1(i)=zz1(i)*pp1(i)/pc+1
      zzp1(i)=(zz1(i)-1)/pp1(i)
      zzt1(i)=81*tc**3/(64*tt1(i)**4)-9*tc/(128*tt1(i)*tt1(i))
      zzt1(i)=zzt1(i)*pp1(i)/pc
      rro1(i)=pp1(i)/(r*tt1(i)*zz1(i))
      ww1(i)=dabs(ar*rro1(i)*f*uu1(i)*uu1(i))/(2*d)
      aas1(i)=((1+zzt1(i)*tt1(i)/zz1(i))**2)*pp1(i)
      aas1(i)=aas1(i)/(rro1(i)*tt1(i)*cp)+zzp1(i)*pp1(i)/zz1(i)
      aas1(i)=(1-aas1(i))*rro1(i)/pp1(i)
      aas1(i)=1/dsqrt(dabs(aas1(i)))
      hht1(i)=pi*cp*st*d*rro1(i)*uu1(i)*(tw-tt1(i))
      count=count+1
c
c       second order procedure
c
      if(uq.eq.0.0) goto 21
      if(uu1(i).eq.0.0) goto 21
      posq=2*dt/(1/uq+1/uu1(i))
      goto 23
   21 posq=dt*(uq+uu1(i))/2
   23 posr=2*dt/(1/(ur+asr)+1/(uu1(i)+aas1(i)))
      poss=2*dt/(1/(ass-us)+1/(aas1(i)-uu1(i)))
c
      pq=p1(i-1)+(2*dx-posq)*(p1(i)-p1(i-2))/(4*dx)
      uq=u1(i-1)+(2*dx-posq)*(u1(i)-u1(i-2))/(4*dx)
      tq=t1(i-1)+(2*dx-posq)*(t1(i)-t1(i-2))/(4*dx)
      zq=z1(i-1)+(2*dx-posq)*(z1(i)-z1(i-2))/(4*dx)
      wq=w1(i-1)+(2*dx-posq)*(w1(i)-w1(i-2))/(4*dx)
      ztq=zt1(i-1)+(2*dx-posq)*(zt1(i)-zt1(i-2))/(4*dx)
      roq=ro1(i-1)+(2*dx-posq)*(ro1(i)-ro1(i-2))/(4*dx)
      asq=as1(i-1)+(2*dx-posq)*(as1(i)-as1(i-2))/(4*dx)
      htq=ht1(i-1)+(2*dx-posq)*(ht1(i)-ht1(i-2))/(4*dx)
      pr=p1(i-1)+(2*dx-posr)*(p1(i)-p1(i-2))/(4*dx)
      ur=u1(i-1)+(2*dx-posr)*(u1(i)-u1(i-2))/(4*dx)
      tr=t1(i-1)+(2*dx-posr)*(t1(i)-t1(i-2))/(4*dx)
      zr=z1(i-1)+(2*dx-posr)*(z1(i)-z1(i-2))/(4*dx)
      wr=w1(i-1)+(2*dx-posr)*(w1(i)-w1(i-2))/(4*dx)
```

```
      ror=ro1(i-1)+(2*dx-posr)*(ro1(i)-ro1(i-2))/(4*dx)
      asr=as1(i-1)+(2*dx-posr)*(as1(i)-as1(i-2))/(4*dx)
      htr=ht1(i-1)+(2*dx-posr)*(ht1(i)-ht1(i-2))/(4*dx)
      ps=p1(i+1)-(dx-poss)*(p1(i+2)-p1(i))/(2*dx)
      us=u1(i+1)-(dx-poss)*(u1(i+2)-u1(i))/(2*dx)
      ts=t1(i+1)-(dx-poss)*(t1(i+2)-t1(i))/(2*dx)
      zs=z1(i+1)-(dx-poss)*(z1(i+2)-z1(i))/(2*dx)
      ws=w1(i+1)-(dx-poss)*(w1(i+2)-w1(i))/(2*dx)
      zts=zt1(i+1)-(dx-poss)*(zt1(i+2)-zt1(i))/(2*dx)
      ros=ro1(i+1)-(dx-poss)*(ro1(i+2)-ro1(i))/(2*dx)
      ass=as1(i+1)-(dx-poss)*(as1(i+2)-as1(i))/(2*dx)
      hts=ht1(i+1)-(dx-poss)*(ht1(i+2)-ht1(i))/(2*dx)
      pq=pq+(2*dx-posq)**2*(p1(i)+p1(i-2)-2*p1(i-1))/(8*dx*dx)
      uq=uq+(2*dx-posq)**2*(u1(i)+u1(i-2)-2*u1(i-1))/(8*dx*dx)
      tq=tq+(2*dx-posq)**2*(t1(i)+t1(i-2)-2*t1(i-1))/(8*dx*dx)
      zq=zq+(2*dx-posq)**2*(z1(i)+z1(i-2)-2*z1(i-1))/(8*dx*dx)
      wq=wq+(2*dx-posq)**2*(w1(i)+w1(i-2)-2*w1(i-1))/(8*dx*dx)
      ztq=ztq+(2*dx-posq)**2*(zt1(i)+zt1(i-2)-2*zt1(i-1))/(8*dx*dx)
      roq=roq+(2*dx-posq)**2*(ro1(i)+ro1(i-2)-2*ro1(i-1))/(8*dx*dx)
      asq=asq+(2*dx-posq)**2*(as1(i)+as1(i-2)-2*as1(i-1))/(8*dx*dx)
      htq=htq+(2*dx-posq)**2*(ht1(i)+ht1(i-2)-2*ht1(i-1))/(8*dx*dx)
      pr=pr+(2*dx-posr)**2*(p1(i)+p1(i-2)-2*p1(i-1))/(8*dx*dx)
      ur=ur+(2*dx-posr)**2*(u1(i)+u1(i-2)-2*u1(i-1))/(8*dx*dx)
      tr=tr+(2*dx-posr)**2*(t1(i)+t1(i-2)-2*t1(i-1))/(8*dx*dx)
      zr=zr+(2*dx-posr)**2*(z1(i)+z1(i-2)-2*z1(i-1))/(8*dx*dx)
      wr=wr+(2*dx-posr)**2*(w1(i)+w1(i-2)-2*w1(i-1))/(8*dx*dx)
      ztr=ztr+(2*dx-posr)**2*(zt1(i)+zt1(i-2)-2*zt1(i-1))/(8*dx*dx)
      ror=ror+(2*dx-posr)**2*(ro1(i)+ro1(i-2)-2*ro1(i-1))/(8*dx*dx)
      asr=asr+(2*dx-posr)**2*(as1(i)+as1(i-2)-2*as1(i-1))/(8*dx*dx)
      htr=htr+(2*dx-posr)**2*(ht1(i)+ht1(i-2)-2*ht1(i-1))/(8*dx*dx)
      ps=ps+(dx-poss)**2*(p1(i+2)+p1(i)-2*p1(i+1))/(2*dx*dx)
      us=us+(dx-poss)**2*(u1(i+2)+u1(i)-2*u1(i+1))/(2*dx*dx)
      ts=ts+(dx-poss)**2*(t1(i+2)+t1(i)-2*t1(i+1))/(2*dx*dx)
      zs=zs+(dx-poss)**2*(z1(i+2)+z1(i)-2*z1(i+1))/(2*dx*dx)
      ws=ws+(dx-poss)**2*(w1(i+2)+w1(i)-2*w1(i+1))/(2*dx*dx)
      zts=zts+(dx-poss)**2*(zt1(i+2)+zt1(i)-2*zt1(i+1))/(2*dx*dx)
      ros=ros+(dx-poss)**2*(ro1(i+2)+ro1(i)-2*ro1(i+1))/(2*dx*dx)
      ass=ass+(dx-poss)**2*(as1(i+2)+as1(i)-2*as1(i+1))/(2*dx*dx)
      hts=hts+(dx-poss)**2*(ht1(i+2)+ht1(i)-2*ht1(i+1))/(2*dx*dx)
C
      a(1)=-(1+tt1(i)*zzt1(i)/zz1(i))/rro1(i)
      a(1)=(a(1)-(1+tq*ztq/zq)/roq)/(2*cp)
      a(2)=1.0
      a(3)=0.0
      a(4)=(1/(ror*asr)+1/(rro1(i)*aas1(i)))/2
      a(5)=0.0
      a(6)=1.0
      a(7)=(-1/(ros*ass)-1/(rro1(i)*aas1(i)))/2
      a(8)=0.0
      a(9)=1.0
C
      b(1)=(htq+wq*uq)/roq+(hht1(i)+ww1(i)*uu1(i))/rro1(i)
      b(1)=b(1)*dt/(2*cp*ar)+a(1)*pq+tq
      b(2)=(1+tt1(i)*zzt1(i)/zz1(i))*(hht1(i)+ww1(i)*uu1(i))
      b(2)=b(2)*aas1(i)/(rro1(i)*tt1(i))
      b(2)=(b(2)+asr*(1+tr*ztr/zr)*(htr+wr*ur)/(ror*tr))/cp
      b(2)=(b(2)-(wr/ror+ww1(i)/rro1(i)))*dt/(2*ar)
      b(2)=b(2)-g*dt*dsin(th)+a(4)*pr+ur
      b(3)=(1+tt1(i)*zzt1(i)/zz1(i))*(hht1(i)+ww1(i)*uu1(i))
      b(3)=b(3)*aas1(i)/(rro1(i)*tt1(i))
      b(3)=(-b(3)-ass*(1+ts*zts/zs)*(hts+ws*us)/(ros*ts))/cp
      b(3)=(b(3)-(ws/ros+ww1(i)/rro1(i)))*dt/(2*ar)
      b(3)=b(3)-g*dt*dsin(th)+a(7)*ps+us
```

```
      call dminv(a,3,det,11,mm)
      if(det.ne.0.0) goto 12
      write(6,31) i
   31 format('pipe2 (2nd order) i=',i3)
      stop "no inverse"
   12 pit=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tit=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uit=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      difp=dabs(pp1(i)-pit)/pit
      dift=dabs(tt1(i)-tit)/tit
      if(count.gt.200) goto 15
      if(difp.gt.0.01) goto 13
      if(dift.lt.0.01) goto 14
   13 pp1(i)=pit
      tt1(i)=tit
      uu1(i)=uit
      pdif=difp
      tdif=dift
      goto 10
C
   15 write(6,16)i
   16 format('sub2 - no iteration for i=',i4,' in pipe 1 (sub2)')
   14 pp1(i)=pit
      tt1(i)=tit
      uu1(i)=uit
      return
      end
```

```
c
      subroutine sub3(p2, t2, u2, dx, pp2, tt2, uu2, tc, pc, r,
     & ar, f, d, cp, pi, st, tw, g, dt, i, z2, zp2, zt2, ro2, w2, as2,
     & ht2, th)
c
c       This subroutine calculates p, t and u at internal
c       boundary points between different grid sizes in pipe 2.
c
      implicit double precision (a-h, o-z)
      dimension p2(300), t2(300), u2(300), z2(300),
     & zt2(300), w2(300), as2(300), ht2(300), pp2(300),
     & tt2(300), uu2(300), zz2(300), zzp2(300), zzt2(300),
     & rro2(300), ww2(300), aas2(300), hht2(300),
     & a(9), b(3), ro2(300), zp2(300)
      integer ll(3), mm(3), count
c
c       First order approximation
c
      posr=dt*2/(1/(u2(i)+as2(i))+1/(u2(i-1)+as2(i-1)))
      poss=dt*2/(1/(as2(i)-u2(i))+1/(as2(i+1)-u2(i+1)))
      pr=posr/dx*p2(i-1)+(1-posr/dx)*p2(i)
      tr=posr/dx*t2(i-1)+(1-posr/dx)*t2(i)
      ur=posr/dx*u2(i-1)+(1-posr/dx)*u2(i)
      zr=posr/dx*z2(i-1)+(1-posr/dx)*z2(i)
      ztr=posr/dx*zt2(i-1)+(1-posr/dx)*zt2(i)
      ror=posr/dx*ro2(i-1)+(1-posr/dx)*ro2(i)
      asr=posr/dx*as2(i-1)+(1-posr/dx)*as2(i)
      htr=posr/dx*ht2(i-1)+(1-posr/dx)*ht2(i)
      wr=posr/dx*w2(i-1)+(1-posr/dx)*w2(i)
      ps=poss/(2*dx)*p2(i+1)+(1-poss/(2*dx))*p2(i)
      ts=poss/(2*dx)*t2(i+1)+(1-poss/(2*dx))*t2(i)
      us=poss/(2*dx)*u2(i+1)+(1-poss/(2*dx))*u2(i)
      zs=poss/(2*dx)*z2(i+1)+(1-poss/(2*dx))*z2(i)
      zts=poss/(2*dx)*zt2(i+1)+(1-poss/(2*dx))*zt2(i)
      ros=poss/(2*dx)*ro2(i+1)+(1-poss/(2*dx))*ro2(i)
      ass=poss/(2*dx)*as2(i+1)+(1-poss/(2*dx))*as2(i)
      hts=poss/(2*dx)*ht2(i+1)+(1-poss/(2*dx))*ht2(i)
      ws=poss/(2*dx)*w2(i+1)+(1-poss/(2*dx))*w2(i)
      x1=asr*dt*(1+tr*ztr/zr)/(ror*cp*tr*ar)
      x2=ass*dt*(1+ts*zts/zs)/(ros*cp*ts*ar)
      a(4)=1/(ror*asr)-wr*ur*x1/(2*pr)+wr*dt/(2*ar*ror*pr)
      a(5)=wr*ur*x1/(2*tr)-wr*dt/(2*ar*ror*tr)
      a(7)=-1/(ros*ass)+ws*us*x2/(2*ps)+ws*dt/(2*ar*ros*ps)
      a(8)=-ws*us*x2/(2*ts)-ws*dt/(2*ar*ros*ts)
      if(ur.eq.0.0) goto 20
      if(us.eq.0.0) goto 21
      a(6)=1-wr*x1+wr*dt/(ar*ror*ur)
      a(9)=1+ws*x2+ws*dt/(ar*ros*us)
      goto 21
   20 a(6)=1
      a(9)=1
   21 b(2)=htr*x1+ur+pr/(ror*asr)-g*dt*dsin(th)
      b(3)=-hts*x2+us-ps/(ros*ass)-g*dt*dsin(th)
c
      if(u2(i).le.0.0) goto 1
      if(u2(i-1).eq.0.0) goto 7
      posq=dt*2/(1/u2(i)+1/u2(i-1))
      goto 17
    7 posq=dt*u2(i)/2
   17 pq=posq/dx*p2(i-1)+(1-posq/dx)*p2(i)
      tq=posq/dx*t2(i-1)+(1-posq/dx)*t2(i)
      uq=posq/dx*u2(i-1)+(1-posq/dx)*u2(i)
      zq=posq/dx*z2(i-1)+(1-posq/dx)*z2(i)
```

```
      roq=posq/dx*ro2(i-1)+(1-posq/dx)*ro2(i)
      asq=posq/dx*as2(i-1)+(1-posq/dx)*as2(i)
      htq=posq/dx*ht2(i-1)+(1-posq/dx)*ht2(i)
      wq=posq/dx*w2(i-1)+(1-posq/dx)*w2(i)
      goto 3
c
    1 if(u2(i).eq.0.0) goto 2
      if(u2(i+1).eq.0.0) goto 2
      posq=dabs(dt*2/(1/u2(i)+1/u2(i+1)))
      goto 5
    2 posq=dabs(dt*(u2(i)+u2(i+1))/2)
    5 pq=posq/(2*dx)*p2(i+1)+(1-posq/(2*dx))*p2(i)
      tq=posq/(2*dx)*t2(i+1)+(1-posq/(2*dx))*t2(i)
      uq=posq/(2*dx)*u2(i+1)+(1-posq/(2*dx))*u2(i)
      zq=posq/(2*dx)*z2(i+1)+(1-posq/(2*dx))*z2(i)
      ztq=posq/(2*dx)*zt2(i+1)+(1-posq/(2*dx))*zt2(i)
      roq=posq/(2*dx)*ro2(i+1)+(1-posq/(2*dx))*ro2(i)
      asq=posq/(2*dx)*as2(i+1)+(1-posq/(2*dx))*as2(i)
      htq=posq/(2*dx)*ht2(i+1)+(1-posq/(2*dx))*ht2(i)
      wq=posq/(2*dx)*w2(i+1)+(1-posq/(2*dx))*w2(i)
c
    3 a(1)=-(1+tq*ztq/zq)/(roq*cp)-wq*uq*dt/(2*roq*cp*ar*pq)
      a(2)=1+wq*uq*dt/(2*roq*cp*ar*tq)
      a(3)=-wq*dt/(roq*cp*ar)
      b(1)=htq*dt/(roq*cp*ar)+tq-pq*(1+tq*ztq/zq)/(roq*cp)
c
      call dminv(a,3,det,11,mm)
      if(det.ne.0.0)goto 9
      write(6,30) i
   30 format('pipe 2 i=',i3)
      stop "no inverse"
    9 pp2(i)=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tt2(i)=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uu2(i)=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      count=0
      psave=pp2(i)
      tsave=tt2(i)
      usave=uu2(i)
      pdif=pp2(i)*1000
      tdif=tt2(i)*1000
c
   10 zz2(i)=9*tc/(128*tt2(i))-27*tc**3/(64*tt2(i)**3)
      zz2(i)=zz2(i)*pp2(i)/pc+1
      zzp2(i)=(zz2(i)-1)/pp2(i)
      zzt2(i)=81*tc**3/(64*tt2(i)**4)-9*tc/(128*tt2(i)*tt2(i))
      zzt2(i)=zzt2(i)*pp2(i)/pc
      rro2(i)=pp2(i)/(r*tt2(i)*zz2(i))
      ww2(i)=dabs(ar*rro2(i)*f*uu2(i)*uu2(i))/(2*d)
      aas2(i)=((1+zzt2(i)*tt2(i)/zz2(i))**2)*pp2(i)
      aas2(i)=aas2(i)/(rro2(i)*tt2(i)*cp)+zzp2(i)*pp2(i)/zz2(i)
      aas2(i)=(1-aas2(i))*rro2(i)/pp2(i)
      aas2(i)=1/dsqrt(dabs(aas2(i)))
      hht2(i)=pi*cp*st*d*rro2(i)*uu2(i)*(tw-tt2(i))
      count=count+1
c
c      Second order procedure
c
      if(uq.eq.0.0) goto 8
      if(uu2(i).eq.0.0) goto 8
      posq=dabs(dt*2/(1/uq+1/uu2(i)))
      goto 11
    8 posq=dabs(dt*(uq+uu2(i))/2)
   11 posr=2*dt/(1/(ur+asr)+1/(uu2(i)+aas2(i)))
```

```
      pr=p2(i-1)+(dx-posr)*(p2(i)-p2(i-2))/(2*dx)
      ur=u2(i-1)+(dx-posr)*(u2(i)-u2(i-2))/(2*dx)
      tr=t2(i-1)+(dx-posr)*(t2(i)-t2(i-2))/(2*dx)
      zr=z2(i-1)+(dx-posr)*(z2(i)-z2(i-2))/(2*dx)
      wr=w2(i-1)+(dx-posr)*(w2(i)-w2(i-2))/(2*dx)
      ztr=zt2(i-1)+(dx-posr)*(zt2(i)-zt2(i-2))/(2*dx)
      ror=ro2(i-1)+(dx-posr)*(ro2(i)-ro2(i-2))/(2*dx)
      asr=as2(i-1)+(dx-posr)*(as2(i)-as2(i-2))/(2*dx)
      htr=ht2(i-1)+(dx-posr)*(ht2(i)-ht2(i-2))/(2*dx)
      ps=p2(i+1)-(2*dx-poss)*(p2(i+2)-p2(i))/(4*dx)
      us=u2(i+1)-(2*dx-poss)*(u2(i+2)-u2(i))/(4*dx)
      ts=t2(i+1)-(2*dx-poss)*(t2(i+2)-t2(i))/(4*dx)
      zs=z2(i+1)-(2*dx-poss)*(z2(i+2)-z2(i))/(4*dx)
      ws=w2(i+1)-(2*dx-poss)*(w2(i+2)-w2(i))/(4*dx)
      zts=zt2(i+1)-(2*dx-poss)*(zt2(i+2)-zt2(i))/(4*dx)
      ros=ro2(i+1)-(2*dx-poss)*(ro2(i+2)-ro2(i))/(4*dx)
      ass=as2(i+1)-(2*dx-poss)*(as2(i+2)-as2(i))/(4*dx)
      hts=ht2(i+1)-(2*dx-poss)*(ht2(i+2)-ht2(i))/(4*dx)
      pr=pr+(dx-posr)**2*(p2(i)+p2(i-2)-2*p2(i-1))/(2*dx*dx)
      ur=ur+(dx-posr)**2*(u2(i)+u2(i-2)-2*u2(i-1))/(2*dx*dx)
      tr=tr+(dx-posr)**2*(t2(i)+t2(i-2)-2*t2(i-1))/(2*dx*dx)
      zr=zr+(dx-posr)**2*(z2(i)+z2(i-2)-2*z2(i-1))/(2*dx*dx)
      wr=wr+(dx-posr)**2*(w2(i)+w2(i-2)-2*w2(i-1))/(2*dx*dx)
      ztr=ztr+(dx-posr)**2*(zt2(i)+zt2(i-2)-2*zt2(i-1))/(2*dx*dx)
      ror=ror+(dx-posr)**2*(ro2(i)+ro2(i-2)-2*ro2(i-1))/(2*dx*dx)
      asr=asr+(dx-posr)**2*(as2(i)+as2(i-2)-2*as2(i-1))/(2*dx*dx)
      htr=htr+(dx-posr)**2*(ht2(i)+ht2(i-2)-2*ht2(i-1))/(2*dx*dx)
      ps=ps+(2*dx-poss)**2*(p2(i+2)+p2(i)-2*p2(i+1))/(8*dx*dx)
      us=us+(2*dx-poss)**2*(u2(i+2)+u2(i)-2*u2(i+1))/(8*dx*dx)
      ts=ts+(2*dx-poss)**2*(t2(i+2)+t2(i)-2*t2(i+1))/(8*dx*dx)
      zs=zs+(2*dx-poss)**2*(z2(i+2)+z2(i)-2*z2(i+1))/(8*dx*dx)
      ws=ws+(2*dx-poss)**2*(w2(i+2)+w2(i)-2*w2(i+1))/(8*dx*dx)
      zts=zts+(2*dx-poss)**2*(zt2(i+2)+zt2(i)-2*zt2(i+1))/(8*dx*dx)
      ros=ros+(2*dx-poss)**2*(ro2(i+2)+ro2(i)-2*ro2(i+1))/(8*dx*dx)
      ass=ass+(2*dx-poss)**2*(as2(i+2)+as2(i)-2*as2(i+1))/(8*dx*dx)
      hts=hts+(2*dx-poss)**2*(ht2(i+2)+ht2(i)-2*ht2(i+1))/(8*dx*dx)
C
      if((uu2(i)+uq).lt.0.0) goto 4
      pq=p2(i-1)+(dx-posq)*(p2(i)-p2(i-2))/(2*dx)
      uq=u2(i-1)+(dx-posq)*(u2(i)-u2(i-2))/(2*dx)
      tq=t2(i-1)+(dx-posq)*(t2(i)-t2(i-2))/(2*dx)
      zq=z2(i-1)+(dx-posq)*(z2(i)-z2(i-2))/(2*dx)
      wq=w2(i-1)+(dx-posq)*(w2(i)-w2(i-2))/(2*dx)
      ztq=zt2(i-1)+(dx-posq)*(zt2(i)-zt2(i-2))/(2*dx)
      roq=ro2(i-1)+(dx-posq)*(ro2(i)-ro2(i-2))/(2*dx)
      asq=as2(i-1)+(dx-posq)*(as2(i)-as2(i-2))/(2*dx)
      htq=ht2(i-1)+(dx-posq)*(ht2(i)-ht2(i-2))/(2*dx)
      pq=pq+(dx-posq)**2*(p2(i)+p2(i-2)-2*p2(i-1))/(2*dx*dx)
      uq=uq+(dx-posq)**2*(u2(i)+u2(i-2)-2*u2(i-1))/(2*dx*dx)
      tq=tq+(dx-posq)**2*(t2(i)+t2(i-2)-2*t2(i-1))/(2*dx*dx)
      zq=zq+(dx-posq)**2*(z2(i)+z2(i-2)-2*z2(i-1))/(2*dx*dx)
      wq=wq+(dx-posq)**2*(w2(i)+w2(i-2)-2*w2(i-1))/(2*dx*dx)
      ztq=ztq+(dx-posq)**2*(zt2(i)+zt2(i-2)-2*zt2(i-1))/(2*dx*dx)
      roq=roq+(dx-posq)**2*(ro2(i)+ro2(i-2)-2*ro2(i-1))/(2*dx*dx)
      asq=asq+(dx-posq)**2*(as2(i)+as2(i-2)-2*as2(i-1))/(2*dx*dx)
      htq=htq+(dx-posq)**2*(ht2(i)+ht2(i-2)-2*ht2(i-1))/(2*dx*dx)
      goto 6
C
    4 pq=p2(i+1)-(2*dx-posq)*(p2(i+2)-p2(i))/(4*dx)
      uq=u2(i+1)-(2*dx-posq)*(u2(i+2)-u2(i))/(4*dx)
      tq=t2(i+1)-(2*dx-posq)*(t2(i+2)-t2(i))/(4*dx)
      zq=z2(i+1)-(2*dx-posq)*(z2(i+2)-z2(i))/(4*dx)
      wq=w2(i+1)-(2*dx-posq)*(w2(i+2)-w2(i))/(4*dx)
```

```
      roq=ro2(i+1)-(2*dx-posq)*(ro2(i+2)-ro2(i))/(4*dx)
      asq=as2(i+1)-(2*dx-posq)*(as2(i+2)-as2(i))/(4*dx)
      htq=ht2(i+1)-(2*dx-posq)*(ht2(i+2)-ht2(i))/(4*dx)
      pq=pq+(2*dx-posq)**2*(p2(i+2)+p2(i)-2*p2(i+1))/(8*dx*dx)
      uq=uq+(2*dx-posq)**2*(u2(i+2)+u2(i)-2*u2(i+1))/(8*dx*dx)
      tq=tq+(2*dx-posq)**2*(t2(i+2)+t2(i)-2*t2(i+1))/(8*dx*dx)
      zq=zq+(2*dx-posq)**2*(z2(i+2)+z2(i)-2*z2(i+1))/(8*dx*dx)
      wq=wq+(2*dx-posq)**2*(w2(i+2)+w2(i)-2*w2(i+1))/(8*dx*dx)
      ztq=ztq+(2*dx-posq)**2*(zt2(i+2)+zt2(i)-2*zt2(i+1))/(8*dx*dx)
      roq=roq+(2*dx-posq)**2*(ro2(i+2)+ro2(i)-2*ro2(i+1))/(8*dx*dx)
      asq=asq+(2*dx-posq)**2*(as2(i+2)+as2(i)-2*as2(i+1))/(8*dx*dx)
      htq=htq+(2*dx-posq)**2*(ht2(i+2)+ht2(i)-2*ht2(i+1))/(8*dx*dx)
c
    6 a(1)=-(1+tt2(i)*zzt2(i)/zz2(i))/rro2(i)
      a(1)=(a(1)-(1+tq*ztq/zq)/roq)/(2*cp)
      a(2)=1.0
      a(3)=0.0
      a(4)=(1/(ror*asr)+1/(rro2(i)*aas2(i)))/2
      a(5)=0.0
      a(6)=1.0
      a(7)=(-1/(ros*ass)-1/(rro2(i)*aas2(i)))/2
      a(8)=0.0
      a(9)=1.0
c
      b(1)=(htq+wq*uq)/roq+(hht2(i)+ww2(i)*uu2(i))/rro2(i)
      b(1)=b(1)*dt/(2*cp*ar)+a(1)*pq+tq
      b(2)=(1+tt2(i)*zzt2(i)/zz2(i))*(hht2(i)+ww2(i)*uu2(i))
      b(2)=b(2)*aas2(i)/(rro2(i)*tt2(i))
      b(2)=(b(2)+asr*(1+tr*ztr/zr)*(htr+wr*ur)/(ror*tr))/cp
      b(2)=(b(2)-(wr/ror+ww2(i)/rro2(i)))*dt/(2*ar)
      b(2)=b(2)-g*dt*dsin(th)+a(4)*pr+ur
      b(3)=(1+tt2(i)*zzt2(i)/zz2(i))*(hht2(i)+ww2(i)*uu2(i))
      b(3)=b(3)*aas2(i)/(rro2(i)*tt2(i))
      b(3)=(-b(3)-ass*(1+ts*zts/zs)*(hts+ws*us)/(ros*ts))/cp
      b(3)=(b(3)-(ws/ros+ww2(i)/rro2(i)))*dt/(2*ar)
      b(3)=b(3)-g*dt*dsin(th)+a(7)*ps+us
c
      call dminv(a,3,det,11,mm)
      if(det.ne.0.0) goto 12
      write(6,31) i
   31 format('pipe 2 (2nd order) i=',i3)
      stop "no inverse"
   12 pit=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tit=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uit=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      difp=dabs(pp2(i)-pit)/pit
      dift=dabs(tt2(i)-tit)/tit
      if(count.gt.200) goto 15
      if(difp.gt.0.01) goto 13
      if(dift.lt.0.01) goto 14
   13 pp2(i)=pit
      tt2(i)=tit
      uu2(i)=uit
      pdif=dfip
      tdif=dift
      goto 10
c
   15 write(6,16)i
   16 format('sub3 - no iteration for i=',i4,' in pipe 2')
   14 pp2(i)=pit
      tt2(i)=tit
      uu2(i)=uit
      return
      end
```

```
c
      subroutine sub4(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,
     & ar,f,d,cp,pi,st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,
     & ht2,th)
c
c       this subroutine calculates p,t and u at normal
c       internal points downstream of the break.
c
      implicit double precision (a-h,o-z)
      dimension p2(300),t2(300),u2(300),z2(300),
     & zt2(300),w2(300),as2(300),ht2(300),pp2(300),
     & tt2(300),uu2(300),zz2(300),zzp2(300),zzt2(300),
     & rro2(300),ww2(300),aas2(300),hht2(300),
     & a(9),b(3),ro2(300),zp2(300)
      integer ll(3),mm(3),count
c
c       first order approximation
c
      posr=dt*2/(1/(u2(i)+as2(i))+1/(u2(i-1)+as2(i-1)))
      poss=dt*2/(1/(as2(i)-u2(i))+1/(as2(i+1)-u2(i+1)))
      pr=posr/dx*p2(i-1)+(1-posr/dx)*p2(i)
      tr=posr/dx*t2(i-1)+(1-posr/dx)*t2(i)
      ur=posr/dx*u2(i-1)+(1-posr/dx)*u2(i)
      zr=posr/dx*z2(i-1)+(1-posr/dx)*z2(i)
      ztr=posr/dx*zt2(i-1)+(1-posr/dx)*zt2(i)
      ror=posr/dx*ro2(i-1)+(1-posr/dx)*ro2(i)
      asr=posr/dx*as2(i-1)+(1-posr/dx)*as2(i)
      htr=posr/dx*ht2(i-1)+(1-posr/dx)*ht2(i)
      wr=posr/dx*w2(i-1)+(1-posr/dx)*w2(i)
      ps=poss/dx*p2(i+1)+(1-poss/dx)*p2(i)
      ts=poss/dx*t2(i+1)+(1-poss/dx)*t2(i)
      us=poss/dx*u2(i+1)+(1-poss/dx)*u2(i)
      zs=poss/dx*z2(i+1)+(1-poss/dx)*z2(i)
      zts=poss/dx*zt2(i+1)+(1-poss/dx)*zt2(i)
      ros=poss/dx*ro2(i+1)+(1-poss/dx)*ro2(i)
      ass=poss/dx*as2(i+1)+(1-poss/dx)*as2(i)
      hts=poss/dx*ht2(i+1)+(1-poss/dx)*ht2(i)
      ws=poss/dx*w2(i+1)+(1-poss/dx)*w2(i)
      x1=asr*dt*(1+tr*ztr/zr)/(ror*cp*tr*ar)
      x2=ass*dt*(1+ts*zts/zs)/(ros*cp*ts*ar)
      a(4)=1/(ror*asr)-wr*ur*x1/(2*pr)+wr*dt/(2*ar*ror*pr)
      a(5)=wr*ur*x1/(2*tr)-wr*dt/(2*ar*ror*tr)
      a(7)=-1/(ros*ass)+ws*us*x2/(2*ps)+ws*dt/(2*ar*ros*ps)
      a(8)=-ws*us*x2/(2*ts)-ws*dt/(2*ar*ros*ts)
      if(ur.eq.0.0) goto 20
      if(us.eq.0.0) goto 20
      a(6)=1-wr*x1+wr*dt/(ar*ror*ur)
      a(9)=1+ws*x2+ws*dt/(ar*ros*us)
      goto 21
   20 a(6)=1
      a(9)=1
   21 b(2)=htr*x1+ur+pr/(ror*asr)-g*dt*dsin(th)
      b(3)=-hts*x2+us-ps/(ros*ass)-g*dt*dsin(th)
c
      if(u2(i).le.0.0) goto 1
      if(u2(i-1).eq.0.0) goto 1
      posq=dt*2/(1/u2(i)+1/u2(i-1))
      pq=posq/dx*p2(i-1)+(1-posq/dx)*p2(i)
      tq=posq/dx*t2(i-1)+(1-posq/dx)*t2(i)
      uq=posq/dx*u2(i-1)+(1-posq/dx)*u2(i)
      zq=posq/dx*z2(i-1)+(1-posq/dx)*z2(i)
      ztq=posq/dx*zt2(i-1)+(1-posq/dx)*zt2(i)
      roq=posq/dx*ro2(i-1)+(1-posq/dx)*ro2(i)
```

```
      htq=posq/dx*ht2(i-1)+(1-posq/dx)*ht2(i)
      wq=posq/dx*w2(i-1)+(1-posq/dx)*w2(i)
      goto 3
c
    1 if(u2(i).eq.0.0) goto 2
      if(u2(i+1).eq.0.0) goto 2
      posq=dabs(dt*2/(1/u2(i)+1/u2(i+1)))
      goto 7
    2 posq=dabs(dt*(u2(i)+u2(i+1))/2)
    7 pq=posq/dx*p2(i+1)+(1-posq/dx)*p2(i)
      tq=posq/dx*t2(i+1)+(1-posq/dx)*t2(i)
      uq=posq/dx*u2(i+1)+(1-posq/dx)*u2(i)
      zq=posq/dx*z2(i+1)+(1-posq/dx)*z2(i)
      ztq=posq/dx*zt2(i+1)+(1-posq/dx)*zt2(i)
      roq=posq/dx*ro2(i+1)+(1-posq/dx)*ro2(i)
      asq=posq/dx*as2(i+1)+(1-posq/dx)*as2(i)
      htq=posq/dx*ht2(i+1)+(1-posq/dx)*ht2(i)
      wq=posq/dx*w2(i+1)+(1-posq/dx)*w2(i)
c
    3 a(1)=-(1+tq*ztq/zq)/(roq*cp)-wq*uq*dt/(2*roq*cp*ar*pq)
      a(2)=1+wq*uq*dt/(2*roq*cp*ar*tq)
      a(3)=-wq*dt/(roq*cp*ar)
      b(1)=htq*dt/(roq*cp*ar)+tq-pq*(1+tq*ztq/zq)/(roq*cp)
c
    5 call dminv(a,3,det,11,mm)
      if(det.ne.0.0)goto 9
      write(6,30) i
   30 format('pipe 2 i=',i3)
      stop "no inverse"
    9 pp2(i)=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tt2(i)=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uu2(i)=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      count=0
      psave=pp2(i)
      tsave=tt2(i)
      usave=uu2(i)
      pdif=pp2(i)*1000
      tdif=tt2(i)*1000
c
   10 zz2(i)=9*tc/(128*tt2(i))-27*tc**3/(64*tt2(i)**3)
      zz2(i)=zz2(i)*pp2(i)/pc+1
      zzp2(i)=(zz2(i)-1)/pp2(i)
      zzt2(i)=81*tc**3/(64*tt2(i)**4)-9*tc/(128*tt2(i)*tt2(i))
      zzt2(i)=zzt2(i)*pp2(i)/pc
      rro2(i)=pp2(i)/(r*tt2(i)*zz2(i))
      ww2(i)=dabs(ar*rro2(i)*f*uu2(i)*uu2(i))/(2*d)
      aas2(i)=((1+zzt2(i)*tt2(i)/zz2(i))**2)*pp2(i)
      aas2(i)=aas2(i)/(rro2(i)*tt2(i)*cp)+zzp2(i)*pp2(i)/zz2(i)
      aas2(i)=(1-aas2(i))*rro2(i)/pp2(i)
      aas2(i)=1/dsqrt(dabs(aas2(i)))
      hht2(i)=pi*cp*st*d*rro2(i)*uu2(i)*(tw-tt2(i))
      count=count+1
c
c      second order procedure
c
      if(uq.eq.0.0) goto 8
      if(uu2(i).eq.0.0) goto 8
      posq=dabs(dt*2/(1/uq+1/uu2(i)))
      goto 11
    8 posq=dabs(dt*(uq+uu2(i))/2)
   11 posr=2*dt/(1/(ur+asr)+1/(uu2(i)+aas2(i)))
      poss=2*dt/(1/(ass-us)+1/(aas2(i)-uu2(i)))
      pr=p2(i)-posr*(p2(i+1)-p2(i-1))/(2*dx)
```

```
      ur=u2(i)-posr*(u2(i+1)-u2(i-1))/(2*dx)
      zr=z2(i)-posr*(z2(i+1)-z2(i-1))/(2*dx)
      ztr=zt2(i)-posr*(zt2(i+1)-zt2(i-1))/(2*dx)
      ror=ro2(i)-posr*(ro2(i+1)-ro2(i-1))/(2*dx)
      asr=as2(i)-posr*(as2(i+1)-as2(i-1))/(2*dx)
      htr=ht2(i)-posr*(ht2(i+1)-ht2(i-1))/(2*dx)
      wr=w2(i)-posr*(w2(i+1)-w2(i-1))/(2*dx)
      ps=p2(i)+poss*(p2(i+1)-p2(i-1))/(2*dx)
      ts=t2(i)+poss*(t2(i+1)-t2(i-1))/(2*dx)
      us=u2(i)+poss*(u2(i+1)-u2(i-1))/(2*dx)
      zs=z2(i)+poss*(z2(i+1)-z2(i-1))/(2*dx)
      zts=zt2(i)+poss*(zt2(i+1)-zt2(i-1))/(2*dx)
      ros=ro2(i)+poss*(ro2(i+1)-ro2(i-1))/(2*dx)
      ass=as2(i)+poss*(as2(i+1)-as2(i-1))/(2*dx)
      hts=ht2(i)+poss*(ht2(i+1)-ht2(i-1))/(2*dx)
      ws=w2(i)+poss*(w2(i+1)-w2(i-1))/(2*dx)
      pr=pr+posr*posr*(p2(i+1)+p2(i-1)-2*p2(i))/(2*dx*dx)
      tr=tr+posr*posr*(t2(i+1)+t2(i-1)-2*t2(i))/(2*dx*dx)
      ur=ur+posr*posr*(u2(i+1)+u2(i-1)-2*u2(i))/(2*dx*dx)
      zr=zr+posr*posr*(z2(i+1)+z2(i-1)-2*z2(i))/(2*dx*dx)
      ztr=ztr+posr*posr*(zt2(i+1)+zt2(i-1)-2*zt2(i))/(2*dx*dx)
      ror=ror+posr*posr*(ro2(i+1)+ro2(i-1)-2*ro2(i))/(2*dx*dx)
      asr=asr+posr*posr*(as2(i+1)+as2(i-1)-2*as2(i))/(2*dx*dx)
      htr=htr+posr*posr*(ht2(i+1)+ht2(i-1)-2*ht2(i))/(2*dx*dx)
      wr=wr+posr*posr*(w2(i+1)+w2(i-1)-2*w2(i))/(2*dx*dx)
      ps=ps+poss*poss*(p2(i+1)+p2(i-1)-2*p2(i))/(2*dx*dx)
      ts=ts+poss*poss*(t2(i+1)+t2(i-1)-2*t2(i))/(2*dx*dx)
      us=us+poss*poss*(u2(i+1)+u2(i-1)-2*u2(i))/(2*dx*dx)
      zs=zs+poss*poss*(z2(i+1)+z2(i-1)-2*z2(i))/(2*dx*dx)
      zts=zts+poss*poss*(zt2(i+1)+zt2(i-1)-2*zt2(i))/(2*dx*dx)
      ros=ros+poss*poss*(ro2(i+1)+ro2(i-1)-2*ro2(i))/(2*dx*dx)
      ass=ass+poss*poss*(as2(i+1)+as2(i-1)-2*as2(i))/(2*dx*dx)
      hts=hts+poss*poss*(ht2(i+1)+ht2(i-1)-2*ht2(i))/(2*dx*dx)
      ws=ws+poss*poss*(w2(i+1)+w2(i-1)-2*w2(i))/(2*dx*dx)
c
      if ((uu2(i)+uq).lt.0.0) goto 4
      pq=p2(i)-posq*(p2(i+1)-p2(i-1))/(2*dx)
      tq=t2(i)-posq*(t2(i+1)-t2(i-1))/(2*dx)
      uq=u2(i)-posq*(u2(i+1)-u2(i-1))/(2*dx)
      zq=z2(i)-posq*(z2(i+1)-z2(i-1))/(2*dx)
      ztq=zt2(i)-posq*(zt2(i+1)-zt2(i-1))/(2*dx)
      roq=ro2(i)-posq*(ro2(i+1)-ro2(i-1))/(2*dx)
      asq=as2(i)-posq*(as2(i+1)-as2(i-1))/(2*dx)
      htq=ht2(i)-posq*(ht2(i+1)-ht2(i-1))/(2*dx)
      wq=w2(i)-posq*(w2(i+1)-w2(i-1))/(2*dx)
      goto 6
c
    4 pq=p2(i)+posq*(p2(i+1)-p2(i-1))/(2*dx)
      tq=t2(i)+posq*(t2(i+1)-t2(i-1))/(2*dx)
      uq=u2(i)+posq*(u2(i+1)-u2(i-1))/(2*dx)
      zq=z2(i)+posq*(z2(i+1)-z2(i-1))/(2*dx)
      ztq=zt2(i)+posq*(zt2(i+1)-zt2(i-1))/(2*dx)
      roq=ro2(i)+posq*(ro2(i+1)-ro2(i-1))/(2*dx)
      asq=as2(i)+posq*(as2(i+1)-as2(i-1))/(2*dx)
      htq=ht2(i)+posq*(ht2(i+1)-ht2(i-1))/(2*dx)
      wq=w2(i)+posq*(w2(i+1)-w2(i-1))/(2*dx)
c
    6 pq=pq+posq*posq*(p2(i+1)+p2(i-1)-2*p2(i))/(2*dx*dx)
      tq=tq+posq*posq*(t2(i+1)+t2(i-1)-2*t2(i))/(2*dx*dx)
      uq=uq+posq*posq*(u2(i+1)+u2(i-1)-2*u2(i))/(2*dx*dx)
      zq=zq+posq*posq*(z2(i+1)+z2(i-1)-2*z2(i))/(2*dx*dx)
      ztq=ztq+posq*posq*(zt2(i+1)+zt2(i-1)-2*zt2(i))/(2*dx*dx)
      roq=roq+posq*posq*(ro2(i+1)+ro2(i-1)-2*ro2(i))/(2*dx*dx)
```

```
      htq=htq+posq*posq*(ht2(i+1)+ht2(i-1)-2*ht2(i))/(2*dx*dx)
      wq=wq+posq*posq*(w2(i+1)+w2(i-1)-2*w2(i))/(2*dx*dx)
C
      a(1)=-(1+tt2(i)*zzt2(i)/zz2(i))/rro2(i)
      a(1)=(a(1)-(1+tq*ztq/zq)/roq)/(2*cp)
      a(2)=1.0
      a(3)=0.0
      a(4)=(1/(ror*asr)+1/(rro2(i)*aas2(i)))/2
      a(5)=0.0
      a(6)=1.0
      a(7)=(-1/(ros*ass)-1/(rro2(i)*aas2(i)))/2
      a(8)=0.0
      a(9)=1.0
C
      b(1)=(htq+wq*uq)/roq+(hht2(i)+ww2(i)*uu2(i))/rro2(i)
      b(1)=b(1)*dt/(2*cp*ar)+a(1)*pq+tq
      b(2)=(1+tt2(i)*zzt2(i)/zz2(i))*(hht2(i)+ww2(i)*uu2(i))
      b(2)=b(2)*aas2(i)/(rro2(i)*tt2(i))
      b(2)=(b(2)+asr*(1+tr*ztr/zr)*(htr+wr*ur)/(ror*tr))/cp
      b(2)=(b(2)-(wr/ror+ww2(i)/rro2(i)))*dt/(2*ar)
      b(2)=b(2)-g*dt*dsin(th)+a(4)*pr+ur
      b(3)=(1+tt2(i)*zzt2(i)/zz2(i))*(hht2(i)+ww2(i)*uu2(i))
      b(3)=b(3)*aas2(i)/(rro2(i)*tt2(i))
      b(3)=(-b(3)-ass*(1+ts*zts/zs)*(hts+ws*us)/(ros*ts))/cp
      b(3)=(b(3)-(ws/ros+ww2(i)/rro2(i)))*dt/(2*ar)
      b(3)=b(3)-g*dt*dsin(th)+a(7)*ps+us
C
      call dminv(a,3,det,11,mm)
      if(det.ne.0.0) goto 12
      write(6,31) i
   31 format('pipe 2 (2nd order) i=',i3)
      stop "no inverse"
   12 pit=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tit=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uit=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      difp=dabs(pp2(i)-pit)/pit
      dift=dabs(tt2(i)-tit)/tit
      if(count.gt.200) goto 15
      if(difp.gt.0.01) goto 13
      if(dift.lt.0.01) goto 14
   13 pp2(i)=pit
      tt2(i)=tit
      uu2(i)=uit
      pdif=difp
      tdif=dift
      goto 10
C
   15 write(6,16)i
   16 format('sub4 - no iteration for i=',i4,' in pipe 2')
   14 pp2(i)=pit
      tt2(i)=tit
      uu2(i)=uit
      return
      end
```

```
C
      subroutine sub5(p1,t1,u1,dx,pp1,tt1,uu1,tc,pc,r,
     & ar,f,d,cp,pi,st,tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,
     & ht1,th,px,ux,tx,wx,zx,ztx,rox,asx,htx,py,uy,ty,
     & wy,zy,zty,roy,asy,hty,k)
C
C     This subroutine calculates p,t and u at internal
C     boundary points linking different grid sizes.
C
      implicit double precision (a-h,o-z)
      dimension p1(300),t1(300),u1(300),z1(300),
     & zt1(300),w1(300),as1(300),ht1(300),pp1(300),
     & tt1(300),uu1(300),zz1(300),zzp1(300),zzt1(300),
     & rro1(300),ww1(300),aas1(300),hht1(300),
     & px(6),tx(6),ux(6),zx(6),wx(6),ztx(6),
     & rox(6),asx(6),htx(6),py(6),uy(6),ty(6),zy(6),
     & wy(6),zty(6),roy(6),asy(6),hty(6),a(9),b(3),
     & ro1(300)
      integer l1(3),mm(3),count
C
C      First order approximation
C
      if(ux(k).eq.0.0) goto 20
      if(u1(i-1).eq.0.0) goto 20
      posq=dt*4/(1/ux(k)+1/u1(i-1))
      goto 22
   20 posq=dt*(ux(k)+u1(i-1))
   22 posr=dt*4/(1/(ux(k)+asx(k))+1/(u1(i-1)+as1(i-1)))
      poss=dt*2/(1/(asy(k)-uy(k))+1/(as1(i+1)-u1(i+1)))
C
      pq=posq/(2*dx)*p1(i-1)+(1-posq/(2*dx))*px(k)
      tq=posq/(2*dx)*t1(i-1)+(1-posq/(2*dx))*tx(k)
      uq=posq/(2*dx)*u1(i-1)+(1-posq/(2*dx))*ux(k)
      zq=posq/(2*dx)*z1(i-1)+(1-posq/(2*dx))*zx(k)
      ztq=posq/(2*dx)*zt1(i-1)+(1-posq/(2*dx))*ztx(k)
      roq=posq/(2*dx)*ro1(i-1)+(1-posq/(2*dx))*rox(k)
      asq=posq/(2*dx)*as1(i-1)+(1-posq/(2*dx))*asx(k)
      htq=posq/(2*dx)*ht1(i-1)+(1-posq/(2*dx))*htx(k)
      wq=posq/(2*dx)*w1(i-1)+(1-posq/(2*dx))*wx(k)
      pr=posr/(2*dx)*p1(i-1)+(1-posr/(2*dx))*px(k)
      tr=posr/(2*dx)*t1(i-1)+(1-posr/(2*dx))*tx(k)
      ur=posr/(2*dx)*u1(i-1)+(1-posr/(2*dx))*ux(k)
      zr=posr/(2*dx)*z1(i-1)+(1-posr/(2*dx))*zx(k)
      ztr=posr/(2*dx)*zt1(i-1)+(1-posr/(2*dx))*ztx(k)
      ror=posr/(2*dx)*ro1(i-1)+(1-posr/(2*dx))*rox(k)
      asr=posr/(2*dx)*as1(i-1)+(1-posr/(2*dx))*asx(k)
      htr=posr/(2*dx)*ht1(i-1)+(1-posr/(2*dx))*htx(k)
      wr=posr/(2*dx)*w1(i-1)+(1-posr/(2*dx))*wx(k)
      ps=poss/dx*p1(i+1)+(1-poss/dx)*py(k)
      ts=poss/dx*t1(i+1)+(1-poss/dx)*ty(k)
      us=poss/dx*u1(i+1)+(1-poss/dx)*uy(k)
      zs=poss/dx*z1(i+1)+(1-poss/dx)*zy(k)
      zts=poss/dx*zt1(i+1)+(1-poss/dx)*zty(k)
      ros=poss/dx*ro1(i+1)+(1-poss/dx)*roy(k)
      ass=poss/dx*as1(i+1)+(1-poss/dx)*asy(k)
      hts=poss/dx*ht1(i+1)+(1-poss/dx)*hty(k)
      ws=poss/dx*w1(i+1)+(1-poss/dx)*wy(k)
C
      x1=asr*2*dt*(1+tr*ztr/zr)/(ror*cp*tr*ar)
      x2=ass*dt*(1+ts*zts/zs)/(ros*cp*ts*ar)
      a(1)=-(1+tq*ztq/zq)/(roq*cp)-wq*uq*dt/(roq*cp*ar*pq)
      a(2)=1+wq*uq*dt/(roq*cp*ar*tq)
      a(3)=-wq*2*dt/(roq*cp*ar)
```

```
      a(5)=wr*ur*x1/(2*tr)-wr*dt/(ar*ror*tr)
      a(7)=-1/(ros*ass)+ws*us*x2/(2*ps)+ws*dt/(2*ar*ros*ps)
      a(8)=-ws*us*x2/(2*ts)-ws*dt/(2*ar*ros*ts)
      if(ur.eq.0.0) goto 2
      if(us.eq.0.0) goto 2
      a(6)=1-wr*x1+wr*2*dt/(ar*ror*ur)
      a(9)=1+ws*x2+ws*dt/(ar*ros*us)
      goto 1
    2 a(6)=1
      a(9)=1
c
    1 b(1)=htq*2*dt/(roq*cp*ar)+tq-pq*(1+tq*ztq/zq)/(roq*cp)
      b(2)=htr*x1+ur+pr/(ror*asr)-g*2*dt*dsin(th)
      b(3)=-hts*x2+us-ps/(ros*ass)-g*dt*dsin(th)
c
    5 call dminv(a,3,det,11,mm)
      if(det.ne.0.0)goto 9
      write(6,30) i
   30 format('pipe1 i=',i3)
      stop "no inverse"
    9 pp1(i)=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tt1(i)=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uu1(i)=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      count=0
      psave=pp1(i)
      tsave=tt1(i)
      usave=uu1(i)
      pdif=pp1(i)*1000
      tdif=tt1(i)*1000
   10 zz1(i)=9*tc/(128*tt1(i))-27*tc**3/(64*tt1(i)**3)
      zz1(i)=zz1(i)*pp1(i)/pc+1
      zzp1(i)=(zz1(i)-1)/pp1(i)
      zzt1(i)=81*tc**3/(64*tt1(i)**4)-9*tc/(128*tt1(i)*tt1(i))
      zzt1(i)=zzt1(i)*pp1(i)/pc
      rro1(i)=pp1(i)/(r*tt1(i)*zz1(i))
      ww1(i)=dabs(ar*rro1(i)*f*uu1(i)*uu1(i))/(2*d)
      aas1(i)=((1+zzt1(i)*tt1(i)/zz1(i))**2)*pp1(i)
      aas1(i)=aas1(i)/(rro1(i)*tt1(i)*cp)+zzp1(i)*pp1(i)/zz1(i)
      aas1(i)=(1-aas1(i))*rro1(i)/pp1(i)
      aas1(i)=1/dsqrt(dabs(aas1(i)))
      hht1(i)=pi*cp*st*d*rro1(i)*uu1(i)*(tw-tt1(i))
      count=count+1
c
c       Second order procedure
c
      if(uq.eq.0.0) goto 21
      if(uu1(i).eq.0.0) goto 21
      posq=4*dt/(1/uq+1/uu1(i))
      goto 23
   21 posq=dt*(uq+uu1(i))
   23 posr=4*dt/(1/(ur+asr)+1/(uu1(i)+aas1(i)))
      poss=2*dt/(1/(ass-us)+1/(aas1(i)-uu1(i)))
c
      pq=p1(i-1)+(2*dx-posq)*(px(k)-p1(i-2))/(4*dx)
      uq=u1(i-1)+(2*dx-posq)*(ux(k)-u1(i-2))/(4*dx)
      tq=t1(i-1)+(2*dx-posq)*(tx(k)-t1(i-2))/(4*dx)
      zq=z1(i-1)+(2*dx-posq)*(zx(k)-z1(i-2))/(4*dx)
      wq=w1(i-1)+(2*dx-posq)*(wx(k)-w1(i-2))/(4*dx)
      ztq=zt1(i-1)+(2*dx-posq)*(ztx(k)-zt1(i-2))/(4*dx)
      roq=ro1(i-1)+(2*dx-posq)*(rox(k)-ro1(i-2))/(4*dx)
      asq=as1(i-1)+(2*dx-posq)*(asx(k)-as1(i-2))/(4*dx)
      htq=ht1(i-1)+(2*dx-posq)*(htx(k)-ht1(i-2))/(4*dx)
      pr=p1(i-1)+(2*dx-posr)*(px(k)-p1(i-2))/(4*dx)
```

```
      tr=t1(i-1)+(2*dx-posr)*(tx(k)-t1(i-2))/(4*dx)
      zr=z1(i-1)+(2*dx-posr)*(zx(k)-z1(i-2))/(4*dx)
      wr=w1(i-1)+(2*dx-posr)*(wx(k)-w1(i-2))/(4*dx)
      ztr=zt1(i-1)+(2*dx-posr)*(ztx(k)-zt1(i-2))/(4*dx)
      ror=ro1(i-1)+(2*dx-posr)*(rox(k)-ro1(i-2))/(4*dx)
      asr=as1(i-1)+(2*dx-posr)*(asx(k)-as1(i-2))/(4*dx)
      htr=ht1(i-1)+(2*dx-posr)*(htx(k)-ht1(i-2))/(4*dx)
      ps=p1(i+1)-(dx-poss)*(p1(i+2)-py(k))/(2*dx)
      us=u1(i+1)-(dx-poss)*(u1(i+2)-uy(k))/(2*dx)
      ts=t1(i+1)-(dx-poss)*(t1(i+2)-ty(k))/(2*dx)
      zs=z1(i+1)-(dx-poss)*(z1(i+2)-zy(k))/(2*dx)
      ws=w1(i+1)-(dx-poss)*(w1(i+2)-wy(k))/(2*dx)
      zts=zt1(i+1)-(dx-poss)*(zt1(i+2)-zty(k))/(2*dx)
      ros=ro1(i+1)-(dx-poss)*(ro1(i+2)-roy(k))/(2*dx)
      ass=as1(i+1)-(dx-poss)*(as1(i+2)-asy(k))/(2*dx)
      hts=ht1(i+1)-(dx-poss)*(ht1(i+2)-hty(k))/(2*dx)
      pq=pq+(2*dx-posq)**2*(px(k)+p1(i-2)-2*p1(i-1))/(8*dx*dx)
      uq=uq+(2*dx-posq)**2*(ux(k)+u1(i-2)-2*u1(i-1))/(8*dx*dx)
      tq=tq+(2*dx-posq)**2*(tx(k)+t1(i-2)-2*t1(i-1))/(8*dx*dx)
      zq=zq+(2*dx-posq)**2*(zx(k)+z1(i-2)-2*z1(i-1))/(8*dx*dx)
      wq=wq+(2*dx-posq)**2*(wx(k)+w1(i-2)-2*w1(i-1))/(8*dx*dx)
      ztq=ztq+(2*dx-posq)**2*(ztx(k)+zt1(i-2)-2*zt1(i-1))/(8*dx*dx)
      roq=roq+(2*dx-posq)**2*(rox(k)+ro1(i-2)-2*ro1(i-1))/(8*dx*dx)
      asq=asq+(2*dx-posq)**2*(asx(k)+as1(i-2)-2*as1(i-1))/(8*dx*dx)
      htq=htq+(2*dx-posq)**2*(htx(k)+ht1(i-2)-2*ht1(i-1))/(8*dx*dx)
      pr=pr+(2*dx-posr)**2*(px(k)+p1(i-2)-2*p1(i-1))/(8*dx*dx)
      ur=ur+(2*dx-posr)**2*(ux(k)+u1(i-2)-2*u1(i-1))/(8*dx*dx)
      tr=tr+(2*dx-posr)**2*(tx(k)+t1(i-2)-2*t1(i-1))/(8*dx*dx)
      zr=zr+(2*dx-posr)**2*(zx(k)+z1(i-2)-2*z1(i-1))/(8*dx*dx)
      wr=wr+(2*dx-posr)**2*(wx(k)+w1(i-2)-2*w1(i-1))/(8*dx*dx)
      ztr=ztr+(2*dx-posr)**2*(ztx(k)+zt1(i-2)-2*zt1(i-1))/(8*dx*dx)
      ror=ror+(2*dx-posr)**2*(rox(k)+ro1(i-2)-2*ro1(i-1))/(8*dx*dx)
      asr=asr+(2*dx-posr)**2*(asx(k)+as1(i-2)-2*as1(i-1))/(8*dx*dx)
      htr=htr+(2*dx-posr)**2*(htx(k)+ht1(i-2)-2*ht1(i-1))/(8*dx*dx)
      ps=ps+(dx-poss)**2*(p1(i+2)+py(k)-2*p1(i+1))/(2*dx*dx)
      us=us+(dx-poss)**2*(u1(i+2)+uy(k)-2*u1(i+1))/(2*dx*dx)
      ts=ts+(dx-poss)**2*(t1(i+2)+ty(k)-2*t1(i+1))/(2*dx*dx)
      zs=zs+(dx-poss)**2*(z1(i+2)+zy(k)-2*z1(i+1))/(2*dx*dx)
      ws=ws+(dx-poss)**2*(w1(i+2)+wy(k)-2*w1(i+1))/(2*dx*dx)
      zts=zts+(dx-poss)**2*(zt1(i+2)+zty(k)-2*zt1(i+1))/(2*dx*dx)
      ros=ros+(dx-poss)**2*(ro1(i+2)+roy(k)-2*ro1(i+1))/(2*dx*dx)
      ass=ass+(dx-poss)**2*(as1(i+2)+asy(k)-2*as1(i+1))/(2*dx*dx)
      hts=hts+(dx-poss)**2*(ht1(i+2)+hty(k)-2*ht1(i+1))/(2*dx*dx)
C
      a(1)=-(1+tt1(i)*zzt1(i)/zz1(i))/rro1(i)
      a(1)=(a(1)-(1+tq*ztq/zq)/roq)/(2*cp)
      a(2)=1.0
      a(3)=0.0
      a(4)=(1/(ror*asr)+1/(rro1(i)*aas1(i)))/2
      a(5)=0.0
      a(6)=1.0
      a(7)=(-1/(ros*ass)-1/(rro1(i)*aas1(i)))/2
      a(8)=0.0
      a(9)=1.0
C
      b(1)=(htq+wq*uq)/roq+(hht1(i)+ww1(i)*uu1(i))/rro1(i)
      b(1)=b(1)*dt/(cp*ar)+a(1)*pq+tq
      b(2)=(1+tt1(i)*zzt1(i)/zz1(i))*(hht1(i)+ww1(i)*uu1(i))
      b(2)=b(2)*aas1(i)/(rro1(i)*tt1(i))
      b(2)=(b(2)+asr*(1+tr*ztr/zr)*(htr+wr*ur)/(ror*tr))/cp
      b(2)=(b(2)-(wr/ror+ww1(i)/rro1(i)))*dt/ar
      b(2)=b(2)-g*2*dt*dsin(th)+a(4)*pr+ur
      b(3)=(1+tt1(i)*zzt1(i)/zz1(i))*(hht1(i)+ww1(i)*uu1(i))
```

```
      b(3)=(b(3)-ass*(1+ts*zts/zs)*(hts+ws*us)/(ros*ts))/cp
      b(3)=(b(3)-(ws/ros+ww1(i)/rro1(i)))*dt/(2*ar)
      b(3)=b(3)-g*dt*dsin(th)+a(7)*ps+us
C
      call dminv(a,3,det,ll,mm)
      if(det.ne.0.0) goto 12
      write(6,31) i
   31 format('pipe 1 (2nd order) i=',i3)
      stop "no inverse"
   12 pit=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tit=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uit=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      difp=dabs(pp1(i)-pit)/pit
      dift=dabs(tt1(i)-tit)/tit
      if (count.gt.200) goto 15
      if(difp.gt.0.01) goto 13
      if(dift.lt.0.01) goto 14
   13 pp1(i)=pit
      tt1(i)=tit
      uu1(i)=uit
      pdif=difp
      tdif=dift
      goto 10
C
   15 write(6,16)i
   16 format('sub5 - no iteration for i=',i4,'in pipe 1 (sub5)')
   14 pp1(i)=pit
      tt1(i)=tit
      uu1(i)=uit
      return
      end
```

```
c
      subroutine sub6(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,
     & ar,f,d,cp,pi,st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,
     & ht2,th,pz,tz,uz,zz,ztz,roz,asz,htz,wz,k)
c
c       This subroutine calculates p,t and u at internal
c       boundary points linking different grid sizes in pipe 2.
c
      implicit double precision (a-h,o-z)
      dimension p2(300),t2(300),u2(300),z2(300),
     & zt2(300),w2(300),as2(300),ht2(300),pp2(300),
     & tt2(300),uu2(300),zz2(300),zzp2(300),zzt2(300),
     & rro2(300),ww2(300),aas2(300),hht2(300),
     & a(9),b(3),ro2(300),pz(6),tz(6),uz(6),wz(6),
     & zz(6),ztz(6),roz(6),asz(6),htz(6)
      integer ll(3),mm(3),count
c
c       First order approximation
c
      posr=dt*2/(1/(u2(i)+as2(i))+1/(u2(i-1)+as2(i-1)))
      poss=dt*4/(1/(as2(i+1)-u2(i+1))+1/(asz(k)-uz(k)))
      pr=posr/dx*p2(i-1)+(1-posr/dx)*p2(i)
      tr=posr/dx*t2(i-1)+(1-posr/dx)*t2(i)
      ur=posr/dx*u2(i-1)+(1-posr/dx)*u2(i)
      zr=posr/dx*z2(i-1)+(1-posr/dx)*z2(i)
      ztr=posr/dx*zt2(i-1)+(1-posr/dx)*zt2(i)
      ror=posr/dx*ro2(i-1)+(1-posr/dx)*ro2(i)
      asr=posr/dx*as2(i-1)+(1-posr/dx)*as2(i)
      htr=posr/dx*ht2(i-1)+(1-posr/dx)*ht2(i)
      wr=posr/dx*w2(i-1)+(1-posr/dx)*w2(i)
      ps=poss/(2*dx)*p2(i+1)+(1-poss/(2*dx))*pz(k)
      ts=poss/(2*dx)*t2(i+1)+(1-poss/(2*dx))*tz(k)
      us=poss/(2*dx)*u2(i+1)+(1-poss/(2*dx))*uz(k)
      zs=poss/(2*dx)*z2(i+1)+(1-poss/(2*dx))*zz(k)
      zts=poss/(2*dx)*zt2(i+1)+(1-poss/(2*dx))*ztz(k)
      ros=poss/(2*dx)*ro2(i+1)+(1-poss/(2*dx))*roz(k)
      ass=poss/(2*dx)*as2(i+1)+(1-poss/(2*dx))*asz(k)
      hts=poss/(2*dx)*ht2(i+1)+(1-poss/(2*dx))*htz(k)
      ws=poss/(2*dx)*w2(i+1)+(1-poss/(2*dx))*wz(k)
      x1=asr*dt*(1+tr*ztr/zr)/(ror*cp*tr*ar)
      x2=ass*2*dt*(1+ts*zts/zs)/(ros*cp*ts*ar)
      a(4)=1/(ror*asr)-wr*ur*x1/(2*pr)+wr*dt/(2*ar*ror*pr)
      a(5)=wr*ur*x1/(2*tr)-wr*dt/(2*ar*ror*tr)
      a(7)=-1/(ros*ass)+ws*us*x2/(2*ps)+ws*dt/(ar*ros*ps)
      a(8)=-ws*us*x2/(2*ts)-ws*dt/(ar*ros*ts)
      if(ur.eq.0.0) goto 30
      if(us.eq.0.0) goto 30
      a(6)=1-wr*x1+wr*dt/(ar*ror*ur)
      a(9)=1+ws*x2+ws*2*dt/(ar*ros*us)
      goto 31
   30 a(6)=1
      a(9)=1
   31 b(2)=htr*x1+ur+pr/(ror*asr)-g*dt*dsin(th)
      b(3)=-hts*x2+us-ps/(ros*ass)-g*dt*dsin(th)
c
      if(u2(i).le.0.0) goto 1
      if(u2(i-1).eq.0.0) goto 2
      posq=dt*2/(1/u2(i)+1/u2(i-1))
      goto 5
    2 posq=dt*u2(i)/2
    5 pq=posq/dx*p2(i-1)+(1-posq/dx)*p2(i)
      tq=posq/dx*t2(i-1)+(1-posq/dx)*t2(i)
      uq=posq/dx*u2(i-1)+(1-posq/dx)*u2(i)
```

```
      ztq=posq/dx*zt2(i-1)+(1-posq/dx)*zt2(i)
      roq=posq/dx*ro2(i-1)+(1-posq/dx)*ro2(i)
      asq=posq/dx*as2(i-1)+(1-posq/dx)*as2(i)
      htq=posq/dx*ht2(i-1)+(1-posq/dx)*ht2(i)
      wq=posq/dx*w2(i-1)+(1-posq/dx)*w2(i)
      a(1)=-(1+tq*ztq/zq)/(roq*cp)-wq*uq*dt/(2*roq*cp*ar*pq)
      a(2)=1+wq*uq*dt/(2*roq*cp*ar*tq)
      a(3)=-wq*dt/(roq*cp*ar)
      b(1)=htq*dt/(roq*cp*ar)+tq-pq*(1+tq*ztq/zq)/(roq*cp)
      goto 3
c
    1 if(u2(i).eq.0.0) goto 7
      if(u2(i+1).eq.0.0) goto 7
      posq=dabs(dt*4/(1/u2(i)+1/u2(i+1)))
      goto 8
    7 posq=dabs(dt*(u2(i)+u2(i+1)))
    8 pq=posq/(2*dx)*p2(i+1)+(1-posq/(2*dx))*pz(k)
      tq=posq/(2*dx)*t2(i+1)+(1-posq/(2*dx))*tz(k)
      uq=posq/(2*dx)*u2(i+1)+(1-posq/(2*dx))*uz(k)
      zq=posq/(2*dx)*z2(i+1)+(1-posq/(2*dx))*zz(k)
      ztq=posq/(2*dx)*zt2(i+1)+(1-posq/(2*dx))*ztz(k)
      roq=posq/(2*dx)*ro2(i+1)+(1-posq/(2*dx))*roz(k)
      asq=posq/(2*dx)*as2(i+1)+(1-posq/(2*dx))*asz(k)
      htq=posq/(2*dx)*ht2(i+1)+(1-posq/(2*dx))*htz(k)
      wq=posq/(2*dx)*w2(i+1)+(1-posq/(2*dx))*wz(k)
c
      a(1)=-(1+tq*ztq/zq)/(roq*cp)-wq*uq*dt/(roq*cp*ar*pq)
      a(2)=1+wq*uq*dt/(roq*cp*ar*tq)
      a(3)=-wq*2*dt/(roq*cp*ar)
      b(1)=htq*2*dt/(roq*cp*ar)+tq-pq*(1+tq*ztq/zq)/(roq*cp)
c
    3 call dminv(a,3,det,11,mm)
      if(det.ne.0.0)goto 9
      write(6,32) i
   32 format('pipe 2 i=',i3)
      stop "no inverse"
    9 pp2(i)=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tt2(i)=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uu2(i)=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      count=0
      psave=pp2(i)
      usave=uu2(i)
      tsave=tt2(i)
      pdif=pp2(i)*1000
      tdif=tt2(i)*1000
c
   10 zz2(i)=9*tc/(128*tt2(i))-27*tc**3/(64*tt2(i)**3)
      zz2(i)=zz2(i)*pp2(i)/pc+1
      zzp2(i)=(zz2(i)-1)/pp2(i)
      zzt2(i)=81*tc**3/(64*tt2(i)**4)-9*tc/(128*tt2(i)*tt2(i))
      zzt2(i)=zzt2(i)*pp2(i)/pc
      rro2(i)=pp2(i)/(r*tt2(i)*zz2(i))
      ww2(i)=dabs(ar*rro2(i)*f*uu2(i)*uu2(i))/(2*d)
      aas2(i)=((1+zzt2(i)*tt2(i)/zz2(i))**2)*pp2(i)
      aas2(i)=aas2(i)/(rro2(i)*tt2(i)*cp)+zzp2(i)*pp2(i)/zz2(i)
      aas2(i)=(1-aas2(i))*rro2(i)/pp2(i)
      aas2(i)=1/dsqrt(dabs(aas2(i)))
      hht2(i)=pi*cp*st*d*rro2(i)*uu2(i)*(tw-tt2(i))
      count=count+1
c
c      Second order procedure
c
      posr=2*dt/(1/(ur+asr)+1/(uu2(i)+aas2(i)))
```

```
      pr=p2(i-1)+(dx-posr)*(p2(i)-p2(i-2))/(2*dx)
      ur=u2(i-1)+(dx-posr)*(u2(i)-u2(i-2))/(2*dx)
      tr=t2(i-1)+(dx-posr)*(t2(i)-t2(i-2))/(2*dx)
      zr=z2(i-1)+(dx-posr)*(z2(i)-z2(i-2))/(2*dx)
      wr=w2(i-1)+(dx-posr)*(w2(i)-w2(i-2))/(2*dx)
      ztr=zt2(i-1)+(dx-posr)*(zt2(i)-zt2(i-2))/(2*dx)
      ror=ro2(i-1)+(dx-posr)*(ro2(i)-ro2(i-2))/(2*dx)
      asr=as2(i-1)+(dx-posr)*(as2(i)-as2(i-2))/(2*dx)
      htr=ht2(i-1)+(dx-posr)*(ht2(i)-ht2(i-2))/(2*dx)
      ps=p2(i+1)-(2*dx-poss)*(p2(i+2)-pz(k))/(4*dx)
      us=u2(i+1)-(2*dx-poss)*(u2(i+2)-uz(k))/(4*dx)
      ts=t2(i+1)-(2*dx-poss)*(t2(i+2)-tz(k))/(4*dx)
      zs=z2(i+1)-(2*dx-poss)*(z2(i+2)-zz(k))/(4*dx)
      ws=w2(i+1)-(2*dx-poss)*(w2(i+2)-wz(k))/(4*dx)
      zts=zt2(i+1)-(2*dx-poss)*(zt2(i+2)-ztz(k))/(4*dx)
      ros=ro2(i+1)-(2*dx-poss)*(ro2(i+2)-roz(k))/(4*dx)
      ass=as2(i+1)-(2*dx-poss)*(as2(i+2)-asz(k))/(4*dx)
      hts=ht2(i+1)-(2*dx-poss)*(ht2(i+2)-htz(k))/(4*dx)
      pr=pr+(dx-posr)**2*(p2(i)+p2(i-2)-2*p2(i-1))/(2*dx*dx)
      ur=ur+(dx-posr)**2*(u2(i)+u2(i-2)-2*u2(i-1))/(2*dx*dx)
      tr=tr+(dx-posr)**2*(t2(i)+t2(i-2)-2*t2(i-1))/(2*dx*dx)
      zr=zr+(dx-posr)**2*(z2(i)+z2(i-2)-2*z2(i-1))/(2*dx*dx)
      wr=wr+(dx-posr)**2*(w2(i)+w2(i-2)-2*w2(i-1))/(2*dx*dx)
      ztr=ztr+(dx-posr)**2*(zt2(i)+zt2(i-2)-2*zt2(i-1))/(2*dx*dx)
      ror=ror+(dx-posr)**2*(ro2(i)+ro2(i-2)-2*ro2(i-1))/(2*dx*dx)
      asr=asr+(dx-posr)**2*(as2(i)+as2(i-2)-2*as2(i-1))/(2*dx*dx)
      htr=htr+(dx-posr)**2*(ht2(i)+ht2(i-2)-2*ht2(i-1))/(2*dx*dx)
      ps=ps+(2*dx-poss)**2*(p2(i+2)+pz(k)-2*p2(i+1))/(8*dx*dx)
      us=us+(2*dx-poss)**2*(u2(i+2)+uz(k)-2*u2(i+1))/(8*dx*dx)
      ts=ts+(2*dx-poss)**2*(t2(i+2)+tz(k)-2*t2(i+1))/(8*dx*dx)
      zs=zs+(2*dx-poss)**2*(z2(i+2)+zz(k)-2*z2(i+1))/(8*dx*dx)
      ws=ws+(2*dx-poss)**2*(w2(i+2)+wz(k)-2*w2(i+1))/(8*dx*dx)
      zts=zts+(2*dx-poss)**2*(zt2(i+2)+ztz(k)-2*zt2(i+1))/(8*dx*dx)
      ros=ros+(2*dx-poss)**2*(ro2(i+2)+roz(k)-2*ro2(i+1))/(8*dx*dx)
      ass=ass+(2*dx-poss)**2*(as2(i+2)+asz(k)-2*as2(i+1))/(8*dx*dx)
      hts=hts+(2*dx-poss)**2*(ht2(i+2)+htz(k)-2*ht2(i+1))/(8*dx*dx)
c
      if((uu2(i)+uq).lt.0.0) goto 4
      if(uq.eq.0.0) goto 11
      if(uu2(i).eq.0.0) goto 11
      posq=dabs(dt*2/(1/uu2(i)+1/uq))
      goto 20
   11 posq=dabs(dt*(uq+uu2(i))/2)
   20 pq=p2(i-1)+(dx-posq)*(p2(i)-p2(i-2))/(2*dx)
      uq=u2(i-1)+(dx-posq)*(u2(i)-u2(i-2))/(2*dx)
      tq=t2(i-1)+(dx-posq)*(t2(i)-t2(i-2))/(2*dx)
      zq=z2(i-1)+(dx-posq)*(z2(i)-z2(i-2))/(2*dx)
      wq=w2(i-1)+(dx-posq)*(w2(i)-w2(i-2))/(2*dx)
      ztq=zt2(i-1)+(dx-posq)*(zt2(i)-zt2(i-2))/(2*dx)
      roq=ro2(i-1)+(dx-posq)*(ro2(i)-ro2(i-2))/(2*dx)
      asq=as2(i-1)+(dx-posq)*(as2(i)-as2(i-2))/(2*dx)
      htq=ht2(i-1)+(dx-posq)*(ht2(i)-ht2(i-2))/(2*dx)
      pq=pq+(dx-posq)**2*(p2(i)+p2(i-2)-2*p2(i-1))/(2*dx*dx)
      uq=uq+(dx-posq)**2*(u2(i)+u2(i-2)-2*u2(i-1))/(2*dx*dx)
      tq=tq+(dx-posq)**2*(t2(i)+t2(i-2)-2*t2(i-1))/(2*dx*dx)
      zq=zq+(dx-posq)**2*(z2(i)+z2(i-2)-2*z2(i-1))/(2*dx*dx)
      wq=wq+(dx-posq)**2*(w2(i)+w2(i-2)-2*w2(i-1))/(2*dx*dx)
      ztq=ztq+(dx-posq)**2*(zt2(i)+zt2(i-2)-2*zt2(i-1))/(2*dx*dx)
      roq=roq+(dx-posq)**2*(ro2(i)+ro2(i-2)-2*ro2(i-1))/(2*dx*dx)
      asq=asq+(dx-posq)**2*(as2(i)+as2(i-2)-2*as2(i-1))/(2*dx*dx)
      htq=htq+(dx-posq)**2*(ht2(i)+ht2(i-2)-2*ht2(i-1))/(2*dx*dx)
      goto 6
c
```

```
      if(uu2(i).eq.0.0) goto 21
      posq=dabs(dt*4/(1/uq+1/uu2(i)))
      goto 19
   21 posq=dabs(dt*(uq+uu2(i)))
   19 pq=p2(i+1)-(2*dx-posq)*(p2(i+2)-pz(k))/(4*dx)
      uq=u2(i+1)-(2*dx-posq)*(u2(i+2)-uz(k))/(4*dx)
      tq=t2(i+1)-(2*dx-posq)*(t2(i+2)-tz(k))/(4*dx)
      zq=z2(i+1)-(2*dx-posq)*(z2(i+2)-zz(k))/(4*dx)
      wq=w2(i+1)-(2*dx-posq)*(w2(i+2)-wz(k))/(4*dx)
      ztq=zt2(i+1)-(2*dx-posq)*(zt2(i+2)-ztz(k))/(4*dx)
      roq=ro2(i+1)-(2*dx-posq)*(ro2(i+2)-roz(k))/(4*dx)
      asq=as2(i+1)-(2*dx-posq)*(as2(i+2)-asz(k))/(4*dx)
      htq=ht2(i+1)-(2*dx-posq)*(ht2(i+2)-htz(k))/(4*dx)
      pq=pq+(2*dx-posq)**2*(p2(i+2)+pz(k)-2*p2(i+1))/(8*dx*dx)
      uq=uq+(2*dx-posq)**2*(u2(i+2)+uz(k)-2*u2(i+1))/(8*dx*dx)
      tq=tq+(2*dx-posq)**2*(t2(i+2)+tz(k)-2*t2(i+1))/(8*dx*dx)
      zq=zq+(2*dx-posq)**2*(z2(i+2)+zz(k)-2*z2(i+1))/(8*dx*dx)
      wq=wq+(2*dx-posq)**2*(w2(i+2)+wz(k)-2*w2(i+1))/(8*dx*dx)
      ztq=ztq+(2*dx-posq)**2*(zt2(i+2)+ztz(k)-2*zt2(i+1))/(8*dx*dx)
      roq=roq+(2*dx-posq)**2*(ro2(i+2)+roz(k)-2*ro2(i+1))/(8*dx*dx)
      asq=asq+(2*dx-posq)**2*(as2(i+2)+asz(k)-2*as2(i+1))/(8*dx*dx)
      htq=htq+(2*dx-posq)**2*(ht2(i+2)+htz(k)-2*ht2(i+1))/(8*dx*dx)
c
    6 a(1)=-(1+tt2(i)*zzt2(i)/zz2(i))/rro2(i)
      a(1)=(a(1)-(1+tq*ztq/zq)/roq)/(2*cp)
      a(2)=1.0
      a(3)=0.0
      a(4)=(1/(ror*asr)+1/(rro2(i)*aas2(i)))/2
      a(5)=0.0
      a(6)=1.0
      a(7)=(-1/(ros*ass)-1/(rro2(i)*aas2(i)))/2
      a(8)=0.0
      a(9)=1.0
c
      if(uu2(i).lt.0.0) goto 17
      b(1)=(htq+wq*uq)/roq+(hht2(i)+ww2(i)*uu2(i))/rro2(i)
      b(1)=b(1)*dt/(2*cp*ar)+a(1)*pq+tq
      goto 18
c
   17 b(1)=(htq+wq*uq)/roq+(hht2(i)+ww2(i)*uu2(i))/rro2(i)
      b(1)=b(1)*dt/(cp*ar)+a(1)*pq+tq
   18 b(2)=(1+tt2(i)*zzt2(i)/zz2(i))*(hht2(i)+ww2(i)*uu2(i))
      b(2)=b(2)*aas2(i)/(rro2(i)*tt2(i))
      b(2)=(b(2)+asr*(1+tr*ztr/zr)*(htr+wr*ur)/(ror*tr))/cp
      b(2)=(b(2)-(wr/ror+ww2(i)/rro2(i)))*dt/(2*ar)
      b(2)=b(2)-g*dt*dsin(th)+a(4)*pr+ur
      b(3)=(1+tt2(i)*zzt2(i)/zz2(i))*(hht2(i)+ww2(i)*uu2(i))
      b(3)=b(3)*aas2(i)/(rro2(i)*tt2(i))
      b(3)=(-b(3)-ass*(1+ts*zts/zs)*(hts+ws*us)/(ros*ts))/cp
      b(3)=(b(3)-(ws/ros+ww2(i)/rro2(i)))*dt/ar
      b(3)=b(3)-g*2*dt*dsin(th)+a(7)*ps+us
c
      call dminv(a,3,det,11,mm)
      if(det.ne.0.0) goto 12
      write(6,33) i
   33 format('pipe 2 (2nd order) i=',i3)
      stop "no inverse"
   12 pit=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tit=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uit=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      difp=dabs(pp2(i)-pit)/pit
      dift=dabs(tt2(i)-tit)/tit
      if(count.gt.200) goto 15
```

```
      if(dift.lt.0.01) goto 14
   13 pp2(i)=pit
      tt2(i)=tit
      uu2(i)=uit
      pdif=difp
      tdif=dift
      goto 10
c
   15 write(6,16)i
   16 format('sub6 - no iteration for i=',i4,' in pipe 2')
   14 pp2(i)=pit
      tt2(i)=tit
      uu2(i)=uit
      return
      end
```

```
c
      subroutine break1 (p1,t1,u1,p2,t2,u2,pp1,tt1,uu1,
     & z1,zt1,ro1,as1,ht1,w1,dx,tc,pc,r,ar,f,d,cp,pi,
     & st,tw,g,dt,i,th,z2,zt2,ro2,as2,ht2,w2)
c
c       this subroutine calculates steady flow conditions
c       at the break point prior to failure (pipe 1).
c
      implicit double precision (a-h,o-z)
      dimension p1(300),t1(300),u1(300),z1(300),
     & zt1(300),w1(300),as1(300),ht1(300),
     & pp1(300),tt1(300),uu1(300),p2(300),t2(300),
     & u2(300),z2(300),zt2(300),ro2(300),w2(300),
     & as2(300),ht2(300),zz1(300),zzp1(300),
     & zzt1(300),rro1(300),ww1(300),aas1(300),
     & hht1(300),a(9),b(3),ro1(300)
      integer ll(3),mm(3),count
c
c       first order approximation
c
      if(u1(i).eq.0.0) goto 1
      if(u1(i-1).eq.0.0) goto 1
      posq=dt*2/(1/(u1(i-1))+1/u1(i))
      goto 2
    1 posq=dt*(u1(i)+u1(i-1))/2
    2 posr=dt*2/(1/(u1(i-1)+as1(i-1))+1/(u1(i)+as1(i)))
      poss=dt*2/(1/(as2(1)-u2(1))+1/(as2(2)-u2(2)))
c
      pq=posq/dx*p1(i-1)+(1-posq/dx)*p1(i)
      tq=posq/dx*t1(i-1)+(1-posq/dx)*t1(i)
      uq=posq/dx*u1(i-1)+(1-posq/dx)*u1(i)
      zq=posq/dx*z1(i-1)+(1-posq/dx)*z1(i)
      ztq=posq/dx*zt1(i-1)+(1-posq/dx)*zt1(i)
      roq=posq/dx*ro1(i-1)+(1-posq/dx)*ro1(i)
      asq=posq/dx*as1(i-1)+(1-posq/dx)*as1(i)
      htq=posq/dx*ht1(i-1)+(1-posq/dx)*ht1(i)
      wq=posq/dx*w1(i-1)+(1-posq/dx)*w1(i)
      pr=posr/dx*p1(i-1)+(1-posr/dx)*p1(i)
      tr=posr/dx*t1(i-1)+(1-posr/dx)*t1(i)
      ur=posr/dx*u1(i-1)+(1-posr/dx)*u1(i)
      zr=posr/dx*z1(i-1)+(1-posr/dx)*z1(i)
      ztr=posr/dx*zt1(i-1)+(1-posr/dx)*zt1(i)
      ror=posr/dx*ro1(i-1)+(1-posr/dx)*ro1(i)
      asr=posr/dx*as1(i-1)+(1-posr/dx)*as1(i)
      htr=posr/dx*ht1(i-1)+(1-posr/dx)*ht1(i)
      wr=posr/dx*w1(i-1)+(1-posr/dx)*w1(i)
      ps=poss/dx*p2(2)+(1-poss/dx)*p2(1)
      ts=poss/dx*t2(2)+(1-poss/dx)*t2(1)
      us=poss/dx*u2(2)+(1-poss/dx)*u2(1)
      zs=poss/dx*z2(2)+(1-poss/dx)*z2(1)
      zts=poss/dx*zt2(2)+(1-poss/dx)*zt2(1)
      ros=poss/dx*ro2(2)+(1-poss/dx)*ro2(1)
      ass=poss/dx*as2(2)+(1-poss/dx)*as2(1)
      hts=poss/dx*ht2(2)+(1-poss/dx)*ht2(1)
      ws=poss/dx*w2(2)+(1-poss/dx)*w2(1)
c
      x1=asr*dt*(1+tr*ztr/zr)/(ror*cp*tr*ar)
      x2=ass*dt*(1+ts*zts/zs)/(ros*cp*ts*ar)
      a(1)=-(1+tq*ztq/zq)/(roq*cp)-wq*uq*dt/(2*roq*cp*ar*pq)
      a(2)=1+wq*uq*dt/(2*roq*cp*ar*tq)
      a(3)=-wq*dt/(roq*cp*ar)
      a(4)=1/(ror*asr)-wr*ur*x1/(2*pr)+wr*dt/(2*ar*ror*pr)
      a(5)=wr*ur*x1/(2*tr)-wr*dt/(2*ar*ror*tr)
```

```
      a(7)=-1/(1+us*ass)+ws*us*x2/(2*ps)+ws*ut/(2*ar*ros*ps)
      a(8)=-ws*us*x2/(2*ts)-ws*dt/(2*ar*ros*ts)
      if(ur.eq.0.0) goto 20
      if(us.eq.0.0) goto 20
      a(6)=1-wr*x1+wr*dt/(ar*ror*ur)
      a(9)=1+ws*x2+ws*dt/(ar*ros*us)
      goto 21
   20 a(6)=1.0
      a(9)=1.0
c
   21 b(1)=htq*dt/(roq*cp*ar)+tq-pq*(1+tq*ztq/zq)/(roq*cp)
      b(2)=htr*x1+ur+pr/(ror*asr)-g*dt*dsin(th)
      b(3)=-hts*x2+us-ps/(ros*ass)-g*dt*dsin(th)
c
    5 call dminv(a,3,det,11,mm)
      if(det.ne.0.0)goto 9
      stop "no inverse"
    9 pp1(i)=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tt1(i)=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uu1(i)=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      count=0
      psave=pp1(i)
      usave=uu1(i)
      tsave=tt1(i)
      pdif=pp1(i)*1000
      udif=uu1(i)*1000
      tdif=tt1(i)*1000
   10 zz1(i)=9*tc/(128*tt1(i))-27*tc**3/(64*tt1(i)**3)
      zz1(i)=zz1(i)*pp1(i)/pc+1
      zzp1(i)=(zz1(i)-1)/pp1(i)
      zzt1(i)=81*tc**3/(64*tt1(i)**4)-9*tc/(128*tt1(i)*tt1(i))
      zzt1(i)=zzt1(i)*pp1(i)/pc
      rro1(i)=pp1(i)/(r*tt1(i)*zz1(i))
      ww1(i)=dabs(ar*rro1(i)*f*uu1(i)*uu1(i))/(2*d)
      aas1(i)=((1+zzt1(i)*tt1(i)/zz1(i))**2)*pp1(i)
      aas1(i)=aas1(i)/(rro1(i)*tt1(i)*cp)+zzp1(i)*pp1(i)/zz1(i)
      aas1(i)=(1-aas1(i))*rro1(i)/pp1(i)
      aas1(i)=1/dsqrt(dabs(aas1(i)))
      hht1(i)=pi*cp*st*d*rro1(i)*uu1(i)*(tw-tt1(i))
      count=count+1
c
c       second order procedure
c
      if(uq.eq.0.0) goto 3
      if(uu1(i).eq.0.0) goto 3
      posq=2*dt/(1/uq+1/uu1(i))
      goto 4
    3 posq=dt*(uq+uu1(i))/2
    4 posr=2*dt/(1/(ur+asr)+1/(uu1(i)+aas1(i)))
      poss=2*dt/(1/(ass-us)+1/(aas1(i)-uu1(i)))
c
      pq=p1(i)-posq*(p2(2)-p1(i-1))/(2*dx)
      tq=t1(i)-posq*(t2(2)-t1(i-1))/(2*dx)
      uq=u1(i)-posq*(u2(2)-u1(i-1))/(2*dx)
      zq=z1(i)-posq*(z2(2)-z1(i-1))/(2*dx)
      ztq=zt1(i)-posq*(zt2(2)-zt1(i-1))/(2*dx)
      roq=ro1(i)-posq*(ro2(2)-ro1(i-1))/(2*dx)
      asq=as1(i)-posq*(as2(2)-as1(i-1))/(2*dx)
      htq=ht1(i)-posq*(ht2(2)-ht1(i-1))/(2*dx)
      wq=w1(i)-posq*(w2(2)-w1(i-1))/(2*dx)
      pr=p1(i)-posr*(p2(2)-p1(i-1))/(2*dx)
      tr=t1(i)-posr*(t2(2)-t1(i-1))/(2*dx)
      ur=u1(i)-posr*(u2(2)-u1(i-1))/(2*dx)
      zr=z1(i)-posr*(z2(2)-z1(i-1))/(2*dx)
```

```
      ror=ro1(i)-posr*(ro2(2)-ro1(i-1))/(2*dx)
      asr=as1(i)-posr*(as2(2)-as1(i-1))/(2*dx)
      htr=ht1(i)-posr*(ht2(2)-ht1(i-1))/(2*dx)
      wr=w1(i)-posr*(w2(2)-w1(i-1))/(2*dx)
      ps=p1(i)+poss*(p2(2)-p1(i-1))/(2*dx)
      ts=t1(i)+poss*(t2(2)-t1(i-1))/(2*dx)
      us=u1(i)+poss*(u2(2)-u1(i-1))/(2*dx)
      zs=z1(i)+poss*(z2(2)-z1(i-1))/(2*dx)
      zts=zt1(i)+poss*(zt2(2)-zt1(i-1))/(2*dx)
      ros=ro1(i)+poss*(ro2(2)-ro1(i-1))/(2*dx)
      ass=as1(i)+poss*(as2(2)-as1(i-1))/(2*dx)
      hts=ht1(i)+poss*(ht2(2)-ht1(i-1))/(2*dx)
      ws=w1(i)+poss*(w2(2)-w1(i-1))/(2*dx)
      pq=pq+posq*posq*(p2(2)+p1(i-1)-2*p1(i))/(2*dx*dx)
      tq=tq+posq*posq*(t2(2)+t1(i-1)-2*t1(i))/(2*dx*dx)
      uq=uq+posq*posq*(u2(2)+u1(i-1)-2*u1(i))/(2*dx*dx)
      zq=zq+posq*posq*(z2(2)+z1(i-1)-2*z1(i))/(2*dx*dx)
      ztq=ztq+posq*posq*(zt2(2)+zt1(i-1)-2*zt1(i))/(2*dx*dx)
      roq=roq+posq*posq*(ro2(2)+ro1(i-1)-2*ro1(i))/(2*dx*dx)
      asq=asq+posq*posq*(as2(2)+as1(i-1)-2*as1(i))/(2*dx*dx)
      htq=htq+posq*posq*(ht2(2)+ht1(i-1)-2*ht1(i))/(2*dx*dx)
      wq=wq+posq*posq*(w2(2)+w1(i-1)-2*w1(i))/(2*dx*dx)
      pr=pr+posr*posr*(p2(2)+p1(i-1)-2*p1(i))/(2*dx*dx)
      tr=tr+posr*posr*(t2(2)+t1(i-1)-2*t1(i))/(2*dx*dx)
      ur=ur+posr*posr*(u2(2)+u1(i-1)-2*u1(i))/(2*dx*dx)
      zr=zr+posr*posr*(z2(2)+z1(i-1)-2*z1(i))/(2*dx*dx)
      ztr=ztr+posr*posr*(zt2(2)+zt1(i-1)-2*zt1(i))/(2*dx*dx)
      ror=ror+posr*posr*(ro2(2)+ro1(i-1)-2*ro1(i))/(2*dx*dx)
      asr=asr+posr*posr*(as2(2)+as1(i-1)-2*as1(i))/(2*dx*dx)
      htr=htr+posr*posr*(ht2(2)+ht1(i-1)-2*ht1(i))/(2*dx*dx)
      wr=wr+posr*posr*(w2(2)+w1(i-1)-2*w1(i))/(2*dx*dx)
      ps=ps+poss*poss*(p2(2)+p1(i-1)-2*p1(i))/(2*dx*dx)
      ts=ts+poss*poss*(t2(2)+t1(i-1)-2*t1(i))/(2*dx*dx)
      us=us+poss*poss*(u2(2)+u1(i-1)-2*u1(i))/(2*dx*dx)
      zs=zs+poss*poss*(z2(2)+z1(i-1)-2*z1(i))/(2*dx*dx)
      zts=zts+poss*poss*(zt2(2)+zt1(i-1)-2*zt1(i))/(2*dx*dx)
      ros=ros+poss*poss*(ro2(2)+ro1(i-1)-2*ro1(i))/(2*dx*dx)
      ass=ass+poss*poss*(as2(2)+as1(i-1)-2*as1(i))/(2*dx*dx)
      hts=hts+poss*poss*(ht2(2)+ht1(i-1)-2*ht1(i))/(2*dx*dx)
      ws=ws+poss*poss*(w2(2)+w1(i-1)-2*w1(i))/(2*dx*dx)
C
      a(1)=-(1+tt1(i)*zzt1(i)/zz1(i))/rro1(i)
      a(1)=(a(1)-(1+tq*ztq/zq)/roq)/(2*cp)
      a(2)=1.0
      a(3)=0.0
      a(4)=(1/(ror*asr)+1/(rro1(i)*aas1(i)))/2
      a(5)=0.0
      a(6)=1.0
      a(7)=(-1/(ros*ass)-1/(rro1(i)*aas1(i)))/2
      a(8)=0.0
      a(9)=1.0
C
      b(1)=(htq+wq*uq)/roq+(hht1(i)+ww1(i)*uu1(i))/rro1(i)
      b(1)=b(1)*dt/(2*cp*ar)+a(1)*pq+tq
      b(2)=(1+tt1(i)*zzt1(i)/zz1(i))*(hht1(i)+ww1(i)*uu1(i))
      b(2)=b(2)*aas1(i)/(rro1(i)*tt1(i))
      b(2)=(b(2)+asr*(1+tr*ztr/zr)*(htr+wr*ur)/(ror*tr))/cp
      b(2)=(b(2)-(wr/ror+ww1(i)/rro1(i)))*dt/(2*ar)
      b(2)=b(2)-g*dt*dsin(th)+a(4)*pr+ur
      b(3)=(1+tt1(i)*zzt1(i)/zz1(i))*(hht1(i)+ww1(i)*uu1(i))
      b(3)=b(3)*aas1(i)/(rro1(i)*tt1(i))
      b(3)=(-b(3)-ass*(1+ts*zts/zs)*(hts+ws*us)/(ros*ts))/cp
      b(3)=(b(3)-(ws/ros+ww1(i)/rro1(i)))*dt/(2*ar)
```

```
c
      call dminv(a,3,det,11,mm)
      if(det.ne.0.0) goto 12
      stop "no inverse"
   12 pit=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
      tit=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
      uit=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
      difp=dabs(pp1(i)-pit)/pit
      dift=dabs(tt1(i)-tit)/tit
      if(difp.gt.(4*pdif)) goto 15
      if(dift.gt.(4*tdif)) goto 15
      if(count.gt.200) goto 15
      if(difp.gt.0.01) goto 13
      if(dift.lt.0.01) goto 14
   13 pp1(i)=pit
      tt1(i)=tit
      uu1(i)=uit
      pdif=difp
      tdif=dift
      goto 10
c
   15 write(6,16)i
   16 format('divergence - no iteration for i=',i4,'in pipe 1')
      pit=psave
      tit=tsave
      uit=usave
   14 pp1(i)=pit
      tt1(i)=tit
      uu1(i)=uit
      return
      end
```

```
c
      subroutine break2(pp1,tt1,uu1,pp2,tt2,uu2,m1,i)
c
c       this subroutine calculates steady flow conditions
c       at the break point prior to failure (pipe 2).
c
      implicit double precision(a-h,o-z)
      dimension pp1(300),tt1(300),uu1(300),pp2(300),
     & tt2(300),uu2(300)
c
      pp2(1)=pp1(m1)
      tt2(1)=tt1(m1)
      uu2(1)=uu1(m1)
      return
      end
```

```
c
      subroutine break3(p1,t1,u1,pp1,tt1,uu1,z1,zt1,ro1,as1,ht1,
     & w1,dx,tc,pc,r,ar,f,d,cp,pi,st,tw,g,dt,i,th,pat,tat)
c
c       This subroutine calculates the conditions at the break
c       using the equalisation pressure as defined by Bannister
c       and Mucklow.
c
      implicit double precision(a-h,o-z)
      dimension p1(300),t1(300),u1(300),z1(300),zt1(300),
     & w1(300),as1(300),ht1(300),pp1(300),tt1(300),
     & uu1(300),zz1(300),zzt1(300),zzp1(300),rro1(300),
     & ww1(300),aas1(300),hht1(300),a(9),ro1(300)
      integer count
c
c       first order approximation
c
      if(u1(i).eq.0.0) goto 1
      if(u1(i-1).eq.0.0) goto 1
      posq=dt*2/(1/u1(i)+1/u1(i-1))
      goto 2
    1 posq=dt*(u1(i)+u1(i-1))/2
    2 posr=dt*2/(1/(u1(i)+as1(i))+1/(u1(i-1)+as1(i-1)))
      pq=posq/dx*p1(i-1)+(1-posq/dx)*p1(i)
      uq=posq/dx*u1(i-1)+(1-posq/dx)*u1(i)
      tq=posq/dx*t1(i-1)+(1-posq/dx)*t1(i)
      zq=posq/dx*z1(i-1)+(1-posq/dx)*z1(i)
      ztq=posq/dx*zt1(i-1)+(1-posq/dx)*zt1(i)
      roq=posq/dx*ro1(i-1)+(1-posq/dx)*ro1(i)
      asq=posq/dx*as1(i-1)+(1-posq/dx)*as1(i)
      htq=posq/dx*ht1(i-1)+(1-posq/dx)*ht1(i)
      wq=posq/dx*w1(i-1)+(1-posq/dx)*w1(i)
      pr=posr/dx*p1(i-1)+(1-posr/dx)*p1(i)
      ur=posr/dx*u1(i-1)+(1-posr/dx)*u1(i)
      tr=posr/dx*t1(i-1)+(1-posr/dx)*t1(i)
      zr=posr/dx*z1(i-1)+(1-posr/dx)*z1(i)
      ztr=posr/dx*zt1(i-1)+(1-posr/dx)*zt1(i)
      ror=posr/dx*ro1(i-1)+(1-posr/dx)*ro1(i)
      asr=posr/dx*as1(i-1)+(1-posr/dx)*as1(i)
      htr=posr/dx*ht1(i-1)+(1-posr/dx)*ht1(i)
      wr=posr/dx*w1(i-1)+(1-posr/dx)*w1(i)
c
      x1=1/(ar*ror)-asr*ur*(1+tr*ztr/zr)/(ror*cp*tr*ar)
      a(1)=-wq/(roq*cp*ar)
      a(2)=wq*uq/(2*roq*cp*ar*tq)+1/dt
      a(3)=x1*wr/ur+1/dt
      a(4)=-x1*wr/(2*tr)
      a(5)=(1+tq*ztq/zq)*(pp1(i)-pq)/(roq*cp*dt)+tq/dt
      a(5)=a(5)+htq/(roq*cp*ar)+wq*uq*pp1(i)/(2*pq*roq*cp*ar)
      a(6)=(1+tr*ztr/zr)*asr*htr/(ror*cp*tr*ar)-g*dsin(th)
      a(6)=a(6)-x1*wr*pp1(i)/(2*pr)+ur/dt-(pp1(i)-pr)/(ror*asr*dt)
c
      uu1(i)=(a(4)*a(5)-a(2)*a(6))/(a(1)*a(4)-a(2)*a(3))
      tt1(i)=(a(1)*a(6)-a(3)*a(5))/(a(1)*a(4)-a(2)*a(3))
      count=0
      tsave=tt1(i)
      usave=uu1(i)
      tdif=tt1(i)*1000
   10 zz1(i)=9*tc/(128*tt1(i))-27*tc**3/(64*tt1(i)**3)
      zz1(i)=zz1(i)*pp1(i)/pc+1
      zzp1(i)=(zz1(i)-1)/pp1(i)
      zzt1(i)=81*tc**3/(64*tt1(i)**4)-9*tc/(128*tt1(i)*tt1(i))
      zzt1(i)=zzt1(i)*pp1(i)/pc
```

```
      ww1(i)=dabs(ar*rro1(i)*f*uu1(i)*uu1(i))/(2*d)
      aas1(i)=((1+zzt1(i)*tt1(i)/zz1(i))**2)*pp1(i)
      aas1(i)=aas1(i)/(rro1(i)*tt1(i)*cp)+zzp1(i)*pp1(i)/zz1(i)
      aas1(i)=(1-aas1(i))*rro1(i)/pp1(i)
      aas1(i)=1/dsqrt(dabs(aas1(i)))
      hht1(i)=pi*cp*st*d*rro1(i)*uu1(i)*(tw-tt1(i))
      count=count+1
c
c       Second order procedure
c
      if(uq.eq.0.0) goto 3
      if(uu1(i).eq.0.0) goto 3
      posq=2*dt/(1/uq+1/uu1(i))
      goto 4
    3 posq=dt*(uq+uu1(i))/2
    4 posr=2*dt/(1/(ur+asr)+1/(uu1(i)+aas1(i)))
      pq=p1(i-1)+(1-posq/dx)*(p1(i)-p1(i-2))/2
      tq=t1(i-1)+(1-posq/dx)*(t1(i)-t1(i-2))/2
      uq=u1(i-1)+(1-posq/dx)*(u1(i)-u1(i-2))/2
      zq=z1(i-1)+(1-posq/dx)*(z1(i)-z1(i-2))/2
      ztq=zt1(i-1)+(1-posq/dx)*(zt1(i)-zt1(i-2))/2
      roq=ro1(i-1)+(1-posq/dx)*(ro1(i)-ro1(i-2))/2
      asq=as1(i-1)+(1-posq/dx)*(as1(i)-as1(i-2))/2
      htq=ht1(i-1)+(1-posq/dx)*(ht1(i)-ht1(i-2))/2
      wq=w1(i-1)+(1-posq/dx)*(w1(i)-w1(i-2))/2
      pr=p1(i-1)+(1-posr/dx)*(p1(i)-p1(i-2))/2
      tr=t1(i-1)+(1-posr/dx)*(t1(i)-t1(i-2))/2
      ur=u1(i-1)+(1-posr/dx)*(u1(i)-u1(i-2))/2
      zr=z1(i-1)+(1-posr/dx)*(z1(i)-z1(i-2))/2
      ztr=zt1(i-1)+(1-posr/dx)*(zt1(i)-zt1(i-2))/2
      ror=ro1(i-1)+(1-posr/dx)*(ro1(i)-ro1(i-2))/2
      asr=as1(i-1)+(1-posr/dx)*(as1(i)-as1(i-2))/2
      htr=ht1(i-1)+(1-posr/dx)*(ht1(i)-ht1(i-2))/2
      wr=w1(i-1)+(1-posr/dx)*(w1(i)-w1(i-2))/2
      pq=pq+(p1(i)+p1(i-2)-2*p1(i-1))*(1-posq/dx)**2/2
      tq=tq+(t1(i)+t1(i-2)-2*t1(i-1))*(1-posq/dx)**2/2
      uq=uq+(u1(i)+u1(i-2)-2*u1(i-1))*(1-posq/dx)**2/2
      zq=zq+(z1(i)+z1(i-2)-2*z1(i-1))*(1-posq/dx)**2/2
      ztq=ztq+(zt1(i)+zt1(i-2)-2*zt1(i-1))*(1-posq/dx)**2/2
      roq=roq+(ro1(i)+ro1(i-2)-2*ro1(i-1))*(1-posq/dx)**2/2
      asq=asq+(as1(i)+as1(i-2)-2*as1(i-1))*(1-posq/dx)**2/2
      htq=htq+(ht1(i)+ht1(i-2)-2*ht1(i-1))*(1-posq/dx)**2/2
      wq=wq+(w1(i)+w1(i-2)-2*w1(i-1))*(1-posq/dx)**2/2
      pr=pr+(p1(i)+p1(i-2)-2*p1(i-1))*(1-posr/dx)**2/2
      tr=tr+(t1(i)+t1(i-2)-2*t1(i-1))*(1-posr/dx)**2/2
      ur=ur+(u1(i)+u1(i-2)-2*u1(i-1))*(1-posr/dx)**2/2
      zr=zr+(z1(i)+z1(i-2)-2*z1(i-1))*(1-posr/dx)**2/2
      ztr=ztr+(zt1(i)+zt1(i-2)-2*zt1(i-1))*(1-posr/dx)**2/2
      ror=ror+(ro1(i)+ro1(i-2)-2*ro1(i-1))*(1-posr/dx)**2/2
      asr=asr+(as1(i)+as1(i-2)-2*as1(i-1))*(1-posr/dx)**2/2
      htr=htr+(ht1(i)+ht1(i-2)-2*ht1(i-1))*(1-posr/dx)**2/2
      wr=wr+(w1(i)+w1(i-2)-2*w1(i-1))*(1-posr/dx)**2/2
c
      tit=(1+tq*ztq/zq)/roq+(1+tt1(i)*zzt1(i)/zz1(i))/rro1(i)
      tit=tit*(pp1(i)-pq)/(2*cp)+tq
      titt=(htq+wq*uq)/roq+(hht1(i)+ww1(i)*uu1(i))/rro1(i)
      tit=titt*dt/(2*cp*ar)+tit
      uit=(1+tt1(i)*zzt1(i)/zz1(i))*(hht1(i)+ww1(i)*uu1(i))*aas1(i)
      uit=uit/(rro1(i)*tt1(i))+(1+tr*ztr/zr)*(htr+wr*ur)*asr/(ror*tr)
      uit=uit*dt/(2*cp*ar)+ur-g*dt*dsin(th)
      uit=uit-(1/(rro1(i)*aas1(i))+1/(ror*asr))*(pp1(i)-pr)/2
      uit=uit-(ww1(i)/rro1(i)+wr/ror)*dt/(2*ar)
      dift=dabs(tt1(i)-tit)/tit
```

```
      if(count.gt.200) goto 15
      if(dift.lt.0.01)goto 14
      tt1(i)=tit
      uu1(i)=uit
      tdif=dift
      goto 10
   15 write(6,16)i
   16 format('divergence - no iteration for i=',i4,' in pipe 1')
      tit=tsave
      uit=usave
   14 tt1(i)=tit
      uu1(i)=uit
      return
      end
```

```
c
      subroutine break4(p2,t2,u2,pp2,tt2,uu2,z2,zt2,ro2,as2,ht2,
     & w2,dx,tc,pc,r,ar,f,d,cp,pi,st,tw,g,dt,i,th,pat,tt1,m1)
c
c       This subroutine calculates conditions at the point
c       immediately downstream of the break after the break
c       has occurred
c
      implicit double precision(a-h,o-z)
      dimension p2(300),t2(300),u2(300),z2(300),zt2(300),
     & w2(300),as2(300),ht2(300),ro2(300),pp2(300),tt2(300),
     & uu2(300),zz2(300),zzt2(300),zzp2(300),rro2(300),
     & ww2(300),aas2(300),hht2(300),a(9),tt1(300)
      integer m1,count
c
c       First order approximation
c
      poss=dt*2/(1/(as2(1)-u2(1))+1/(as2(2)-u2(2)))
      ps=poss/dx*p2(2)+(1-poss/dx)*p2(1)
      ts=poss/dx*t2(2)+(1-poss/dx)*t2(1)
      us=poss/dx*u2(2)+(1-poss/dx)*u2(1)
      zs=poss/dx*z2(2)+(1-poss/dx)*z2(1)
      zts=poss/dx*zt2(2)+(1-poss/dx)*zt2(1)
      ros=poss/dx*ro2(2)+(1-poss/dx)*ro2(1)
      ass=poss/dx*as2(2)+(1-poss/dx)*as2(1)
      hts=poss/dx*ht2(2)+(1-poss/dx)*ht2(1)
      ws=poss/dx*w2(2)+(1-poss/dx)*w2(1)
c
      if(u2(1).ge.0.0) goto 20
      if(u2(2).eq.0.0) goto 1
      posq=dabs(dt*2/(1/u2(1)+1/u2(2)))
      goto 2
    1 posq=dabs(u2(1)*dt/2)
    2 pq=posq/dx*p2(2)+(1-posq/dx)*p2(1)
      uq=posq/dx*u2(2)+(1-posq/dx)*u2(1)
      tq=posq/dx*t2(2)+(1-posq/dx)*t2(1)
      zq=posq/dx*z2(2)+(1-posq/dx)*z2(1)
      ztq=posq/dx*zt2(2)+(1-posq/dx)*zt2(1)
      roq=posq/dx*ro2(2)+(1-posq/dx)*ro2(1)
      asq=posq/dx*as2(2)+(1-posq/dx)*as2(1)
      htq=posq/dx*ht2(2)+(1-posq/dx)*ht2(1)
      wq=posq/dx*w2(2)+(1-posq/dx)*w2(1)
      goto 21
c
   20 if(u2(1).gt.0.0) goto 23
      pq=p2(1)
      uq=0.0
      tq=t2(1)
      zq=z2(1)
      ztq=zt2(1)
      roq=ro2(1)
      asq=as2(1)
      htq=0.0
      wq=0.0
c
   21 a(1)=1/dt+wq*uq/(2*roq*cp*ar*tq)
      a(2)=-wq/(roq*cp*ar)
      a(3)=(1+tq*ztq/zq)*(pq-pp2(1))/(roq*cp*dt)-htq/(roq*cp*ar)-tq/dt
      a(3)=a(3)-wq*uq*pp2(1)/(roq*cp*ar*2*pq)
   23 x1=ass*(1+ts*zts/zs)/(ros*cp*ar*ts)
      a(4)=ws*(-x1*us-1/(ar*ros))/(2*ts)
      a(5)=1/dt+x1*ws+ws/(ar*ros*us)
      a(6)=(ps-pp2(1))/(ros*ass*dt)-us/dt+ws*pp2(1)/(2*ar*ros*ps)
```

```
c
      if(u2(1).le.0.0) goto 24
      tt2(1)=tt1(m1)
      uu2(1)=(-a(4)*tt2(1)-a(6))/a(5)
      goto 25
c
   24 tt2(1)=(a(2)*a(6)-a(5)*a(3))/(a(5)*a(1)-a(2)*a(4))
      uu2(1)=(a(1)*a(6)-a(4)*a(3))/(a(4)*a(2)-a(1)*a(5))
c
   25 psave=pp2(1)
      usave=uu2(1)
      tsave=tt2(1)
      count=0
      tdif=tt2(1)*1000
      udif=uu2(1)*1000
   10 zz2(1)=9*tc/(128*tt2(1))-27*tc**3/(64*tt2(1)**3)
      zz2(1)=zz2(1)*pp2(1)/pc+1
      zzp2(1)=(zz2(1)-1)/pp2(1)
      zzt2(1)=81*tc**3/(64*tt2(1)**4)-9*tc/(128*tt2(1)*tt2(1))
      zzt2(1)=zzt2(1)*pp2(1)/pc
      rro2(1)=pp2(1)/(r*tt2(1)*zz2(1))
      ww2(1)=dabs(ar*rro2(1)*f*uu2(1)*uu2(1))/(2*d)
      aas2(1)=((1+tt2(1)*zzt2(1)/zz2(1))**2)*pp2(1)
      aas2(1)=aas2(1)/(rro2(1)*tt2(1)*cp)+zzp2(1)*pp2(1)/zz2(1)
      aas2(1)=(1-aas2(1))*rro2(1)/pp2(1)
      aas2(1)=1/dsqrt(dabs(aas2(1)))
      hht2(1)=pi*cp*st*d*rro2(1)*uu2(1)*(tw-tt2(1))
      count=count+1
c
c       Second order procedure
c
      poss=2*dt/(1/(ass-us)+1/(aas2(1)-uu2(1)))
      ps=p2(2)-(dx-poss)*(p2(3)-p2(1))/(2*dx)
      ts=t2(2)-(dx-poss)*(t2(3)-t2(1))/(2*dx)
      us=u2(2)-(dx-poss)*(u2(3)-u2(1))/(2*dx)
      zs=z2(2)-(dx-poss)*(z2(3)-z2(1))/(2*dx)
      zts=zt2(2)-(dx-poss)*(zt2(3)-zt2(1))/(2*dx)
      ros=ro2(2)-(dx-poss)*(ro2(3)-ro2(1))/(2*dx)
      ass=as2(2)-(dx-poss)*(as2(3)-as2(1))/(2*dx)
      hts=ht2(2)-(dx-poss)*(ht2(3)-ht2(1))/(2*dx)
      ws=w2(2)-(dx-poss)*(w2(3)-w2(1))/(2*dx)
      ps=ps+(dx-poss)**2*(p2(3)+p2(1)-2*p2(2))/(2*dx*dx)
      ts=ts+(dx-poss)**2*(t2(3)+t2(1)-2*t2(2))/(2*dx*dx)
      us=us+(dx-poss)**2*(u2(3)+u2(1)-2*u2(2))/(2*dx*dx)
      zs=zs+(dx-poss)**2*(z2(3)+z2(1)-2*z2(2))/(2*dx*dx)
      zts=zts+(dx-poss)**2*(zt2(3)+zt2(1)-2*zt2(2))/(2*dx*dx)
      ros=ros+(dx-poss)**2*(ro2(3)+ro2(1)-2*ro2(2))/(2*dx*dx)
      ass=ass+(dx-poss)**2*(as2(3)+as2(1)-2*as2(2))/(2*dx*dx)
      hts=hts+(dx-poss)**2*(ht2(3)+ht2(1)-2*ht2(2))/(2*dx*dx)
      ws=ws+(dx-poss)**2*(w2(3)+w2(1)-2*w2(2))/(2*dx*dx)
c
      if(u2(1).ge.0.0) goto 26
      posq=dabs(2*dt/(1/uq+1/uu2(1)))
      pq=p2(2)-(dx-posq)*(p2(3)-p2(1))/(2*dx)
      tq=t2(2)-(dx-posq)*(t2(3)-t2(1))/(2*dx)
      uq=u2(2)-(dx-posq)*(u2(3)-u2(1))/(2*dx)
      zq=z2(2)-(dx-posq)*(z2(3)-z2(1))/(2*dx)
      ztq=zt2(2)-(dx-posq)*(zt2(3)-zt2(1))/(2*dx)
      roq=ro2(2)-(dx-posq)*(ro2(3)-ro2(1))/(2*dx)
      asq=as2(2)-(dx-posq)*(as2(3)-as2(1))/(2*dx)
      htq=ht2(2)-(dx-posq)*(ht2(3)-ht2(1))/(2*dx)
      wq=w2(2)-(dx-posq)*(w2(3)-w2(1))/(2*dx)
      pq=pq+(dx-posq)**2*(p2(3)+p2(1)-2*p2(2))/(2*dx*dx)
```

```
      uq=uq+(dx-posq)**2*(u2(3)+u2(1)-2*u2(2))/(2*dx*dx)
      zq=zq+(dx-posq)**2*(z2(3)+z2(1)-2*z2(2))/(2*dx*dx)
      ztq=ztq+(dx-posq)**2*(zt2(3)+zt2(1)-2*zt2(2))/(2*dx*dx)
      roq=roq+(dx-posq)**2*(ro2(3)+ro2(1)-2*ro2(2))/(2*dx*dx)
      asq=asq+(dx-posq)**2*(as2(3)+as2(1)-2*as2(2))/(2*dx*dx)
      htq=htq+(dx-posq)**2*(ht2(3)+ht2(1)-2*ht2(2))/(2*dx*dx)
      wq=wq+(dx-posq)**2*(w2(3)+w2(1)-2*w2(2))/(2*dx*dx)
      goto 27
   26 if(u2(1).gt.0.0) goto 28
      pq=p2(1)
      uq=0.0
      tq=t2(1)
      zq=z2(1)
      ztq=zt2(1)
      roq=ro2(1)
      asq=as2(1)
      htq=0.0
      wq=0.0
   27 tit=(1+tq*ztq/zq)/roq+(1+tt2(1)*zzt2(1)/zz2(1))/rro2(1)
      tit=tit*(pp2(1)-pq)+(hht2(1)+ww2(1)*uu2(1))*dt/(rro2(1)*ar)
      tit=(tit+(htq+wq*uq)*dt/(roq*ar))/(2*cp)+tq
      goto 29
   28 tit=tt1(m1)
   29 uit=aas2(1)*(1+tt2(1)*zzt2(1)/zz2(1))*(hht2(1)+ww2(1)*uu2(1))
      uit=uit/(rro2(1)*tt2(1))+ass*(1+ts*zts/zs)*(hts+ws*us)/(ros*ts)
      uit=(uit/(cp*ar)+ww2(1)/(ar*rro2(1))+ws/(ar*ros))*dt
      uit=((pp2(1)-ps)*(1/(ros*ass)+1/(rro2(1)*aas2(1)))-uit)/2
      uit=uit-g*dt*dsin(th)+us
      dift=dabs(tt2(1)-tit)/tit
      if (dift.gt.(4*tdif)) goto 15
      if(count.gt.200) goto 15
      if (dift.lt.0.01) goto 13
      uu2(1)=uit
      tt2(1)=tit
      tdif=dift
      goto 10
   15 write(6,16)
   16 format('divergence - no iteration for i=1 in pipe 2')
      tit=tsave
      uit=usave
   13 tt2(1)=tit
      uu2(1)=uit
      return
      end
```

```
c**********************************************************************
c
      subroutine subup (p1,t1,u1,dx,tt1,uu1,tc,pc,r,ar,
     & f,d,cp,pi,st,tw,g,dt,i,z1,zp1,zt1,ro1,w1,as1,ht1,th,
     & pp1,tt1,uu1)
c
c       this subroutine calculates p,u and t at the upstream
c       end of the pipe assuming constant pressure and
c       constant mass flow rate.
c
      implicit double precision (a-h,o-z)
      dimension p1(300),t1(300),u1(300),z1(300),zt1(300),
     & zp1(300),ro1(300),as1(300),ht1(300),w1(300),
     & tt1(300),uu1(300),zz1(300),zzt1(300),zzp1(300),
     & rro1(300),aas1(300),hht1(300),ww1(300),pp1(300)
      integer count
c
c       first order approximation
c
      poss=dt*2/(1/(as1(1)-u1(1))+1/(as1(2)-u1(2)))
      ps=(poss/dx)*p1(2)+(1-poss/dx)*p1(1)
      us=(poss/dx)*u1(2)+(1-poss/dx)*u1(1)
      ts=(poss/dx)*t1(2)+(1-poss/dx)*t1(1)
      zs=(poss/dx)*z1(2)+(1-poss/dx)*z1(1)
      zts=(poss/dx)*zt1(2)+(1-poss/dx)*zt1(1)
      ros=(poss/dx)*ro1(2)+(1-poss/dx)*ro1(1)
      ass=(poss/dx)*as1(2)+(1-poss/dx)*as1(1)
      hts=(poss/dx)*ht1(2)+(1-poss/dx)*ht1(1)
      ws=(poss/dx)*w1(2)+(1-poss/dx)*w1(1)
      x1=1/dt+(1+ts*zts/zs)*ws*ass/(ros*cp*ts*ar)
      x1=x1+ws/(ar*ros*us)
      x2=(1+ts*zts/zs)*(hts-ws*us/2)*ass/(ros*cp*ts*ar)
      x2=x2-ws/(2*ar*ros)-us/dt-(p1(1)-ps)/(ros*ass*dt)
      x2=x2+g*dsin(th)
      x3=(1+ts*zts/zs)*ass*ws*us*ro1(1)*u1(1)/(cp*ts)
      x3=(x3+ws*ro1(1)*u1(1))/(2*ar*ros**2)
      uu1(1)=(dsqrt(x2*x2-4*x1*x3)-x2)/(2*x1)
      usave=uu1(1)
      count=0
      udif=uu1(1)*1000
    4 rro1(1)=ro1(1)*u1(1)/uu1(1)
      tt1(1)=p1(1)/(rro1(1)*z1(1)*r)
      zz1(1)=9*tc/(128*tt1(1))-27*tc**3/(64*tt1(1)**3)
      zz1(1)=zz1(1)*p1(1)/pc+1
      zzp1(1)=(zz1(1)-1)/p1(1)
      zzt1(1)=81*tc**3/(64*tt1(1)**4)-9*tc/(128*tt1(1)*tt1(i))
      zzt1(1)=zzt1(1)*p1(1)/pc
      ww1(1)=dabs(ar*rro1(1)*f*uu1(1)*uu1(i))/(2*d)
      aas1(1)=((1+zzt1(1)*tt1(1)/zz1(1))**2)
      aas1(1)=aas1(1)/(rro1(1)*tt1(1)*cp)+zzp1(1)/zz1(1)
      aas1(1)=(1/p1(1)-aas1(1))*rro1(1)
      aas1(1)=1/dsqrt(dabs(aas1(1)))
      hht1(1)=pi*cp*st*d*rro1(1)*uu1(1)*(tw-tt1(1))
      count=count+1
c
c       second order procedure
c
      poss=dt*2/(1/(aas1(1)-uu1(1))+1/(ass-us))
      ps=p1(2)-(dx-poss)*(p1(3)-p1(1))/(2*dx)
      ps=ps+(dx-poss)**2*(p1(3)+p1(1)-2*p1(2))/(2*dx*dx)
      ts=t1(2)-(dx-poss)*(t1(3)-t1(1))/(2*dx)
      ts=ts+(dx-poss)**2*(t1(3)+t1(1)-2*t1(2))/(2*dx*dx)
      us=u1(2)-(dx-poss)*(u1(3)-u1(1))/(2*dx)
      us=us+(dx-poss)**2*(u1(3)+u1(1)-2*u1(2))/(2*dx*dx)
```

```
      zs=zs+(dx-poss)**2*(z1(3)-z1(1))/(2*dx*dx)
      ws=w1(2)-(dx-poss)*(w1(3)-w1(1))/(2*dx)
      ws=ws+(dx-poss)**2*(w1(3)+w1(1)-2*w1(2))/(2*dx*dx)
      zts=zt1(2)-(dx-poss)*(zt1(3)-zt1(1))/(2*dx)
      zts=zts+(dx-poss)**2*(zt1(3)+zt1(1)-2*zt1(2))/(2*dx*dx)
      ros=ro1(2)-(dx-poss)*(ro1(3)-ro1(1))/(2*dx)
      ros=ros+(dx-poss)**2*(ro1(3)+ro1(1)-2*ro1(2))/(2*dx*dx)
      ass=as1(2)-(dx-poss)*(as1(3)-as1(1))/(2*dx)
      ass=ass+(dx-poss)**2*(as1(3)+as1(1)-2*as1(2))/(2*dx*dx)
      hts=ht1(2)-(dx-poss)*(ht1(3)-ht1(1))/(2*dx)
      hts=hts+(dx-poss)**2*(ht1(3)+ht1(1)-2*ht1(2))/(2*dx*dx)
C
      uit=us+(1/(ros*ass)+1/(rro1(1)+aas1(1)))*(p1(1)-ps)/2
      x1=ass*(1+ts*zts/zs)*(hts+ws*us)/(ros*ts)
      x2=aas1(1)*(1+tt1(1)*zzt1(1)/zz1(1))*(hht1(1)+ww1(1)*uu1(1))
      x2=x2/(rro1(1)*tt1(1))
      x3=(x1+x2)/(cp*ar)+ws/(ar*ros)+ww1(1)/(ar+rro1(1))
      x3=x3*dt/2+g*dt*dsin(th)
      uit=uit-x3
      difu=dabs(uu1(1)-uit)
      if (count.gt.200) goto 1
      if(difu.lt.0.01) goto 3
      uu1(1)=uit
      goto 4
    1 write(22,2)
    2 format('no iteration for i=1 in pipe 1')
      uit=usave
    3 uu1(i)=uit
      rro1(1)=ro1(1)*u1(1)/uu1(1)
      tt1(1)=p1(1)/(rro1(1)*zz1(1)*r)
      pp1(1)=p1(1)
      return
      end
```

```
*****************************************************************************
c
      subroutine down1(p2,t2,u2,dx,pp2,tt2,uu2,tc,pc,r,ar,
     & th,f,d,cp,pi,st,tw,g,dt,i,z2,zp2,zt2,ro2,w2,as2,
     & ht2)
c
c       this subroutine calculates p,u and t at the downstream
c       boundary condition assuming a constant temperature
c       non-return valve situation.(valve closes if u<0 m/s)
c
      implicit double precision (a-h,o-z)
      dimension p2(300),u2(300),t2(300),z2(300),zt2(300),ro2(300),
     & pp2(300),uu2(300),tt2(300),rro2(300),w2(300),as2(300),ht2(300),
     & zz2(300),zzp2(300),zzt2(300),ww2(300),aas2(300),hht2(300)
      integer count
c
c       First order approximation
c
      posr=dt*2/(1/(u2(i)+as2(i))+1/(u2(i-1)+as2(i-1)))
      pr=(posr/dx)*p2(i-1)+(1-posr/dx)*p2(i)
      ur=(posr/dx)*u2(i-1)+(1-posr/dx)*u2(i)
      tr=(posr/dx)*t2(i-1)+(1-posr/dx)*t2(i)
      zr=(posr/dx)*z2(i-1)+(1-posr/dx)*z2(i)
      ztr=(posr/dx)*zt2(i-1)+(1-posr/dx)*zt2(i)
      ror=(posr/dx)*ro2(i-1)+(1-posr/dx)*ro2(i)
      asr=(posr/dx)*as2(i-1)+(1-posr/dx)*as2(i)
      htr=(posr/dx)*ht2(i-1)+(1-posr/dx)*ht2(i)
      wr=(posr/dx)*w2(i-1)+(1-posr/dx)*w2(i)
c
      x1=1/(ror*asr)-asr*wr*ur*dt*(1+tr*ztr/zr)/(2*pr*ror*cp*tr*ar)
      x1=x1+wr*dt/(2*ar*ror*pr)
      x2=1-asr*wr*dt*(1+tr*ztr/zr)/(ror*cp*tr*ar)+wr*dt/(ar*ror*ur)
      x3=asr*dt*(1+tr*ztr/zr)*(htr-wr*ur*t2(i)/(2*tr))/(ror*cp*tr*ar)
      x3=x3+wr*dt*t2(i)/(2*tr*ar*ror)-g*dt*dsin(th)+pr/(ror*asr)+ur
c
      if(u2(i).eq.0.0) goto 3
      if(uu2(i-1).lt.0.0) goto 3
      if(u2(i-1).eq.0.0) goto 1
      posq=dabs(dt*2/(1/u2(i)+1/u2(i-1)))
      goto 2
    1 posq=dt*u2(i)/2
    2 pq=(posq/dx)*p2(i-1)+(1-posq/dx)*p2(i)
      uq=(posq/dx)*u2(i-1)+(1-posq/dx)*u2(i)
      tq=(posq/dx)*t2(i-1)+(1-posq/dx)*t2(i)
      zq=(posq/dx)*z2(i-1)+(1-posq/dx)*z2(i)
      ztq=(posq/dx)*zt2(i-1)+(1-posq/dx)*zt2(i)
      roq=(posq/dx)*ro2(i-1)+(1-posq/dx)*ro2(i)
      asq=(posq/dx)*as2(i-1)+(1-posq/dx)*as2(i)
      htq=(posq/dx)*ht2(i-1)+(1-posq/dx)*ht2(i)
      wq=(posq/dx)*w2(i-1)+(1-posq/dx)*w2(i)
c
      x4=-(1+tq*ztq/zq)/(roq*cp)-wq*uq*dt/(2*roq*cp*ar*pq)
      x5=-wq*dt/(roq*cp*ar)
      x6=(htq-wq*uq*t2(i)/(2*tq))*dt/(ar*roq*cp)-t2(i)+tq
      x6=x6-(1+tq*ztq/zq)*pq/(roq*cp)
c
      uu2(i)=(x3*x4-x1*x6)/(x4*x2-x1*x5)
      pp2(i)=(x5*x3-x2*x6)/(x1*x5-x4*x2)
      goto 4
    3 uu2(i)=0.0
      pp2(i)=x3/x1
    4 psave=pp2(i)
      count=0
      pdif=pp2(i)*1000
```

```
      vv2(i)=v2(i)
   10 zz2(i)=9*tc/(128*tt2(i))-27*tc**3/(64*tt2(i)**3)
      zz2(i)=zz2(i)*pp2(i)/pc+1
      zzp2(i)=(zz2(i)-1)/pp2(i)
      zzt2(i)=81*tc**3/(64*tt2(i)**4)-9*tc/(128*tt2(i)*tt2(i))
      zzt2(i)=zzt2(i)*pp2(i)/pc
      rro2(i)=pp2(i)/(r*tt2(i)*zz2(i))
      ww2(i)=dabs(ar*rro2(i)*f*uu2(i)*uu2(i))/(2*d)
      aas2(i)=((1+zzt2(i)*tt2(i)/zz2(i))**2)*pp2(i)
      aas2(i)=aas2(i)/(rro2(i)*tt2(i)*cp)+zzp2(i)*pp2(i)/zz2(i)
      aas2(i)=(1-aas2(i))*rro2(i)/pp2(i)
      aas2(i)=1/dsqrt(dabs(aas2(i)))
      hht2(i)=pi*cp*st*d*rro2(i)*uu2(i)*(tw-tt2(i))
      count=count+1
c
c       second order procedure
c
      posr=dt*2/(1/(asr+ur)+1/(aas2(i)+uu2(i)))
      pr=p2(i-1)+(dx-posr)*(p2(i)-p2(i-2))/(2*dx)
      ur=u2(i-1)+(dx-posr)*(u2(i)-u2(i-2))/(2*dx)
      tr=t2(i-1)+(dx-posr)*(t2(i)-t2(i-2))/(2*dx)
      zr=z2(i-1)+(dx-posr)*(z2(i)-z2(i-2))/(2*dx)
      ztr=zt2(i-1)+(dx-posr)*(zt2(i)-zt2(i-2))/(2*dx)
      ror=ro2(i-1)+(dx-posr)*(ro2(i)-ro2(i-2))/(2*dx)
      asr=as2(i-1)+(dx-posr)*(as2(i)-as2(i-2))/(2*dx)
      htr=ht2(i-1)+(dx-posr)*(ht2(i)-ht2(i-2))/(2*dx)
      wr=w2(i-1)+(dx-posr)*(w2(i)-w2(i-2))/(2*dx)
      pr=pr+(dx-posr)**2*(p2(i)+p2(i-2)-2*p2(i-1))/(2*dx*dx)
      ur=ur+(dx-posr)**2*(u2(i)+u2(i-2)-2*u2(i-1))/(2*dx*dx)
      tr=tr+(dx-posr)**2*(t2(i)+t2(i-2)-2*t2(i-1))/(2*dx*dx)
      zr=zr+(dx-posr)**2*(z2(i)+z2(i-2)-2*z2(i-1))/(2*dx*dx)
      ztr=ztr+(dx-posr)**2*(zt2(i)+zt2(i-2)-2*zt2(i-1))/(2*dx*dx)
      ror=ror+(dx-posr)**2*(ro2(i)+ro2(i-2)-2*ro2(i-1))/(2*dx*dx)
      asr=asr+(dx-posr)**2*(as2(i)+as2(i-2)-2*as2(i-1))/(2*dx*dx)
      htr=htr+(dx-posr)**2*(ht2(i)+ht2(i-2)-2*ht2(i-1))/(2*dx*dx)
      wr=wr+(dx-posr)**2*(w2(i)+w2(i-2)-2*w2(i-1))/(2*dx*dx)
c
      x1=(1/(ror*asr)+1/(rro2(i)*aas2(i)))/2
      x2=1.0
      x3=aas2(i)*(1+tt2(i)*zzt2(i)/zz2(i))*(hht2(i)+ww2(i)*uu2(i))
      x3=x3/(rro2(i)*tt2(i))+asr*(1+tr*ztr/zr)*(htr+wr*ur)/(ror*tr)
      x3=(x3/cp-ww2(i)/rro2(i)-wr/ror)*dt/(2*ar)-g*dt*dsin(th)
      x3=x3+x1*pr+ur
c
      if(uu2(i).le.0.0) goto 5
      posq=dabs(dt*2/(1/uu2(i)+1/uq))
      pq=p2(i-1)+(dx-posq)*(p2(i)-p2(i-2))/(2*dx)
      uq=u2(i-1)+(dx-posq)*(u2(i)-u2(i-2))/(2*dx)
      tq=t2(i-1)+(dx-posq)*(t2(i)-t2(i-2))/(2*dx)
      zq=z2(i-1)+(dx-posq)*(z2(i)-z2(i-2))/(2*dx)
      ztq=zt2(i-1)+(dx-posq)*(zt2(i)-zt2(i-2))/(2*dx)
      roq=ro2(i-1)+(dx-posq)*(ro2(i)-ro2(i-2))/(2*dx)
      asq=as2(i-1)+(dx-posq)*(as2(i)-as2(i-2))/(2*dx)
      htq=ht2(i-1)+(dx-posq)*(ht2(i)-ht2(i-2))/(2*dx)
      wq=w2(i-1)+(dx-posq)*(w2(i)-w2(i-2))/(2*dx)
      pq=pq+(dx-posq)**2*(p2(i)+p2(i-2)-2*p2(i-1))/(2*dx*dx)
      uq=uq+(dx-posq)**2*(u2(i)+u2(i-2)-2*u2(i-1))/(2*dx*dx)
      tq=tq+(dx-posq)**2*(t2(i)+t2(i-2)-2*t2(i-1))/(2*dx*dx)
      zq=zq+(dx-posq)**2*(z2(i)+z2(i-2)-2*z2(i-1))/(2*dx*dx)
      ztq=ztq+(dx-posq)**2*(zt2(i)+zt2(i-2)-2*zt2(i-1))/(2*dx*dx)
      roq=roq+(dx-posq)**2*(ro2(i)+ro2(i-2)-2*ro2(i-1))/(2*dx*dx)
      asq=asq+(dx-posq)**2*(as2(i)+as2(i-2)-2*as2(i-1))/(2*dx*dx)
      htq=htq+(dx-posq)**2*(ht2(i)+ht2(i-2)-2*ht2(i-1))/(2*dx*dx)
      wq=wq+(dx-posq)**2*(w2(i)+w2(i-2)-2*w2(i-1))/(2*dx*dx)
```

```
      x4=(-(1+tt2(i)*zzt2(i)/zz2(i))/rro2(i)-(1+tq*ztq/zq)/roq)/(2*cp)
      x5=0.0
      x6=((hht2(i)+ww2(i)*uu2(i))/rro2(i)+(htq+wq*uq)/roq)*dt/(2*cp*ar)
      x6=x6-tt2(i)+tq+x4*pq
c
      uit=(x3*x4-x1*x6)/x4
      pit=x6/x4
      goto 6
    5 uit=0.0
      pit=x3/x1
    6 difp=dabs(pp2(i)-pit)/pit
      if(difp.gt.(4*pdif)) goto 7
      if(count.gt.200) goto 7
      if(difp.lt.0.01) goto 8
      pp2(i)=pit
      uu2(i)=uit
      pdif=difp
      goto 10
c
    7 write(6,9)i
    9 format('divergence - no iteration for i=',i4,' in pipe 2')
      pit=psave
      uit=usave
    8 pp2(i)=pit
      uu2(i)=uit
      return
      end
```

```
      subroutine getfil(i)
      character*80 filnam
1233  write(*,1234)
1234  format(1x,'filename ? ',$)
      read*, filnam
      open(unit=i,file=filnam,err=1235)
      rewind i
      return
1235  print*, "Can't open",filnam
      goto 1233
      end
```

```
      subroutine dminv(a,n,d,l,m)
c
c        Description of parameters
c              a - Input matrix replaced by inverse on exit
c                  Matrix a is the whole matrix stored in a single
c                  dimension array columnwise.
c
c              n - Order of matrix 'a'
c
c              d - Resultant determinant
c
c              l - Work vector of length 'n'
c
c              m - Work vector of length 'n'
c
c
      double precision a(*),d,biga,hold
      integer l(*),m(*)
      d=1.0
      nk=-n
      do 80 k=1,n
      nk=nk+n
      l(k)=k
      m(k)=k
      kk=nk+k
      biga=a(kk)
      do 20 j=k,n
      iz=n*(j-1)
      do 20 i=k,n
      ij=iz+i
   10 if(dabs(biga)-dabs(a(ij)))15,20,20
   15 biga=a(ij)
      l(k)=i
      m(k)=j
   20 continue
c
c        Interchange rows
c
      j=l(k)
      if(j-k)35,35,25
   25 ki=k-n
      do 30 i=1,n
      ki=ki+n
      hold=-a(ki)
      ji=ki-k+j
      a(ki)=a(ji)
   30 a(ji)=hold
c
c        Interchange columns
c
   35 i=m(k)
      if(i-k)45,45,38
   38 jp=n*(i-1)
      do 40 j=1,n
      jk=nk+j
      ji=jp+j
      hold=-a(jk)
      a(jk)=a(ji)
   40 a(ji)=hold
c
c        Divide column by minus pivot value ( -biga )
c
   45 if(biga)48,46,48
   46 d=0.0
      return
```

```
      if(i-k)50,55,50
   50 ik=nk+i
      a(ik)=a(ik)/(-biga)
   55 continue
C
C        Reduce matrix
C
      do 65 i=1,n
      ik=nk+i
      hold=a(ik)
      ij=i-n
      do 65 j=1,n
      ij=ij+n
      if(i-k)60,65,60
   60 if(j-k)62,65,62
   62 kj=ij-i+k
      a(ij)=hold*a(kj)+a(ij)
   65 continue
C
C        Divide row by pivot
C
      kj=k-n
      do 75 j=1,n
      kj=kj+n
      if(j-k)70,75,70
   70 a(kj)=a(kj)/biga
   75 continue
C
C        Product of pivots
C
      d=d*biga
C
C        Replace pivot by reciprocal
C
      a(kk)=1.0/biga
   80 continue
C
C        Final row and column interchange
C
      k=n
  100 k=k-1
      if(k)150,150,105
  105 i=l(k)
      if(i-k)120,120,108
  108 jq=n*(k-1)
      jr=n*(i-1)
      do 110 j=1,n
      jk=jq+j
      hold=a(jk)
      ji=jr+j
      a(jk)=-a(ji)
  110 a(ji)=hold
  120 j=m(k)
      if(j-k)100,100,125
  125 ki=k-n
      do 130 i=1,n
      ki=ki+n
      hold=a(ki)
      ji=ki-k+j
      a(ki)=-a(ji)
  130 a(ji)=hold
      goto 100
  150 return
      end
```

GRAPHICS PROGRAMS

```
c       Routine to plot graphs of pressure vs time
c
      real*4 p(20),u(20),t(20),wisen(20),wisot(20),time(500),pplot(500)
      dimension int(20)
      character*20 yorn
  800 format(A20)
      iflag=0
      ipen=0
      open(unit=1,file='theory')
      rewind (1)
      open(unit=2,file='expt')
      rewind (2)
      yspot=65.0
  100 print*,'Enter value of i required for plot'
      rewind(1)
      rewind(2)
      read(5,*) iset
      read(1,*) nint
      write(6,25)
   25 format('Do you require the experimental data plotted?')
      read (5,800) yorn
      if(yorn.eq.'N'.or.yorn.eq.'n') goto 300
      read(2,*) nint2
      ipen=1
      l=0
   20 l=l+1
      read(2,*,end=400) time (l)
      do 3 i=1,nint2
      read (2,*) int(i),p(i),u(i),t(i),wisen(i),wisot(i)
    3 continue
      do 4 j=1,nint2
      if(int(j).ne.iset) goto 4
      pplot(l)=p(j)
      goto 20
    4 if(j.eq.nint2) goto 400
      continue
      goto 20
  400 l=l-1
      goto 500
  300 l=0
   10 l=l+1
      read(1,*,end=200) time(l)
      do 1 i=1,nint
      read(1,*) int(i),p(i),u(i),t(i),wisen(i),wisot(i)
    1 continue
      do 2 j=1,nint
      if(int(j).ne.iset) goto 2
      pplot(l)=p(j)
    2 continue
      goto 10
  200 l=l-1
  500 pmax=10000 <-- Maximum Pressure Value
      if(iflag.gt.0) goto 11
      call hp7550
      call devpap(380.0,250.0,0)
      call window(2)
      call pensel(1,0.5,2)
```

```
      call axipos(1,30.0,30.0,150.0,1)
      call axipos(1,30.0,30.0,150.0,2)
      call axisca(1,1,0.0,time(1),1)
      call axisca(1,1/2,0.0,pmax,2)
      call axidra(1,1,1)
      call axidra(-2,-1,2)
      call movto2(80.0,15.0)
      call chastr('time (secs)')
      call movto2(15.0,80.0)
      call chaang(90.0)
      call chastr('pressure (KPa)')
      call chaang(0.0)
      call movto2(90.0,200.0)
      call chastr('Foothills Test NABTF3  ')
   11 if(ipen.eq.1) goto 12
      call pensel(1,0.5,2)
      call grapol(time,pplot,1)
      goto 13
   12 call pensel(2,0.5,2)
      print*,'Enter code for symbol type (1-6)'
      read(5,*) nsym
      call grapol(time,pplot,1)
      call grasym(time,pplot,1,nsym,1)
      yspot=yspot-5.0
      call movto2(150.0,yspot)
      call symbol(nsym)
      call movto2(155.0,yspot)
      call chastr('i= ')
      call chaint(iset,4)
   13 call chamod
      ipen=0
      iflag=1
      print*,'Do you require another i? Y or N'
      read(5,800) yorn
      if(yorn.eq.'Y'.or.yorn.eq.'y') goto 100
      call piccle
      call devend
      stop
      end
```

← Title of Graph

```
c       Routine to plot graphs of pressure vs wavespeed
c
        real*4 p(20),u(20),t(20),wisen(20),wisot(20),time(500),pplot(500)
        real*4 wplot(500)
        dimension int(20)
        character*20 yorn
  800   format(A20)
        iflag=0
        ipen=0
        open(unit=1,file='theory')
        rewind (1)
        open(unit=2,file='expt')
        rewind (2)
        yspot=100.0
  100   print*,'Enter value of i required for plot'
        rewind (1)
        rewind (2)
        read(5,*) iset
        read(1,*) nint
        write(6,25)
   25   format('Do you require the experimental data plotted?')
        read(5,800) yorn
        if (yorn.eq.'N'.or.yorn.eq.'n') goto 700
        read(2,*) nint2
        ipen=1
        l=0
   20   l=l+1
        read(2,*,end=400) time(l)
        do 3 i=1,nint2
        read(2,*) int(i),p(i),u(i),t(i),wisen(i),wisot(i)
    3   continue
        do 4 j=1,nint2
        if(int(j).ne.iset) goto 4
        pplot(l)=p(j)
        wplot(l)=wisen(j)
    4   continue
        goto 20
  400   l=l-1
        goto 500
  700   write(6,22)
   22   format('Theoretical Plot:-')
        write(6,21)
   21   format('Do you want isothermal (1) or isentropic (2) wavespeed?')
        read(5,*) ii
        if(ii.eq.2) goto 300
        l=0
   30   l=l+1
        read (1,*,end=600) time (l)
        do 5 i=1,nint
        read(1,*) int(i),p(i),u(i),t(i),wisen(i),wisot(i)
    5   continue
        do 6 j=1,nint
        if(int(j).ne.iset) goto 6
        pplot(l)=p(j)
        wplot(l)=wisot(j)
    6   continue
        goto 30
  600   l=l-1
        goto 500
  300   l=0
   10   l=l+1
```

```
      read(1,*,end=200) time(1)
      do 1 i=1,nint
      read(1,*) int(i),p(i),u(i),t(i),wisen(i),wisot(i)
    1 continue
      do 2 j=1,nint
      if(int(j).ne.iset) goto 2
      pplot(1)=p(j)
      wplot(1)=wisen(j)
    2 continue
      goto 10
  200 l=l-1
  500 wmax=500
      pmax=10000
      if(iflag.gt.0) goto 11
      call hp7550
      call devpap(380.0,250.0,0)
      call window(2)
      call pensel(1,0.5,2)
      call axipos(1,30.0,30.0,150.0,1)
      call axipos(1,30.0,30.0,150.0,2)
      call axisca(1,1,0.0,wmax,1)
      call axisca(1,1/2,0.0,pmax,2)
      call axidra(1,1,1)
      call axidra(-2,-1,2)
      call movto2(80.0,15.0)
      call chastr('wavespeed (m/s)')
      call movto2(15.0,80.0)
      call chaang(90.0)
      call chastr('pressure (KPa)')
      call chaang(0.0)
      call movto2(40.0,200.0)
      call chastr('Groves Shock Tube Test - Natural Gas')
   11 if(ipen.eq.1) goto 24
      call pensel(1,0.5,2)
      call grapol(wplot,pplot,1)
      goto 23
   24 print*,'Enter code for symbol type (1-6)'
      read(5,*) nsym
      call pensel(2,0.5,2)
      call grapol(wplot,pplot,1)
      call grasym(wplot,pplot,1,nsym,0)
      yspot=yspot-5.0
      call movto2(150.0,yspot)
      call symbol(nsym)
      call movto2(155.0,yspot)
      call chastr('i= ')
      call chaint(iset,4)
   23 call chamod
      ipen=0
      iflag=1
      print*,'Do you require another i? Y or N'
      read(5,800) yorn
      if(yorn.eq.'Y'.or.yorn.eq.'y') goto 100
      call piccle
      call devend
      stop
      end
```

— Maximum Wavespeed and Pressure Values

Title of Graph

APPENDIX VI. PREPARATION OF GAS DATA

i) Specific Heat at Constant Pressure C_p

From the known molar compositions (Table A1, page 222), the mass composition of the gas mixtures could be calculated using the Method of Mixtures:-

$$m_i = \frac{x_i \ M_i}{\sum_{i=1,n} (x_i \ M_i)} \qquad (1)$$

where m_i = mass percentage of component i

x_i = molar percentage of component i

M_i = molecular weight of component i

n = number of components.

The calculated mean percentages are presented in Table A2 on page 223.

The specific heats of the mixtures were then obtained by applying the following formula:-

$$\overline{C_p} = \sum_{i=1,n} m_i \ C_{p_i} \qquad (2)$$

where $\overline{C_p}$ = mean specific heat of mixture

m_i = mass percentage of component i

C_{p_i} = specific heat of component i

n = number of components.

EXAMPLE

To calculate the specific heat of the natural gas mixture used by Groves [1978]:

From Table A1, the molecular composition (neglecting components of less than 0.001%) is:-

Molecular %	Component	Molecular Weight (From Table A3)
1.498	N_2	28.013
1.073	CO_2	44.010
83.266	CH_4	16.043
9.608	C_2H_6	30.070
3.597	C_3H_8	44.094
0.3414	iC_4H_{10}	58.124
0.4581	nC_4H_{10}	58.124
0.0403	iC_5H_{12}	72.151
0.0342	nC_5H_{12}	72.151
0.0046	C_6H_{14}	86.178

the mass percentage of, for example, methane (CH_4) is

$$m_{CH_4} = \frac{83.266 \times 16.043}{((1.498 \times 28.013)+(1.073 \times 44.010)+(83.266 \times 16.043)+\ldots)}$$

$$= 69.40\%$$

The mass percentage of the other components were calculated in a similar manner and the mean specific heat could then be obtained from the specific heats of the components given in Table A3 on page 224.

$$C_p = (0.0218 \times 1037)+(0.0245 \times 819)+(0.6940 \times 2174)+(0.1501 \times 1533)+\ldots$$

$$= 1960 \text{ J/kg K.}$$

ii) Specific Gas Constant R

The mean molecular weight of a gas mixture was calculated from the formula:-

$$\bar{M} = \frac{\sum_{i=1,n} (x_i M_i)}{\sum_{i=1,n} x_i} \qquad (3)$$

where x_i = molecular percentage of component i

M_i = molecular weight of component i

n = number of components.

The specific gas constant for the gas mixture could then be calculated from:-

$$R = \frac{R^*}{\bar{M}} \qquad (4)$$

where R = specific gas constant for the gas mixture

R^* = universal gas constant.

EXAMPLE

To calculate the specific gas constant of the natural gas mixture used by Groves [1978]:

$$\text{Mean Molecular Weight } M = \frac{\langle(1.498\text{x}28.013)+(1.073\text{x}44.010)+(83.266\text{x}16.043)+\ldots\rangle}{(1.498 + 1.073 + 83.266 + 9.608 + 3.597 + \ldots\ldots)}$$

$$= 19.264$$

$$\therefore \quad R = \frac{8314}{19.264} \quad \text{J/kg K}$$

$$= 432 \text{ J/kg K.}$$

iii) Critical Temperature T_c

In the chosen procedure, the ratios of the boiling point temperatures of the components of the gas were used to obtain coefficients A_{xy} and A_{yx} from the charts below.

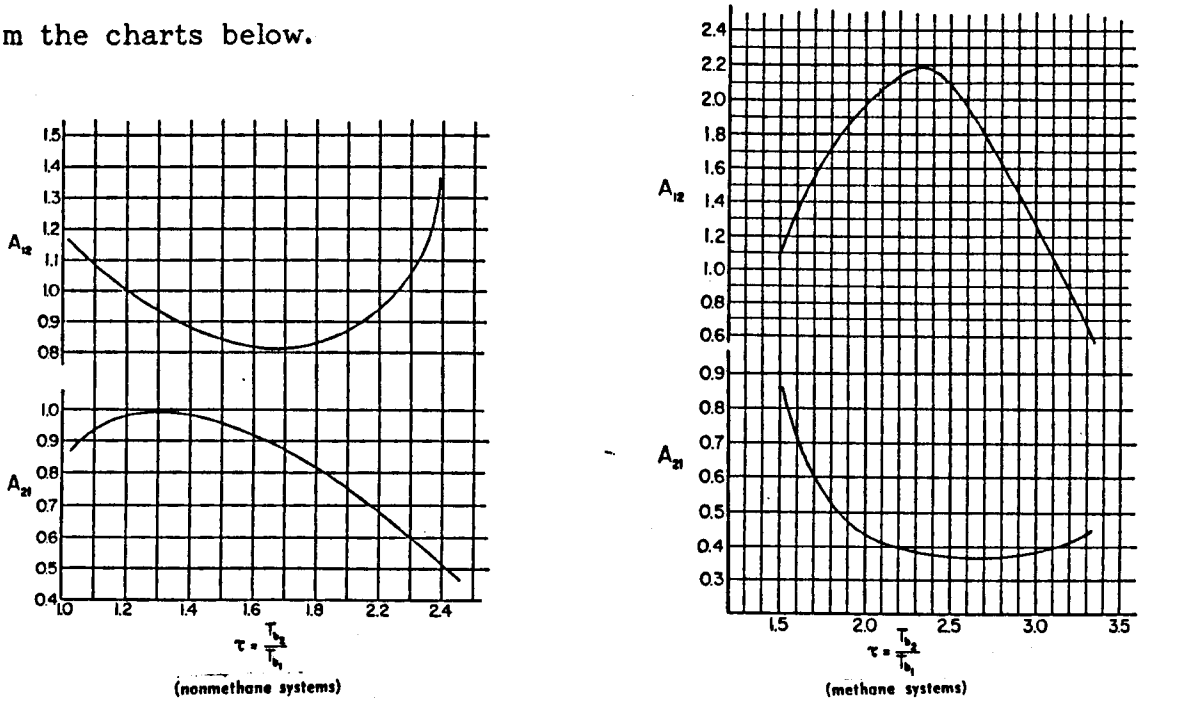


Figure A.2. Coefficient Charts for use in the Method of Grieves and Thodos

The following formula was then implemented to obtain values for the critical temperatures:-

$$T_{cm} = \sum_{i=1,n} \left[\frac{T_{c_i}}{1 + \frac{1}{x_i} \sum_{j=i \neq 1,n} A_{ij} \cdot x_j} \right] \qquad (5)$$

where T_{c_i} = critical temperature of component i (°R)

T_{cm} = mixture critical temperature (°R)

x_i = molar fraction of component i

x_j = molar fraction of component j

A_{ij} = coefficient taken from chart

n = number of components.

EXAMPLE

For Groves' data, neglecting the N_2 and CO_2 components of the natural gas, the main components are:-

	Molar fraction	T_b ('R)	T_c ('R)
1. Methane	0.83266	200.9	342.7
2. Ethane	0.09608	332.3	549.7
3. Propane	0.03597	416.0	665.6

From these three components three binary pairs exist. The ratios of the boiling point temperatures and the corresponding coefficients are listed below:

	T_{b_j}/T_{b_i}	Coefficients (from Figure A.2)	
Methane-ethane	1.65	$A_{12} = 1.46$	$A_{21} = 0.65$
Methane-propane	2.07	$A_{13} = 2.06$	$A_{31} = 0.42$
Ethane-propane	1.25	$A_{23} = 0.966$	$A_{32} = 0.986$

Therefore

$$\bar{T}_{cm} = \frac{342.7}{1 + \frac{9.608}{83.266}(1.46) + \frac{3.597}{83.266}(2.06)} + \frac{549.7}{1 + \frac{83.266}{9.608}(0.65) + \frac{3.597}{9.608}(0.966)}$$

$$+ \frac{665.6}{1 + \frac{83.266}{3.597}(0.42) + \frac{9.608}{3.597}(0.986)}$$

$$= 400.8\ ^\circ R$$

$$= -50.5^\circ C.$$

iv) Critical Pressure P_c

The critical pressures were calculated from the following equation (Prausnitz and Gunn [1958]):-

$$P_{cm} = \frac{R^* \left(\sum_i x_i Z_{c_i}\right) T_{cm}}{\sum_i \left(x_i V_{c_i}\right)} \qquad (6)$$

where P_{cm} = pseudo critical pressure

T_{cm} = critical temperature

x_i = molar fraction

Z_{ci} = compressibility factor of component i

V_{ci} = specific volume of component i

R^* = universal gas constant (= 8.3144 kJ/kmol K).

EXAMPLE

For the natural gas used by Groves, the relevant data taken from Tables A1 and A3 (given on pages 222 and 224) are:

i	x_i	V_{c_i} (cm^3/mol)	Z_{c_i}
1	1.498	89.8	0.290
2	1.073	93.9	0.274
3	83.266	99.2	0.288
4	9.608	148.3	0.285
5	3.597	203.0	0.281
6	0.3414	263	0.283
7	0.4581	255	0.274
8	0.0403	303	0.269
9	0.0342	304	0.263
10	0.0046	370	0.264

Therefore:

$$\sum_i (x_i Z_{c_i}) = (1.498 \times 0.290) + (1.073 \times 0.274) + (83.266 \times 0.288) + \ldots$$

$$= 28.701$$

and

$$\sum_i (x_i V_{c_i}) = (1.498 \times 89.8) + (1.073 \times 93.9) + (83.266 \times 99.2) + \ldots\ldots$$

$$= 10881 \text{ cm}^3/\text{mol}$$

$$\Rightarrow P_{cm} = \frac{8.3144 \times 28.701 \times 222.66 \times 10^3}{10881} \text{ kPa}$$

$$= 4888 \text{ kPa.}$$

REFERENCES

ABARBANEL, S., GOTTLIEB, D. & TURKEL, E. (1975). "Difference Schemes with Fourth Order Accuracy for Hyperbolic Equations". SIAM. J. Appl. Math., Vol.29, pp.329-351.

AMES, W.F. (1977). "Numerical Methods for Partial Differential Equations". 2nd edition, published by Nelson & Sons, London.

ANSORGE, R. (1963). "Die Adams-Verfahren als Charakteristiken-verfahren höherer Ordnung zur Lösung von Hyperbolischen Systemen halblinearer Differentialgleichungen". Num. Math., Vol.5, pp.443-460.

ARRISON, N.L., HANCOX, W.T., SULATISKY, M.T. & BANERJEE, S. (1977). "Blowdown of a recirculating loop with heat addition". Paper 61, Thermodynamics and Fluid Mechanics Group Conference, University of Manchester.

ASCHENBRENNER, J. (1937). "Forshung, auf dem Gebiete des Ingenieurwesens", Vol. 8, part 3, p.118.

BAINES, M.J. (1986). "Moving Finite Element Modelling of Compressible Flow". Appl. Numer. Math., Vol.2, No.6, Dec.1986, pp.495-514.

BAKHTAR, F. (1956). "Effects of Wall Friction and Heat Transfer in a Uniform Shock Tube". Ph.D. Thesis, University of Birmingham.

BANERJEE, S. & HANCOX, W.T. (1978). "On the Development of Methods for Analysing Transient Flow-Boiling". Int. J. Multiphase Flow, Vol.4, pp.437-460.

BANNISTER, F.K. & MUCKLOW, G.F. (1948). "Wave Action Following Sudden Release of Compressed Gas from a Cylinder". Proc. I.Mech.E., Vol.159, p.269.

BENDER, E. (1979). "Simulation of Dynamic Gas Flows in Networks Including Control Loops". Computers and Chemical Engineering, Vol.3, Nos.1-4.

BENSON, R.S., GARG, R.D. & WOOLLATT, D. (1964). "A Numerical Solution of Unsteady Flow Problems". Int. J. Mech. Sci., Vol.6, No.1, p.117.

BLASIUS, H. (1911). "The Law of Similarity for Frictional Phenomena". Physikalische Zeitschrift, Vol.12, pp.1175-1177.

BRITTAIN, I. & FAYERS, F.J. (1976). "A Review of U.K. Developments in Thermal-Hydraulic Methods for Loss of Coolant Accidents". Proc. of C.S.N.I. Specialists Meeting on Transient Two-Phase Flow, Toronto Atomic Energy of Canada, Ltd.

BROWN, F.T., MARGOLIS, D.L. & SHAH, R.P. (1969). "Small Amplitude Frequency Behaviour of Fluid Lines with Turbulent Flow". J. Basic Eng., Trans. ASME, Series D, Vol.91, pp.678-692.

CAMERON, I. (1984). "Performance Analysis of Alberta's Pipelines". J. Can. Pet. Technol., Vol.23, No.6, Nov./Dec. 1984, pp.40-43.

CARVER, M.B. (1980). "Pseudo Characterisic Method of Lines Solution of the Conservation Equations". J. Comput. Phys., Vol.35, pp.57-76.

CHABRILLAC, M. (1976). "Development of Numerical Methods for Thermohydraulic Problems in Reactor Safety". Presented at Int. Conference on Transient Two-Phase Flow, Canada, 1976.

CHAUDHRY, M.H. (1978). "Applied Fluid Transients". Published by Van Nostrand Reinhold Company.

CHEESEMAN, A.P. (1970). "Application of Pressure Transient Studies to Long Distance Oil and Gas Transmission Lines". Symposium on Pressure Transients, City University, Paper No.4, November 1970.

CHENG, L.C. & BOWYER, J.M.(Jr) (1978). "TUBE - Transient Compressible Flow Code". Fluid Transients and Acoustics in the Power Industry, 1978.

CHORIN, A.J. (1976). "Random Choice Solution of Hyperbolic Systems". J. Comput. Phys., Vol.22, pp.517-533.

COLEBROOK, C.F. (1938). "Turbulent Flow in Pipes with Particular Reference to the Transition Region between Smooth and Rough Pipe Laws". J. Inst. Civil Eng., Vol.11, pp.133-156.

COLELLA, P. (1982). "Approximate Solution of the Reimann Problem for Real Gases". Preprint, Lawrence Berkeley Laboratory, University of California, May 1982.

COMBES, G. & ZAOUI, J. (1967). "Analyse des Erreurs Introduites par l'utilisation practique de la méthode des caractéristiques dans le calcul des coups de bélier". La Houille Blanche, Vol.22, No.2, pp.195-202.

COURANT, R. & FRIEDRICHS, K.O. (1948). "Supersonic Flow and Shock Waves". Published by Interscience Publishers, New York.

COURANT, R., ISAACSON, E. & REES, M. (1952). "On the Solution of Nonlinear Hyperbolic Differential Equations by Finite Differences". Comm. Pure Appl. Maths., Vol.5, pp.243-255.

CRONJE, J.S., BISHNOI, P.R. & SVRCEK, W.Y. (1980). "Application of the Characteristic Method to Shock Tube Data that Simulate a Gas Pipeline Rupture". Canadian J. of Chem. Eng., Vol.58, No.3, June 1980, pp.289-294.

EARNSHAW, S. (1860) "On the Mathematical Theory of Sound". Phil. Trans., Royal Society, 1860, Vol.150, p.133.

EDWARDS, A.R. & O'BRIEN, T.P. (1970). "Studies of Phenomena Connected with the Depressurization of Water Reactors". J. Brit. Nuclear Energy Soc., Vol.9, p.125.

ELLIOT, J.N. (1968) "RODFLOW. A Program for Studying Transients in a Power Reactor Coolant Circuit". Unpublished AECL Internal Report, Sheridan Park.

ETTER, D.O. & KAY, W.B. (1961). "Critical Properties of Mixtures of Normal Paraffin Hydrocarbons". J. Chem. Eng. Data, Vol.6, p.409.

FALCUS, J.E., FEARNEHOUGH, G.D., JONES, D.G., JUDE, W. & THOMAS, D.B.J. (1986). "Group Sponsored Burst Test to Evaluate Shear Fracture Behaviour for Underwater Application". Offshore Oil and Gas Pipeline Technology, 1986, European Seminar, Paris, France, 28-29 January, 1986.

FINCHAM, A.E. & GOLDWATER, M.H. (1979). "Simulation Models for Gas Transmission Networks". Trans. Inst. Measurement and Control, Vol.1, pp.3-12.

FORSYTHE, G.E. & WASOW, W.R. (1960). "Finite Difference Methods for Partial Differential Equations". Published by John Wiley & Sons, Inc.

FOX, L. (1962). "Numerical Solutions of Ordinary and Partial Differential Equations". Published by Pergamon Press, London.

FOX, P. (1960). "The Solution of Hyperbolic Partial Differential Equations by Difference Methods". in "Mathematical Methods for Digital Computers" Edited by A. Ralston and H.S. Wilf. Published by John Wiley & Sons, Inc., 1960.

GARY, J. (1978). "On Boundary Conditions for Hyperbolic Difference Schemes". J. Comput. Phys., Vol.26, pp.339-351.

GLIMM, J. (1965). "Solutions in the Large for Nonlinear Hyperbolic Systems of Equations". Comm. Pure Appl. Math., Vol.18, p.697.

GODUNOV, S.K. (1959). "A Finite Difference Method for the Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics". Matematicheskii Sbornik, Vol.47, pp.271-290.

GOLDFINCH, M.C. (1984). "Microcomputers Simulate Natural Gas Networks". Oil Gas J., Vol.82, No.37, Sept.10th, 1984, pp.180, 182, 184.

GOLDWATER, M.H. & FINCHAM, A.E. (1980). "Modelling of Gas Supply Systems". in "Modelling of Dynamic Systems", Volume 2, Edited by H. Nicholson. Published by Peter Peregrinus Ltd., on behalf of the Institution of Electrical Engineers, 1980.

GORTON, R.L. (1978). "Application of Finite Difference Methods to Transient Steam Flow Problems in Power Plant Piping Systems". Fluid Transients and Acoustics in the Power Industry, 1978.

GOTTLIEB, D. & TURKEL, E. (1978). "Boundary Conditions for Multistep Finite Difference Methods for Time Dependent Equations". J. Comput. Phys., Vol.26, pp.181-196.

GRIEVES, R.B. & THODOS, G. (1962). "The Critical Temperatures of Multicomponent Hydrocarbon Systems". A.I.Ch.E. Journal, Vol.8, No.4, pp.550-553.

GRIEVES, R.B. & THODOS, G. (1963). "The Critical Pressures of Multicomponent Hydrocarbon Mixtures and the Critical Densities of Binary Hydrocarbon Mixtures". A.I.Ch.E. Journal, Vol.9, No.1, pp.25-30.

GROVES, T.K. (1976). "Measured Velocity of Decompression Waves in Natural Gases in Pipe Lines". A report prepared for Foothills Pipelines (Yukon) Ltd.

GROVES, T.K., BISHNOI, P.R. & WALLBRIDGE, J.M.E. (1978). "Decompression Wave Velocities in Natural Gases in Pipelines". Can. J. Chem. Eng., Vol.56, pp.664-668, December, 1978.

GUY, J.J. (1967). "Computation of Unsteady Gas Glow in Pipe Networks". Proc. of Symposium "Efficient Methods for Practising Chemical Engineers", Symposium Series No.23, London Inst. of Chem. Eng., pp.139-145.

HAALAND, S.E. (1983). "Simple and Explicit Formulae for the Friction Factor in Turbulent Pipe Flow". ASME. J. of Fluids Eng., Vol.105, p.89.

HANCOX, W.T., MATHERS, W.G. & KAWA, D. (1975). "Analysis of Transient Flow-Boiling; Application of the Method of Characteristics". Paper presented at 15th National Heat Transfer Conference, San Francisco, American Inst. of Chem. Eng., August 1975.

HANCOX, W.T. & NICOLL, W.B. (1972). "A Wall Shear Stress Formula for Adiabatic Two-Phase Flow". Westinghouse Canada Ltd., Report (unpublished) 1972.

HARTREE, D.R. (1952). "Some Practical Methods of Using Characteristics in the Calculation of Non-Steady Compressible Flows". Los Alamos Report LA-HU-1, 1952.

HAYES, D.J. & LUX, M.D. (1979). "Full Scale Crack Arrest Test and Arrester Device. Performance for the FLAGS Gas Line". 64th Annual Fall Technical Conference, Society of Pet. Eng., Las Vegas, September 23-26, 1979. SPE 8220.

HEATH, M.J. & BLUNT, J.C. (1969). "Dynamic Simulation Applied to the Design and Control of a Pipeline Network". J. Inst. Gas Eng., Vol.9, No.4, pp.261-279.

HENRY, L.R. (1969). Published discussion with STONER [1969]. Trans. ASME. J. Basic Eng., Vol.91, No.3, September 1969.

ISSA, R.I. & SPALDING, D.B. (1972). "Unsteady One-dimensional Compressible Frictional Flow with Heat Transfer". J. Mech. Eng. Science, Vol.14, pp.365-369.

ISSA, R.I. (1970). "One-dimensional Unsteady Compressible Flow with Friction and Heat Transfer". M.Sc. Thesis, University of London, 1970.

JONES, D.G. & GOUGH, D.W. (1981). "Rich Gas Decompression Behaviour in Pipelines". British Gas report ERS E 293, September 1981.

KAWABE, R. (1982). "Analytical Method for Thermal-Hydraulic Transients in Piping Networks". Nucl. Eng. Des., Vol.73, No.3, December 1982, pp.441-446.

KAY, W.B. (1936). "Density of Hydrocarbon Gases and Vapours". Ind. Eng. Chem., Vol.28, p.1014.

KÖBES, K. (1910). "Die Durchschlagsgeschwindigkeit beir den Luftsauge und Druckluftbremsen, Studien über unstetige Gasbewegungen". Zeitschrift Osterreichischen Ingeniein und Architektenvereines, Vol.62, p.553.

KRIVOSHEIN, B.L., RADCHENKO, V.P., BOBROVSKIY, S.A., DUBINSKIY, A.V. & SIPERSHTEYN, B.I. (1976). "Certain Mathematical Models of Unsteady Gas Flow in Trunk Pipelines". Fluid Mechanics - Soviet Research, Vol.5, No.2, March/April 1976.

LAKSHMINARAYANAN, P.A., JANAKIRAMAN, P.A., BABU, M.K.G. & MURTHY, B.S. (1979). "Finite Difference Scheme for Unsteady Pipe Flows". Int. J. Mech. Sci., Vol.21, No.9, pp.557-566.

LAX, P. & WENDROFF, B. (1960). "Systems of Conservation Laws". Comm. Pure Appl. Math., Vol.13, pp.217-237.

LISTER, M. (1960). "The Numerical Solution of Hyperbolic Partial Differential Equations by the Method of Characteristics". in "Mathematical Methods for Digital Computers", Edited by A.Ralston and H.S. Wilf. Published by John Wiley & Sons, Inc., 1960.

LOMBARD, C.K., OLIGER, J. & YANG, J.Y. (1982). "A Natural Conservative Flux Difference Splitting for the Hyperbolic Systems of Gasdynamics". Lecture Notes in Physics, Vol.170, Springer-Verlag, Berlin, New York, 1982, pp.364-370.

MacCORMACK, R. (1971). "Proceedings of the Second Int. Conference on Numerical Methods in Fluid Dynamics". Lecture Notes in Physics (M. Holt, Ed.), Vol.8, Springer-Verlag, New York, 1971.

MACK, J.E. (1954). "Density Measurement in Shock Tube Flow with the Chrono-interferometer". Lehigh University Inst. of Research Tech., Report No.4, 1954.

MARTIN, C.S., PADMANABHAN, M. & WIGGERT, D.C. (1976). "Pressure Wave Propagation in Two-Phase Bubbly Air-Water Mixtures". Second Int. Conference on Pressure Surges, 1976.

* JONES, D. G. (1988), British Gas Engineering Research Station, Newcastle. Private Communication.

MARTIN, C.S. & CHAUDHRY, M.H. (1983). "Numerical Methods for Fluid Transient Analysis". Applied Mechanics, Bio-engineering and Fluid Engineering Conf., 1983. Published by ASME Fluids Eng. Div., Vol.4, New York, U.S.A.

MATHERS, W.G., ZUZAK, W.W., McDONALD, B.H. & HANCOX, W.T. (1976). "On Finite Difference Solutions to the Transient Flow-Boiling Equations". Paper presented to Committee on the Safety of Nuclear Installations, Specialists Meeting on Transient Two-Phase Flow, Toronto, Ontario, August 1976. Published by Pergamon Press.

MAXEY, W.A., SYLER, F.A. & EIBER, R.J. (1975). "Fracture Propagation Experiments on 48 inch x 0.72 inch Line Pipe" NEB Public Document NPD-137.

MAYFIELD, F.D. (1942). "Critical States of Two-Component Paraffin Systems". Ind. Eng. Chem., Vol.34, p.843.

MEKEBEL, S. & LORAUD, J.C. (1985). "Study of Variable Flow in Natural Gas Pipelines". Int. Chem. Eng., Vol.25, No.2, April 1985, pp.258-265.

MEKEBEL, S. & LORAUD, J.C. (1983). "Une Étude des écoulements variables dan les conduites de transport de gaz naturel". Entropie, Vol.19, No.111, pp.18-25.

MOODY, L.F. (1947). "An Approximate Formula for Pipe Friction Factors". Trans. ASME, Vol.69, p.1005.

MOORE, K.V. & RETTIG, W.H. (1973). "A Computer Program for Transient Thermal-Hydraulic Analysis". Aerojet Nuclear Company Report, ANCR-1127.

MORETTI, G. (1979). "The λ-Scheme". Computers and Fluids, Vol.7, No.3, pp.191-205.

MORTON, K.W. & PARROTT, A.K. (1980). "Generalized Galerkin Methods for First Order Hyperbolic Equations". J. Comput. Phys., Vol.36, pp.249-270.

MULPURU, S.R. (1983). "Flux Splitting Method for the Numerical Simulation of One-dimensional Compressible Flow". Math. Comput. Simul., Vol.25, No.4, August 1983, pp.309-320.

NIESSNER, H. (1980). "Comparison of Different Numerical Methods for Calculating One-dimensional Unsteady Flows". Presented at the Von Karman Institute for Fluid Dynamics, Lecture No.16 from Lecture Series 1980-81, "Unsteady One-dimensional Flows in Complex Networks and Pressurized Vessels", January 14-18, 1980.

NIKURADSE, J. (1933). "Flow Law in Rough Pipes". Forschungshaft VDI, No.361, July/August 1933.

OLIEMANS, R.V.A. (1976). "Two-Phase Flow in Gas Transmission Pipelines". ASME Publication 76-Pet-25.

ORANJE, L., GRAAFF, R. & FAGERLAND, S. (1985). "The Depressurization of a Dense Phase Flow Gas Pipeline". Publication No.B518-9, Published by International Gas Union, 62, rue de Courcelles, 75008, Paris, France.

ORGANICK, E.I. (1953). "Prediction of Critical Temperatures and Critical Pressures of Complex Hydrocarbon Mixtures". Chem. Eng. Progr. Symposium, Series No.6, 49, 81.

OSHER, S. & SOLOMON, F. (1982). "Upwind Difference Schemes for Hyperbolic System of Conservation Laws". Mathematics of Computation, Vol.38, pp.339-377.

OSIADACZ, A. (1984). "Simulation of Transient Gas Flows in Networks". Int. J. Num. Methods in Fluids, 1984, pp.13-23.

PADMANABHAN, M., AMES, W.F., & MARTIN, C.S. (1978). "Numerical Analysis of Pressure Transients in Bubbly Two-Phase Mixtures by Explicit-Implicit Methods". Journal of Engineering Maths., Vol.12, No.1, January 1978.

PANDOLFI, M. (1984). "Contribution to the Numerical Prediction of Unsteady Flows". AIAA Journal, Vol.22, No.5, May 1984, pp.602-610.

PRAUSNITZ, J.M. & GUNN, R.D. (1958). "Volumetric Properties of Non-Polar Gaseous Mixtures". A. I.Ch .E. Journal, Vol.4.

PREMOLI, A. & HANCOX, W.T. (1976). "An experimental investigation of subcooled blowdown with heat addition". Proc. of CSNI Specialists Meeting on Transient Two-Phase Flow, Toronto. Atomic Energy of Canada Ltd., Toronto, 1976.

RACHFORD, H.H. & DUPONT, T. (1974). "A Fast Highly Accurate Means of Modelling Transient Flow in Gas Pipeline Systems by Variational Methods". Soc. Pet. Eng. Journal, Vol.14, pp.165-178.

ROACHE, P.J. (1972). "Computational Fluid Dynamics", Published by Hermosa Publishers, 1972.

ROBERTS, L. (1958). "On the Numerical Solution of the Equations for Spherical Waves of Finite Amplitude II". J. Maths. & Physics, Vol.36, No.4, January 1958.

ROE, P.L. [1] (1981). "Approximate Reimann Solvers, Parameter Vectors and Difference Schemes". J. Comput. Phys., Vol.43, pp.357-372.

ROE, P.L. [2] (1981). "The Use of the Reimann Problem in Finite Difference Scheme". Lecture Notes in Physics, Vol.141, Springer-Verlag, Berlin, New York, 1981, pp.354-359.

ROE, P.L. & PIKE, J. (1984). "Efficient Construction and Utilization of Approximate Reimann Solutions". Report from Royal Aircraft Establishment, Bedford, U.K.

ROTHWELL, A.B. (1981) "Final Report on the Test Program at the Northern Alberta Burst Test Facility". Foothills Pipelines (Yukon) Ltd., Report No.18031, 10 August, 1981.

SCHLICHTING, H. (1965). "Grenzschicht - Theorie". 5th edition, Karlsruhe, Verlag G. Braun, 1965.

SENS, M., JOUVE, Ph. & PELLETIER, R. (1970). "Détection d'une rupture accidentelle de conduite". Paper IGU/C 37-30 Presented at 11th Int. Gas Conference, Moscow. (English translation: British Gas Internal Report LRS T448).

SHACHAM, M. (1980). "Industrial Engineering Chemistry, Fundamentals". May 1980, pp.228-229.

SHOKIN, Y.I. & KOMPAIETS, L.A. (1987). "A Catalogue of the Extra Boundary Conditions for the Difference Schemes Approximating the Hyperbolic Equations". Computers and Fluids, Vol.15, No.2, pp.119-136.

SMITH, R.V. (1956). "Flow of Natural Gas Through Experimental Pipelines and Transmission Lines". U.S. Bureau of Mines, Monograph 9, American Gas Association, New York.

SOD, G.A. (1978). "A Survey of Several Finite Difference Methods for Systems of Nonlinear Hyperbolic Conservation Laws". J. Comput. Phys., Vol.27, pp.1-31.

SPALDING, D.B. (1969). "A Procedure for Calculating the Unsteady One-dimensional Flow of a Compressible Fluid with Allowance for the Effects of Heat Transfer and Friction". Imperial College Report No. UF/TN/D/2, 1969.

SPIVACK, M. (1984). "New Algorithms for the Pipebreak Problem". University of Reading Report of a feasibility study carried out for British Gas, May-September 1984.

STEGER, J.L. & WARMING, R.F. (1981). "Flux Vector Splitting of the Inviscid Gasdynamic Equations". J. Comput. Phys., Vol.40, pp.263-293.

STONER, M.A. (1969). "Analysis and Control of Natural Gas Piping Systems". Trans. ASME, J. Basic Eng., Vol.91, No.3, September 1969.

STREETER, V.L. & [LAI C.] (1963). "Water-Hammer Analysis including Fluid Friction". Trans. ASCE, Vol.128, Paper No.3502.

STREETER, V.L. & WYLIE, E.B. (1970). "Natural Gas Pipeline Transients". Soc. Pet. Eng. Journal, Vol.10, pp.357-364.

STREETER, V.L. (1971). "Unsteady Flow Calculations by Numerical Methods". Paper No.71-WA/FE-13, Presented at Winter Annual Meeting, Washington, D.C., November 1971 of the American Society of Mechanical Engineers.

STREETER, V.L. (1972). "Numerical Methods for Calculation of Transient Flow". 1st Int. Conference on Pressure Surges, Canterbury, 1972.

SWAMEE, P.K. & JAIN, A.K. (1976). "Explicit Equations for Pipe-Flow Problems". J. of Hyd. Div., ASCE, Vol.102, No.HY5, pp.657-664.

TAYLOR, B.A. (1978). "The Flow in Pipelines following Catastrophic Failure". British Gas Internal Report LRS 338.

THORLEY, A.R.D. & TILEY, C.H. (1987). "Unsteady and Transient Flow of Compressible Fluids in Pipelines - A Review of Theoretical and Some Experimental Studies". Int. J. of Heat and Fluid Flow, Vol.8, No.1, March 1987, pp.3-15.

TRIKHA, A.K. (1975). "An efficient method for simulating Frequency Dependent Friction in Transient Liquid Flow". Trans. ASME, J. Fluids Eng., Vol.97, Ser.1, pp.97-105, March 1975.

UHL, A.E. (1965). "Steady Flow in Gas Pipelines". Institute of Gas Technology, Technical Report No.10. American Gas Association Inc., New York, 1965.

VAN DEEN, J.K. & REINTSEMA, S.R. (1983). "Modelling of High Pressure Gas Transmission Lines". Appl. Math. Modelling, Vol.7, August 1983.

VAN GOETHEM, G. (1978). "Variable Domain Finite Element Analysis of Unsteady Compressible Fluid Flow Problems". Proc. of Int. Conference on Finite Element in Water Resources (2nd). Imperial College of Science & Technology, London, July 1978. Published by Pentech Press, London, U.K.

VAN LEER, B. (1979). "Towards the Ultimate Conservative Difference Scheme. A Second-Order Sequel to Godunov's Method". J. Comput. Phys., Vol.32, pp.101-136.

VARDY, A.E. (1976). "The Use of the Method of Characteristics for the solution of Unsteady Flows in Networks". Presented at 2nd Int. Conference on Pressure Surges, London, 1976.

WATHEN, A.J. & BAINES, M.J. (1983). "On the Structure of the Moving Finite Element Equations". University of Reading, Numerical Analysis Report No.5/83.

WATT, C.S., BOLDY, A.P. & HOBBS, J.M. (1980). "Combination of Finite Difference and Finite Element Techniques in Hydraulic Transient Problems". 3rd Int. Conference on Pressure Surges, Vol.1, 1980. Published by BHRA, pp.43-62.

WEIMANN, A. (1978). "Gas Distribution Network Dynamic Modelling and Simulation with Respect to Network Control and Monitoring". Ph.D. Thesis, Munich Technical University, 1978. (English Translation: British Gas Internal Report No. LRS T435).

WILKINSON, J.F., HOLLIDAY, D.V., BATEY, E.H. & HANNAH, K.W. (1965). "Transient Flow in Natural Gas Transmission Systems". Tracor Inc., Published by American Gas Association, New York, 1964.

WILLIAMS, A.C. (1956). "Mass Flow and Velocity in a Shock tube". Lehigh University Inst. of Research, Technical Report No.5, 1956.

* WYLIE, E.B., STREETER, V.L. & STONER, M.A. (1974). "Unsteady-state Natural-Gas Calculations for Complex Pipe Systems". Soc. Pet. Eng. Journal, Trans. AIME, Vol.14, pp.35-43.

WYLIE, E.B. & STREETER, V.L. (1978). "Fluid Transients". Published by McGraw-Hill Inc., 1978.

YING, S.P. & SHAH, V.J. (1978). "Transient Pressure in Boiler Steam Lines". Fluid Transients and Acoustics in the Power Industry. Presented at Winter Annual Meeting, 1978, ASME.

YOW, W. (1971). "Analysis and Control of Transient Flow in Natural Gas Piping Systems". Ph.D. Thesis, Civil Eng. Dept. University of Michigan, 1971.

YOW, W. (1972). "Numerical Error on Natural Gas Transient Calculations". Trans. ASME, Paper 72-FE-26, pp.422-428.

ZALESAK, S.T. (1979). "Fully Multidimensional Flux-corrected Transport Algorithms for Fluids". J. Comput. Phys., Vol.31, pp.335-362.

ZEMANSKY, M.W. (1968). "Heat and Thermodynamics". 5th Edition, Published by McGraw-Hill Book Co., 1968.

ZIELKE, W. (1968). "Frequency-Dependent Friction in Transient Pipe Flow". ASME, J. Basic Eng., March 1968.

ZIGRANG, D.J. & SYLVESTER, N.D. (1982). "Explicit Approximations to the Solution of Colebrook Friction Factor Equation". A. I. Chem. E. Journal, Vol.28, No.3, 1982.

ZIGRANG, D.J. & SYLVESTER, N.D. (1985). "A Review of Explicit Friction Factor Equations". J. Energy Resources Technology, Trans. ASME, Vol.107, No.2, pp.280-283.

* WOODWARD, P. M. & COLELLA, P. (1982). "The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks". Report UCRL86952, Lawrence Liverpool National Laboratory, University of California.