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
# LIST OF CONTENTS

<table>
<thead>
<tr>
<th>LIST OF CONTENTS</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIST OF CONTENTS</td>
<td>1</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>4</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>7</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>8</td>
</tr>
<tr>
<td>NOMENCLATURE</td>
<td>9</td>
</tr>
</tbody>
</table>

## CHAPTER 1: INTRODUCTION

## CHAPTER 2: THEORETICAL DEVELOPMENT OF THE BASIC EQUATIONS

2.1. Introduction  
2.2. Conservation of Mass  
2.3. Conservation of Linear Momentum  
2.4. Conservation of Energy  
2.5. Basic Equations in terms of Pressure, Velocity, and Temperature  
2.6. The Friction Term  
2.7. The Heat Transfer Term  
2.8. The Compressibility Factor

## CHAPTER 3: REVIEW OF THE METHODS OF SOLUTION

3.1. Introduction  
3.2. The Method of Characteristics  
3.3. Explicit Finite Difference Methods  
3.4. Implicit Finite Difference Methods  
3.5. Solution of a Riemann Problem - Random Choice and Flux Difference Splitting Schemes  
3.6. Finite Element Analysis  
3.7. Discussion

## CHAPTER 4: SOLUTION OF THE CHARACTERISTIC EQUATIONS FOR A LINEBREAK SITUATION

4.1. Introduction  
4.2. General Solution for Internal Points  
4.3. Specific Solutions for Various Grid Point Configurations  
4.3.1. Internal point with equi-distant adjacent grid points  
4.3.2. Internal boundary point between two different grid sizes  
4.3.3. Internal boundary point linking two different grid sizes
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4.</td>
<td>Upstream Boundary Condition</td>
<td>94</td>
</tr>
<tr>
<td>4.5.</td>
<td>Downstream Boundary Condition</td>
<td>95</td>
</tr>
<tr>
<td>4.6.</td>
<td>Break Boundary Condition</td>
<td>96</td>
</tr>
<tr>
<td><strong>CHAPTER 5:</strong></td>
<td><strong>COMPUTER MODEL</strong></td>
<td></td>
</tr>
<tr>
<td>5.1.</td>
<td>Introduction</td>
<td>103</td>
</tr>
<tr>
<td>5.2.</td>
<td>Transient Analysis Program</td>
<td>103</td>
</tr>
<tr>
<td>5.2.1.</td>
<td>Main program</td>
<td>104</td>
</tr>
<tr>
<td>5.2.2.</td>
<td>Subroutines STEAD1 and STEAD2</td>
<td>107</td>
</tr>
<tr>
<td>5.2.3.</td>
<td>Subroutines SUB1 to SUB6</td>
<td>109</td>
</tr>
<tr>
<td>5.2.4.</td>
<td>Subroutines BREAK1 to BREAK4</td>
<td>111</td>
</tr>
<tr>
<td>5.2.5.</td>
<td>Subroutines SUBUP and DOWN1</td>
<td>116</td>
</tr>
<tr>
<td>5.3.</td>
<td>Graphics Program</td>
<td>123</td>
</tr>
<tr>
<td><strong>CHAPTER 6:</strong></td>
<td><strong>EXPERIMENTAL DATA</strong></td>
<td></td>
</tr>
<tr>
<td>6.1.</td>
<td>Introduction</td>
<td>124</td>
</tr>
<tr>
<td>6.2.</td>
<td>Review of Laboratory Experiments</td>
<td>124</td>
</tr>
<tr>
<td>6.3.</td>
<td>Review of Full Size Tests</td>
<td>125</td>
</tr>
<tr>
<td>6.4.</td>
<td>Selection of Test Data</td>
<td>128</td>
</tr>
<tr>
<td>6.5.</td>
<td>Preparation of the Data</td>
<td>134</td>
</tr>
<tr>
<td>6.5.1.</td>
<td>Preparation of gas data</td>
<td>140</td>
</tr>
<tr>
<td>6.5.2.</td>
<td>Preparation of system data</td>
<td>142</td>
</tr>
<tr>
<td><strong>CHAPTER 7:</strong></td>
<td><strong>RESULTS</strong></td>
<td></td>
</tr>
<tr>
<td>7.1.</td>
<td>Introduction</td>
<td>147</td>
</tr>
<tr>
<td>7.2.</td>
<td>Groves' Shock Tube Results</td>
<td>147</td>
</tr>
<tr>
<td>7.3.</td>
<td>British Gas Shock Tube Results</td>
<td>148</td>
</tr>
<tr>
<td>7.4.</td>
<td>Foothills Pipelines (Yukon) Ltd. Full Size Results</td>
<td>152</td>
</tr>
<tr>
<td><strong>CHAPTER 8:</strong></td>
<td><strong>DISCUSSION</strong></td>
<td></td>
</tr>
<tr>
<td>8.1.</td>
<td>General Discussion of Results</td>
<td>180</td>
</tr>
<tr>
<td>8.1.1.</td>
<td>Groves' Shock Tube Results</td>
<td>180</td>
</tr>
<tr>
<td>8.1.2.</td>
<td>British Gas Shock Tube Results</td>
<td>182</td>
</tr>
<tr>
<td>8.1.3.</td>
<td>Foothills Pipelines (Yukon) Ltd. Full Size Results</td>
<td>185</td>
</tr>
<tr>
<td>8.2.</td>
<td>Discussion of Errors</td>
<td>188</td>
</tr>
<tr>
<td><strong>CHAPTER 9:</strong></td>
<td><strong>CONCLUSIONS</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>195</td>
</tr>
</tbody>
</table>
CHAPTER 10: FURTHER WORK

10.1. Investigation of the Wavespeed Error 197
10.2. Further Testing of the Present Model 198
10.3. Improvement of the Stability of the Solution 198
10.4. Further Refinement of the Model 198

APPENDICES:

I. Derivation of the Basic Partial Differential Equations, with Pressure, Temperature and Velocity as the Dependent Variables. 200
II. Friction Factor Relationships 206
III. Implementation of Taylor's Theorem for Various Grid Points 212
IV. Derivation of the Particle Velocity of a Rarefaction Wave 218
V. Program Listings 225
VI. Preparation of Gas Data 293

REFERENCES 300
LIST OF FIGURES

1.1. Typical Phase Diagram for Natural Gas
1.2. Pipe Failure Distribution for Natural Gas Lines

2.1. Control Volume Illustrating the Conservation of Mass
2.2. Control Volume Illustrating the Conservation of Linear Momentum
2.3. Control Volume Illustrating the Conservation of Energy
2.4. Heat Transfer into the Pipe
2.5. Heat Transfer into the Pipe - Simplified Model
2.6. Generalized Compressibility Factor Chart

3.1. Two-Dimensional Natural Grid of Characteristics
3.2. Linear Characteristics on an x-t plane
3.3. Characteristics on a Rectangular Grid for Two Dependent Variables
3.4. Characteristics for Three Dependent Variables on a Rectangular Grid
3.5. Possible Problem Areas when Using the Mesh Method of Characteristics
3.6. The Finite Difference Grid Illustrating a Two-step Method
3.7. An x-t Grid for Illustrating Implicit Finite Difference Methods
3.8. Grid Points used in Various Finite Difference Methods
3.9. Weighted Finite Difference Approximations
3.10. Elemental Section for Flux Difference Splitting
3.11. Discontinuity Between Two Half Sections

4.1. Grid Size Variation for Modelling a Linebreak
4.2. Different Internal Grid Point Configurations
4.3. Effect of Flow Reversal on the Characteristics
4.4. Grid Point Linking Two Different Grid Sizes
4.5. Downstream Boundary Condition
4.6. Pressure Drop at the Break
Data Preparation Sheet
Marching Process of Calculation
Linear Momentum Equation for Steady One-dimensional Pipe Flow
Flow Diagrams for STEAD1 and STEAD2
Flow Diagrams for SUB1, SUB2, and SUB5
Flow Diagrams for SUB3, SUB4, and SUB6
Break Point Prior to Rupture
Break Point After Rupture
Flow Diagram for BREAK1
Flow Diagram for BREAK3
Flow Diagram for BREAK4
Flow Diagram for SUBUP
Flow Diagram for DOWN1

Shock Tube Used by Groves et al.
British Gas Shock Tube
Test Sections Used by Foothills Pipelines (Yukon) Ltd.

P x w Graph for Groves' Shock Tube Test - Methane
P x w Graph for Groves' Shock Tube Test - Argon
P x w Graph for Groves' Shock Tube Test - Natural Gas
P x t Graph for British Gas Shock Tube Test 1)(f = 0.018
P x t Graph for British Gas Shock Tube Test 2)]St = 0.0027)
P x t Graph for British Gas Shock Tube Test 4)]f = 0.018
P x t Graph for British Gas Shock Tube Test 5]St = 0.0027)
P x t Graph for British Gas Shock Tube Test 7)]f = 0.018
P x t Graph for British Gas Shock Tube Test 8)]St = 0.0027)
P x t Graph for British Gas Shock Tube Test 1)]f = 0.01
P x t Graph for British Gas Shock Tube Test 2)]St = 0.0027)
P x t Graph for British Gas Shock Tube Test 4)]f = 0.03
P x t Graph for British Gas Shock Tube Test 5)]St = 0.0027)
P x t Graph for British Gas Shock Tube Test 7)]f = 0.03
P x t Graph for British Gas Shock Tube Test 8)]St = 0.0027)
P x t Graph for Foothills Test Results NABTF1 (West)
P x t Graph for Foothills Test Results NABTF1 (East)
P x w Graph for Foothills Test Results NABTF1
P x t Graph for Foothills Test Results NABTF3 (West)
P x t Graph for Foothills Test Results NABTF3 (East)
7.21. P x w Graph for Foothills Test Results NABTF3
7.22. P x t Graph for Foothills Test Results NABTF4 (West)
7.23. P x w Graph for Foothills Test Results NABTF4 (East)
7.24. P x w Graph for Foothills Test Results NABTF4
7.25. P x t Graph for Foothills Test Results NABTF5 (West)
7.26. P x t Graph for Foothills Test Results NABTF5 (East)
7.27. P x w Graph for Foothills Test Results NABTF5

8.1. Approximation of dp/dΦ on a Finite Grid
8.2. Pipe Rupture in a Full Size Pipe
8.3. Variation of the Specific Heat of Methane

A.1. Model of Fluid Movement in a Tube
A.2. Coefficient Charts for use in the Method of Grieves and Thodos
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:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Cross-sectional area of pipe</td>
<td>m²</td>
</tr>
<tr>
<td>a_s</td>
<td>Isentropic wavespeed</td>
<td>m/s</td>
</tr>
<tr>
<td>C_p</td>
<td>Specific heat at constant pressure</td>
<td>J/kg K</td>
</tr>
<tr>
<td>C_v</td>
<td>Specific heat at constant volume</td>
<td>J/kg K</td>
</tr>
<tr>
<td>d</td>
<td>Diameter of pipe</td>
<td>m</td>
</tr>
<tr>
<td>e</td>
<td>Specific internal energy</td>
<td>J/kg</td>
</tr>
<tr>
<td>f</td>
<td>Darcy friction factor</td>
<td>-</td>
</tr>
<tr>
<td>g</td>
<td>Acceleration due to gravity</td>
<td>m/s²</td>
</tr>
<tr>
<td>h</td>
<td>Specific enthalpy</td>
<td>J/kg</td>
</tr>
<tr>
<td>j</td>
<td>Rectangular coordinates used in explicit finite difference methods</td>
<td>-</td>
</tr>
<tr>
<td>P</td>
<td>Pressure</td>
<td>Pa</td>
</tr>
<tr>
<td>Pr</td>
<td>Prandtl number</td>
<td>-</td>
</tr>
<tr>
<td>Q</td>
<td>Heat transfer rate per unit volume</td>
<td>J/m³s</td>
</tr>
<tr>
<td>R</td>
<td>Specific gas constant</td>
<td>J/kg</td>
</tr>
<tr>
<td>Re</td>
<td>Reynolds number</td>
<td>-</td>
</tr>
<tr>
<td>s</td>
<td>Specific entropy</td>
<td>J/kg K</td>
</tr>
<tr>
<td>St</td>
<td>Stanton number</td>
<td>-</td>
</tr>
<tr>
<td>T</td>
<td>Temperature of the gas</td>
<td>K</td>
</tr>
<tr>
<td>t</td>
<td>Time</td>
<td>s</td>
</tr>
<tr>
<td>u</td>
<td>Velocity of the gas</td>
<td>m/s</td>
</tr>
<tr>
<td>w</td>
<td>Frictional force per unit length of pipe</td>
<td>N/m</td>
</tr>
<tr>
<td>x</td>
<td>Distance along the pipe</td>
<td>m</td>
</tr>
<tr>
<td>x'</td>
<td>Thermodynamic quality or dryness fraction</td>
<td>-</td>
</tr>
<tr>
<td>z</td>
<td>Gas Compressibility factor</td>
<td>-</td>
</tr>
</tbody>
</table>

**Greek Symbols**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>θ</td>
<td>Angle of inclination of pipe to the horizontal</td>
<td>Rad</td>
</tr>
<tr>
<td>ρ</td>
<td>Mean density of the gas</td>
<td>kg/m³</td>
</tr>
<tr>
<td>Ω</td>
<td>Heat flow into the pipe per unit length of pipe and per unit time</td>
<td>J/ms</td>
</tr>
<tr>
<td>ω</td>
<td>actual wave propagation speed selective to the pipe</td>
<td>m/s</td>
</tr>
</tbody>
</table>
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](image)

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](image)

<table>
<thead>
<tr>
<th>Cause</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internal Corrosion</td>
<td>10%</td>
</tr>
<tr>
<td>External Corrosion</td>
<td>17%</td>
</tr>
<tr>
<td>Joint Failure</td>
<td>7%</td>
</tr>
<tr>
<td>Weld Failure</td>
<td>6%</td>
</tr>
<tr>
<td>Third Party Damage</td>
<td>35%</td>
</tr>
<tr>
<td>Other Causes (e.g. operator error or unknown cause)</td>
<td>17%</td>
</tr>
</tbody>
</table>

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

\[
A(\rho + \frac{\partial \rho}{\partial x} \, dx) \left( u + \frac{\partial u}{\partial x} \, dx \right) - \rho u A = -\frac{\partial}{\partial t} (\rho A dx)
\]

Neglecting very small terms:

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

\[
\therefore \frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x} (\rho u) = 0
\]  
(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:

\[
PA - (P + \frac{\partial P}{\partial x} \, dx) \, A \cdot dx - W \, dx - \rho \, A \, g \, \sin \theta \, dx = \frac{\partial}{\partial t} (\rho \, A \, u \, dx) + \frac{\partial}{\partial x} (\rho \, A \, u^2 \, dx)
\]

Dividing through by \(A \cdot dx:-\)

\[
\frac{\partial}{\partial t}(\rho u) + \frac{\partial}{\partial x}(\rho u^2) + \frac{\partial P}{\partial x} + \frac{w}{A} + \rho \, g \, \sin \theta = 0
\]

\[
u \left( \frac{\partial}{\partial t} (\rho u) \right) + \rho \frac{\partial u}{\partial t} + \rho u \frac{\partial u}{\partial x} + \frac{\partial P}{\partial x} + \frac{w}{A} + \rho \, g \, \sin \theta = 0
\]
But from equation (2.1):

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

Therefore:

\[ \rho \frac{\partial u}{\partial t} + \rho u \frac{\partial u}{\partial x} + \frac{\partial P}{\partial x} = -\frac{W}{A} - \rho g \sin \theta \] \hspace{1cm} (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](image)
\[
\frac{\partial}{\partial x} \left\{ (h + \frac{u^2}{2}) \rho u \right\} + \frac{\partial}{\partial t} \left\{ (e + \frac{u^2}{2}) \rho A \right\} + \rho \text{Augsin}\theta = \Omega
\]

Dividing through by \( A \):

\[
\frac{\partial}{\partial x} (\rho u h) + \frac{\partial}{\partial x} \left( \rho \frac{u^3}{2} \right) + \frac{\partial}{\partial t} \left( \rho \frac{u^2}{2} \right) + \frac{\partial}{\partial t} (\rho e) + \rho \text{Augsin}\theta = \frac{\Omega}{A}
\]

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:

\[
\frac{\partial}{\partial x} (h\rho u) + \frac{\partial}{\partial t} (h\rho) + \frac{\partial}{\partial x} \left( \rho \frac{u^3}{2} \right) + \frac{\partial}{\partial t} \left( \rho \frac{u^2}{2} \right) - \frac{\partial p}{\partial t} + \rho \text{Augsin}\theta = \frac{\Omega}{A}
\]

Factorizing out:

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

\[
+ \rho u \frac{\partial u}{\partial t} - \frac{\partial p}{\partial t} = \frac{\Omega}{A} - \rho \text{Augsin}\theta
\]

But from equation (2.1):

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

And from equation (2.2):

\[
u \left\{ \rho \frac{\partial u}{\partial t} + \rho u \frac{\partial u}{\partial x} \right\} = - \frac{\omega u}{A} - \rho \text{Augsin}\theta - u \frac{\partial p}{\partial x}
\]

Therefore:

\[
\frac{\partial h}{\partial t} + \rho u \frac{\partial h}{\partial x} - \frac{\partial p}{\partial t} - u \frac{\partial p}{\partial x} = \frac{\Omega + \omega u}{A}
\]

(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]:

\[ dh = C_p dT + \left[ \frac{T}{\rho} \left( \frac{\partial \rho}{\partial T} \right) + 1 \right] \frac{d\rho}{\rho}, \]

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

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

\[ \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \frac{1}{\rho} \frac{\partial P}{\partial x} = -\frac{W}{\rho} - g \sin \theta \quad (2.5) \]

\[ \frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + \frac{a^2}{C_p} \left( 1 + \frac{T}{z} \left( \frac{\partial T}{\partial P} \right) \right) \frac{\partial u}{\partial x} = \frac{a^2}{C_p} \left( 1 - \frac{P}{z} \left( \frac{\partial P}{\partial T} \right) \right) \frac{\partial T}{\partial x} + \frac{\partial}{\partial x} \left( \frac{\Omega + Wu}{\rho} \right) \quad (2.6) \]

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:

\[ W = \frac{A}{d} \varphi f \frac{ulu}{2} \]  \hspace{1cm} (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 $\Omega$ 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:

$$Q = 2\pi r_1 h (T_1 - T) = 2\pi k \frac{(T_{w2} - T_{w1})}{2\ln (r_2 / r_1)} = 2\pi r_2 h_a (T_A - T_{w2})$$

where $h = $ convective heat transfer coefficient of the boundary layer

$k = $ thermal conductivity of the pipe wall

$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](image-url)
Therefore the heat transfer may be defined as:

\[ \Omega = \pi d (T_w - T) \]  \hspace{1cm} (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,

\[ Nu = \frac{h d}{k} , \]

\[ Pr = \frac{\nu}{\alpha} , \] where \( \alpha = \frac{k}{\rho C_p} \) and \( \nu = \frac{\mu}{\rho} \),

and \( Re = \frac{\rho u d}{\mu} \)

Therefore,

\[ St = \frac{(h d/k)}{(\mu C_p/k)(\rho u d/\mu)} \]

\[ = \frac{h}{\rho u C_p} \]

Substituting this back into equation (2.8):

\[ \Omega = \pi d \rho u C_p St(T_w - T) \]  \hspace{1cm} (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:

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

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:-

1) 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'.

![Generalized Compressibility Factor Chart](image_url)

**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, \(v \approx \frac{1}{c} \).

At the critical point (subscript 'c'):

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

Therefore:

\[
z = \frac{v' - 27/64}{v' - \frac{1}{8} \frac{T_c}{v' r}}
\]

where \(v' = v \cdot \frac{P_c}{RT_c} = \frac{1}{\phi} \frac{P_c}{RT_c}\)

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]**

\[
P = \frac{RT}{v - b} \cdot \exp\left(\frac{-a}{RT_v}\right)
\]

where \(a\) and \(b\) are constants for each gas.

At the critical point:

\[
a = \frac{4R^2 T^2_c}{P_c (\exp)^2} \quad \text{and} \quad b = \frac{RT_c}{P_c (\exp)^2}
\]
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]

\[ P = \frac{RT}{v - b} - \frac{a}{Tv^2} \] where \( a \) and \( b \) are constants for each gas

At the critical point:

\[ a = \frac{27}{64} \frac{R^2 T^3 c}{P_c} \quad \text{and} \quad b = \frac{1}{8} \frac{RT_c}{P_c} \]

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:

\[ z = 1 + \left[ \frac{9}{128} - \frac{27/64}{T^2_r} \right] \frac{1}{v'_r} + \left[ \frac{9}{128} \right]^2 \frac{1}{v'_r^2} + \left[ \frac{9}{128} \right]^3 \frac{1}{v'_r^3} + \ldots \]
In terms of the reduced pressure and temperature only (neglecting higher order terms):-

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

$$= 1 + \left\{ \frac{9T_C}{128T} - \frac{27T^3_C}{64T^4} \right\} \frac{P}{P_c} \quad (2.10)$$

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:-

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

$$\left( \frac{\partial z}{\partial P} \right)_T = \left\{ \frac{9T_C}{128T} - \frac{27T^3_C}{64T^4} \right\} \frac{1}{P_c} \quad (2.12)$$

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:

\[ \begin{bmatrix} \frac{u_t}{A} \end{bmatrix} + \begin{bmatrix} \frac{u_x}{D} \end{bmatrix} = 0 \]  

(3.1)
where subscripts \( t \) and \( x \) denote partial derivatives with respect to time and distances, and where

\[
U = \begin{bmatrix}
  P \\
  u \\
  T
\end{bmatrix}
\]

\[
\Delta = \begin{bmatrix}
  u & \frac{\partial a_s^2}{\partial x} & 0 \\
  1/\rho & u & 0 \\
  0 & \frac{a_s^2}{C_p} \left(1 + \frac{T}{z} \frac{\partial z}{\partial T} \right) & u
\end{bmatrix}
\]

\[
d = \begin{bmatrix}
  -\frac{a_s^2}{C_p T} \left(1 + \frac{T}{z} \frac{\partial z}{\partial T} \right) & \frac{\Omega + \bar{w} u}{A} \\
  \frac{\bar{w}}{A \rho} + g \sin \theta \\
  -\frac{a_s^2}{C_p p} \left(1 - \frac{p}{z} \frac{\partial z}{\partial p} \right) T & \frac{\Omega + \bar{w} u}{A}
\end{bmatrix}
\]

The eigenvalues \( \lambda \) of matrix \( \Delta \) 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 \( T \) such that:

\[
T \Delta = \lambda \ I \ T \tag{3.2}
\]

Then the characteristic equations are given by:

\[
T \frac{d}{dt}(y) + T d = 0 \tag{3.3}
\]
Let $T = \begin{bmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{bmatrix}$

Solving equation (3.2):-

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

From the above:

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

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

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

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

$\Rightarrow t_{22} = \varphi a_s t_{21}$

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

$\Rightarrow t_{32} = -\varphi a_s t_{31}$
Therefore the transformation matrix may be written:

\[
T = \begin{bmatrix}
-\frac{1}{\varphi_{Cp}}(1 + \frac{T_z}{P_z} \frac{\partial z}{\partial T}) & 0 & 1 \\
\frac{1}{\varphi_{as}} & 1 & 0 \\
\frac{1}{\varphi_{as}} & 1 & 0
\end{bmatrix}
\]

and

\[
T \mathbf{d} = \begin{bmatrix}
\frac{a_s^2}{\varphi_{Cp} T}(1 + \frac{T_z}{P_z} \frac{\partial z}{\partial T})^2 \left(\frac{\Omega + W_u}{A}\right) - \frac{a_s^2}{\varphi_{Cp} T}(1 - \frac{T_z}{P_z} \frac{\partial z}{\partial T})^2 \left(\frac{\Omega + W_u}{A}\right) \\
- \frac{a_s}{\varphi_{Cp} T}(1 + \frac{T_z}{P_z} \frac{\partial z}{\partial T}) \left(\frac{\Omega + W_u}{A}\right) + \frac{W}{A\varphi} + gsine \\
- \frac{a_s}{\varphi_{Cp} T}(1 + \frac{T_z}{P_z} \frac{\partial z}{\partial T}) \left(\frac{\Omega + W_u}{A}\right) + \frac{W}{A\varphi} + gsine \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
- \frac{1}{\varphi_{Cp} T} \frac{\partial z}{\partial T} \left(\frac{\Omega + W_u}{A}\right) \\
- \frac{a_s}{\varphi_{Cp} T}(1 + \frac{T_z}{P_z} \frac{\partial z}{\partial T}) \left(\frac{\Omega + W_u}{A}\right) + \frac{W}{A\varphi} + gsine \\
\end{bmatrix}
\]

Hence, the characteristic equations are:

Along \(\frac{dx}{dt} = u\):

\[
- \frac{1}{\varphi_{Cp}}(1 + \frac{T_z}{P_z} \frac{\partial z}{\partial T}) \frac{dP}{dt} + \frac{dT}{dt} - \frac{1}{\varphi_{Cp} A} \frac{\Omega + W_u}{A} = 0 \quad (3.4)
\]
Along $\frac{dx}{dt} = u + a_s$:

$$\frac{1}{\rho a_s} \frac{dP}{dt} + \frac{du}{dt} - \frac{a_s}{\rho C_p T} (1 + \frac{T}{\frac{z}{2} \frac{\partial z}{\partial T} P})(\frac{\Omega + \bar{W} u}{A}) + \frac{W}{A P} + g \sin \theta = 0 \quad (3.5)$$

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

$$-\frac{1}{\rho a_s} \frac{dP}{dt} + \frac{du}{dt} + \frac{a_s}{\rho C_p T} (1 + \frac{T}{\frac{z}{2} \frac{\partial z}{\partial T} P})(\frac{\Omega + \bar{W} u}{A}) + \frac{W}{A P} + g \sin \theta = 0 \quad (3.6)$$

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:

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

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

\[
(u_B - u_{A1}) + \frac{1}{\rho a_{A1}}(P_B - P_{A1}) + (\frac{\dot{V}}{A\rho} + g\sin\theta)_{A1}(t_B - t_{A1}) = 0 \tag{3.8}
\]

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

Similarly for the \( C^- \) characteristics:

\[
(u_B - u_{A2}) - \frac{1}{\rho a_{A2}}(P_B - P_{A2}) + (\frac{\dot{V}}{A\rho} + g\sin\theta)_{A2}(t_B - t_{A2}) = 0 \tag{3.10}
\]

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.

\[
\int_{x_0}^{x_1} f(x) dx \approx \frac{h}{2}(f(x_0) + f(x_1))(x_1 - x_0)
\]

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 $\Delta x$ and $\Delta t$: 

$$\frac{\Delta t}{\Delta x} \leq \frac{1}{|u| + a_s}$$  \,(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, \( \Delta t \), 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](image-url)
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.

![Figure 3.5. Possible Problem areas when using the Mesh Method of Characteristics](image)

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.
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, $\alpha$. 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 $\alpha^2$ which increased the permissible time step by a factor of $\alpha$. The choice of $\alpha$ 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 \((P, u \text{ and } T)\) along the directions of the independent variables \((x \text{ and } t)\). This produces the solution values at evenly spaced points in the physical plane. The finite difference grid is shown in Figure 3.6.

\[ \text{Figure 3.6. The Finite Difference Grid illustrating a two-step method} \]

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:

\[ \frac{\partial}{\partial t} (A) + \frac{\partial}{\partial x} (B) = C \quad (3.13) \]

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**
\[
\frac{\partial}{\partial t} (\rho) + \frac{\partial}{\partial x} (\rho u) = 0
\]  

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

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:-

\[
A(i,j+1) = A(i,j) - \frac{\Delta t}{2\Delta x} (B(i+1,j) - B(i-1,j))
\]  

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

\[
A(i,j+1) = A(i,j) - \frac{\Delta t}{2\Delta x} (B(i+1,j) - B(i-1,j)) + \frac{\omega_f}{4} (A(i+1,j) - 2A(i,j) + A(i-1,j))
\]  

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:

\[
A_{i,j+1} = A_{i,j} - \frac{\Delta t}{2\Delta x}(B_{i+1,j} - B_{i-1,j}) \\
+ \frac{1}{4} \frac{\Delta t^2}{\Delta x^2} \left( \frac{\partial B}{\partial A}(i+1,j) + \frac{\partial B}{\partial A}(i,j) \right) (B_{i+1,j} - B_{i,j}) \\
- \left( \frac{\partial B}{\partial A}(i,j) + \frac{\partial B}{\partial A}(i-1,j) \right) (B_{i,j} - B_{i-1,j}) \tag{3.19}
\]

This method has the disadvantage that additional computing time is required to evaluate \( \frac{\partial B}{\partial A} \) 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: \[ A_{(i+\frac{1}{2},j+\frac{1}{2})} = \frac{1}{2} [A(i+1,j) + A(i,j)] - \frac{\Delta t}{2\Delta x} \left[ B(i+1,j) - B(i,j) \right] \]

\[ + \frac{\Delta t}{2} [C(i+1,j) + C(i,j)] + O(\Delta x^2, \Delta t) \]  \hspace{1cm} (3.20)

SECOND STEP: \[ A(i,j+1) = \frac{\Delta t}{\Delta x} \left[ B(i+\frac{1}{2},j+\frac{1}{2}) - B(i-\frac{1}{2},j+\frac{1}{2}) \right] \]

\[ + \Delta t[C(i+\frac{1}{2},j+\frac{1}{2}) + C(i-\frac{1}{2},j+\frac{1}{2})] + O(\Delta x^2, \Delta t^2) \]  \hspace{1cm} (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 \[ \overline{A}_{(i,j+1)} = \frac{\Delta t}{\Delta x} \left[ B(i+1,j) - B(i,j) \right] \] (3.22)

SECOND STEP \[ A(i,j+1) = \frac{1}{2} \left\{ A(i,j) + \overline{A}(i,j+1) \right\} - \frac{\Delta t}{2\Delta x} \left[ B(i,j+1) - B(i-1,j+1) \right] \] (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 \( C \)
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.

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

If the Courant number ($\alpha_c$) is defined as:

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

then this stability criterion can be given by:

$$\alpha_c \leq 1$$

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

$$\alpha_c \leq \sqrt{\frac{\omega_f}{2}}$$
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 \( \Phi \) at point \( X \) is denoted by \( \Phi_{c1} \).

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:-

\[
\frac{\partial}{\partial t} (A) + \frac{\partial}{\partial x} (B) = C
\] (3.13)
the fully implicit finite difference approximation for the point \((c, l)\)
can be written:

\[
\frac{A_{c1} - A_{c0}}{\Delta t} + \frac{B_{d1} - B_{d0}}{2\Delta x} = C_{c1}
\]  

(3.25)

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

(a) **The Fully Implicit Method.**

```
  b1    c1    d1
     \   \   / \
     \ / \ / \
    c0  \ / \ /
```

(b) **The Crank-Nicolson Method**

```
  b1    c1    d1
     \   \   / \
     \ / \ / \
    b0  \ / \ /
```

(c) **The Centred Difference Method**

```
  c1    d1
     / \ / \
    c0  \ / \ /
```

(d) **The Characteristic Finite Difference Method**

```
  b1    c1    c1    d1
    \   \ positive \lambda / \
     \ / \ \           / \ \
    c0  \ / \ negative \lambda / \ /
```

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:

\[
\frac{A_{c_1} - A_{c_0}}{\Delta t} + \frac{(B_{d_0} - B_{b_0}) + (B_{d_1} - B_{b_1})}{4\Delta x} = C_{c_1}
\] (3.26)

Figure 3.8(b) gives the nodal plan for this method. Guy [1967] and Heath and Blunt [1969J 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 research 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 \(\partial/\partial x (\rho u^2)\): \(\partial/\partial t (\rho u)\): \(\partial \rho/\partial x\) 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:

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

(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:

\[
\frac{\$ (A_{d1} - A_{do}) + (1-\$)(A_{c1} - A_{co})}{\Delta t} + \frac{\$ (B_{d1} - B_{c1}) + (1-\$)(B_{do} - B_{co})}{\Delta x} = (1-\$)(\$C_{do} + (1-\$)C_{co}) + \$ (\$C_{d1} + (1-\$)C_{c1})
\]

(3.23)

where \$ and \$ are the weighting factors

\[0 < \$ < 1, \quad 0 < \$ < 1\]
(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:

\[ T \frac{\partial \mathbf{u}}{\partial t} + \Lambda T \frac{\partial \mathbf{u}}{\partial x} = \mathbf{D} \]  \hspace{1cm} (3.29)

where \( T \) is the transformation matrix defined in equation (3.2), \( \mathbf{u} \) is the column vector of the dependent variables, \( \Lambda \) is the diagonal matrix of the characteristic directions and \( \mathbf{D} \) is equal to \(-T \, \mathbf{d}\) where \( \mathbf{d} \) is defined in equation (3.1). Hence the characteristic form of the basic equations (2.4) to (2.6) is given by:
The difference approximations at point (c, l) of Figure 3.7, for equation (3.29) may be written:

\[ T = \begin{bmatrix}
-1 & T \left( \frac{\partial z}{\partial T} \right) & 0 & 1 \\
\frac{1}{\rho a_s} & 1 & 0 & \ \\
\frac{1}{\rho a_s} & 1 & 0 & \\
\end{bmatrix} \]

\[ u = \begin{bmatrix} p \\ u \\ T \end{bmatrix}, \quad \Lambda = \begin{bmatrix} u & 0 & 0 \\ 0 & u+a_s & 0 \\ 0 & 0 & u-a_s \end{bmatrix} \]

\[ D = \begin{bmatrix}
\frac{1}{C_p C_T^2} \left( \frac{\Omega + W u}{A} \right) \\
\frac{a_s}{C_p C_T^2} \left( 1 + T \left( \frac{\partial z}{\partial T} \right) \right) \left( \frac{\Omega + W u}{A} \right) - \frac{W}{A^2} - g \sin \theta \\
\frac{-a_s}{C_p C_T^2} \left( 1 + T \left( \frac{\partial z}{\partial T} \right) \right) \left( \frac{\Omega + W u}{A} \right) - \frac{W}{A^2} - g \sin \theta \\
\end{bmatrix} \]

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

\[ T_{co} \left( \frac{u_{c1} - u_{co}}{\Delta t} \right) + A_{co} T_{co} \left( \frac{u_{c1} - u_{b1}}{\Delta x} \right) = D_{co} \quad (3.30) \]

\[ T_{co} \left( \frac{u_{c1} - u_{co}}{\Delta t} \right) + A_{co} T_{co} \left( \frac{u_{d1} - u_{c1}}{\Delta x} \right) = D_{co} \quad (3.31) \]

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:

\[ M_{bc} u_{b1} + M_{cc} u_{c1} + M_{cd} u_{d1} = N_c \]
where

\[ M_{bc} = -\Gamma \Delta c T_c \]

\[ M_{cc} = ((I - \Gamma)(\Delta x) + \Gamma(\Delta x))_{\Delta t cd} - (I - 2\Gamma)\Delta_c T_c \]

\[ M_{cd} = (I - \Gamma)\Delta c T_c \]

\[ N_c = ((I - \Gamma)(\Delta x)_{cd} + \Gamma(\Delta x)_{bc})T_c + ((I - \Gamma)(\Delta x)_{cd} + \Gamma(\Delta x)_{bc})T_c \mu_{co} \]

For subsonic flow:-

\[ \Gamma = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \]

if flow is from left to right

\[ \Gamma = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \]

if flow is from right to left.

The above definitions for \( M \) and \( N \) 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 \( D = 0 \)) as:
\[ U_t = (f(u))_x \] (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 \( t = (n + \frac{1}{2})k \) and at points \( x = \pm ih \) and \( x = \pm (i + \frac{1}{2})h \) where \( n \) and \( i \) are integers. In order to do this the near constant initial data is replaced by discontinuous data:

\[
\begin{align*}
  y(x,0) &= u^n_{i+1} \quad \text{for } x > 0 \\
  y(x,0) &= u^n_i \quad \text{for } x < 0
\end{align*}
\] (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 \( \varepsilon_i \) equidistributed over the interval \((-\infty, \infty)\), with values \( \varepsilon_i \), and using this defined the solution of the Riemann problem at the point \((\varepsilon_i^* h, k/2)\) to be \( Y(\varepsilon_i^* h, k/2) \). This value was then allotted to \( u_{i+\frac{1}{2}}^{n+\frac{1}{2}}\) 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:
\[ \frac{\partial}{\partial t} (A) + \frac{\partial}{\partial x} (B) = C \]  

(3.13)

where, for the basic equations

\[
A = \begin{bmatrix}
\rho \\
\rho u \\
u^2 \\
(e + \frac{1}{2})\rho
\end{bmatrix}
\]

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

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

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 \( t_0 \) 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 $t_0$, let there be uniform flow $B_i$ in the interval $(x_i+\frac{1}{2} - x_i)$ and uniform flow $B_{i+1}$ in the interval $(x_{i+1} - x_i+\frac{1}{2})$. A discontinuity ($\Delta_iA$ and/or $\Delta_iB$) separates the two half intervals at the centre $(x_{i+\frac{1}{2}})$. 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 \cite{Roe1981} 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 than 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 \( u \):

\[
W = \frac{A}{d} \cdot f \cdot \frac{P}{RT\zeta} \frac{u|u|}{2}
\]

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:

\[
\overline{W} = \frac{W_1 + W_2}{2}
\]

\[
= W_1 + \left[ \frac{W_2 - W_1}{2} \right]
\]
This equation was substituted into the finite difference form of the characteristic equations and the following set of equations was produced.

Along \( \text{dx/dt} = u \):

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

Along \( \text{dx/dt} = u + a_s \):

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

Along \( \text{dx/dt} = u - a_s \):

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

In these equations, \( \Delta t \) 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:-

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

where:

\[
A(1) = \left[ -\frac{1}{\rho C_p} \left\{ 1 + \frac{T}{z} \left( \frac{\partial T}{\partial T} \right)_p \right\} - \frac{\Delta t \Delta W}{2\rho C_p \rho} \right]_Q
\]
\[
A(2) = \left[ 1 + \frac{\Delta t \Delta W}{2\rho C_p \rho} \right]_Q
\]
\[
A(3) = \left[ -\frac{\Delta t \Delta W}{\rho C_p \rho} \right]_Q
\]
\[
A(4) = \left[ \frac{1}{\rho \alpha_s} - \frac{a_s \Delta t}{\rho C_p} \left\{ 1 + \frac{T}{z} \left( \frac{\partial T}{\partial T} \right)_p \right\} \frac{W_u}{2\rho \rho} + \frac{W \Delta t}{2\rho \rho} \right]_R
\]
\[
A(5) = \left[ \frac{a_s \Delta t \Delta W}{2\rho C_p \rho} \left\{ 1 + \frac{T}{z} \left( \frac{\partial T}{\partial T} \right)_p \right\} - \frac{W \Delta t}{2\rho \rho} \right]_R
\]
\[
A(6) = \left[ 1 - \frac{a_s \Delta t \Delta W}{\rho C_p \rho} \left\{ 1 + \frac{T}{z} \left( \frac{\partial T}{\partial T} \right)_p \right\} + \frac{W \Delta t}{\rho \rho} \right]_R
\]
\[
A(7) = \left[ -\frac{1}{\rho \alpha_s} + \frac{a_s \Delta t \Delta W}{2\rho C_p \rho} \left\{ 1 + \frac{T}{z} \left( \frac{\partial T}{\partial T} \right)_p \right\} + \frac{W \Delta t}{2\rho \rho} \right]_S
\]
\[
A(8) = \left[ -\frac{a_s \Delta t \Delta W}{2\rho C_p \rho} \left\{ 1 + \frac{T}{z} \left( \frac{\partial T}{\partial T} \right)_p \right\} - \frac{W \Delta t}{2\rho \rho} \right]_S
\]
\[
A(9) = \left[ 1 + \frac{a_s \Delta t \Delta W}{\rho C_p \rho} \left\{ 1 + \frac{T}{z} \left( \frac{\partial T}{\partial T} \right)_p \right\} + \frac{W \Delta t}{\rho \rho} \right]_S
\]
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:

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

\[
\text{posR} = 2\Delta t/(1/(u(i) + a_s(i)) + 1/(u(i-1) + a_s(i-1)))
\]

\[
\text{posS} = 2\Delta t/(1/(a_s(i) - u(i)) + 1/(a_s(i+1) - u(i+1)))
\]

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 \( P, T, u, z (\partial z/\partial T) \), \( \varphi, a_s, \Omega \) and \( W \) at points Q, R and S were then calculated by linear interpolation:
\[ P_Q = \frac{\text{posQ}}{\Delta x} \cdot P(i-1) + (1 - \frac{\text{posQ}}{\Delta x}) \cdot P(i) \]

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

\[ P_S = \frac{\text{posS}}{\Delta x} \cdot P(i+1) + (1 - \frac{\text{posS}}{\Delta x}) \cdot P(i) \]

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 \( z \), \( (\partial z/\partial T)p \), \( \rho \), \( a_s \), \( Q \) 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:
\[ posQ = 2\Delta t/(1/u_Q + 1/u_2(i)) \text{ assuming } u_Q \text{ and } u_2(i) \text{ do not equal zero.} \]

\[ posR = 2\Delta t/(1/(u_R + a_{SR}) + 1/(u_2(i) + a_{S2}(i))) \]

\[ posS = 2\Delta t/(1/(a_{SS} - u_S) + 1/(a_{S2}(i) - u_2(i))) \]

where \( u_Q = \text{value of } u \text{ at the base of the path line} \)

\( u_R = \text{value of } u \text{ at the base of the } C^+ \text{ characteristic} \)

\( u_S = \text{value of } u \text{ at the base of the } C^- \text{ characteristic} \)

\( a_{SR} = \text{value of } a_s \text{ at the base of the } C^+ \text{ characteristic} \)

\( a_{SS} = \text{value of } a_s \text{ at the base of the } C^- \text{ characteristic} \)

\( u_2(i) = \text{predicted value of } u \text{ obtained from the first-order approximation} \)

\( a_{S2}(i) = \text{predicted value of } a_s \text{ obtained from the first-order approximation} \)

Taylor's theorem was then used to obtain a formula for quadratic interpolation so that new values of \( P, T, u, z, (\partial z/\partial T)p, \rho, a_s, \Omega \) 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.

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

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

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

(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:

$$A(1) = \left[ \left[ \left( 1 + \frac{T}{\rho} \frac{\partial z}{\partial t} \right) \right]_Q + \left[ \left( 1 + \frac{T}{\rho} \frac{\partial z}{\partial t} \right) \right]_P \right] / 2C_p$$

$$A(2) = 1$$

$$A(3) = 0$$

$$A(4) = \left[ \left( \frac{1}{\rho a_s} \right)_R + \left( \frac{1}{\rho a_s} \right)_P \right] / 2$$

$$A(5) = 0$$

$$A(6) = 1$$

$$A(7) = \left[ \left( -\frac{1}{\rho a_s} \right)_S + \left( -\frac{1}{\rho a_s} \right)_P \right] / 2$$
A(8) = 0

A(9) = 1

\[ B(1) = A(1)P + TQ + \frac{\Delta t}{2C_{PA}} \left( \frac{\Omega + Wu}{\varrho} \right)_Q + \left( \frac{Q + Wu}{\varrho} \right)_{P'} \]

\[ B(2) = A(4)P_R + u_R + \frac{\Delta t}{2C_{PA}} \left[ \left( 1 + \frac{T}{2}\frac{\partial T}{\partial T} \right) \left( \frac{\Omega + Wu}{\varrho T} \right) s_R \right] 
+ \left[ \left( 1 + \frac{T}{2}\frac{\partial T}{\partial T} \right) \left( \frac{\Omega + Wu}{\varrho} \right) s_{P'} \right] - g \cdot \Delta t \sin \theta \]

\[ B(3) = A(7)P_S + u_S - \frac{\Delta t}{2C_{PA}} \left[ \left( 1 + \frac{T}{2}\frac{\partial T}{\partial T} \right) \left( \frac{\Omega + Wu}{\varrho T} \right) s_S \right] 
+ \left[ \left( 1 + \frac{T}{2}\frac{\partial T}{\partial T} \right) \left( \frac{\Omega + Wu}{\varrho} \right) s_{P'} \right] - g \cdot \Delta t \sin \theta \]

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 \( pQ \) 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:

\[ \text{pos}Q = \frac{2A_t}{1/u(i) + 1/u(i+1)} \]

assuming neither \( u(i) \) nor \( u(i+1) \) equal zero

\[ P_Q = \frac{\text{pos}Q}{Ax} \cdot P(i+1) + (1 - \frac{\text{pos}Q}{Ax}) \cdot P(i) \]

and for the second order approximation:

\[ P_Q = P(i) + \frac{\text{pos}Q}{2Ax} \left[ P(i+1) - P(i-1) \right] + \frac{\text{pos}Q^2}{2Ax^2} \left[ P(i+1) + P(i-1) - 2P(i) \right] \]
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:

\[ P_Q = \frac{\text{pos}Q}{2\Delta x} \cdot P(i-1) + \left(1 - \frac{\text{pos}Q}{2\Delta x}\right) \cdot P(i) \]

\[ P_R = \frac{\text{pos}R}{2\Delta x} \cdot P(i-1) + \left(1 - \frac{\text{pos}R}{2\Delta x}\right) \cdot P(i) \]

\[ P_S = \frac{\text{pos}S}{\Delta x} \cdot P(i+1) + \left(1 - \frac{\text{pos}S}{\Delta x}\right) \cdot P(i) \]

(Similarly for variables T, u, z, \((\partial z/\partial T)_{p}, \varphi, a_s, \Omega\) 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 $P$, $T$, $u$, $z$, $(\partial z/\partial T)_P$, $\varphi$, $a_s$, $\Theta$ 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:

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

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

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

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:
\[ P_Q = \frac{\text{pos} Q}{2\Delta x} \cdot P(i+1) + (1 - \frac{\text{pos} Q}{2\Delta x}) \cdot P(i) \text{ etc.} \]

in the first order approximation and equation (14) was used instead of equation (12) 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](image)

In the first order approximation the positions of Q, R and S were defined as:
posQ = \frac{4\Delta t}{1/(u(i-1) + 1/u_X) + 1/(u_X + a_X)} assuming neither u(i-1) nor u_X equal 0

posR = \frac{4\Delta t}{1/(u(i-1) + a_s(i-1)) + 1/(u_X + a_X)}

posS = \frac{2\Delta t}{1/(a_s(i+1) - u(i+1)) + 1/(a_Y - 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 $\Delta t$ 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:

\[ \text{posQ} = 4\Delta t/(1/u_Q + 1/u_2(i)) \] assuming neither \( u_Q \) nor \( u_2(i) \) equal 0

\[ \text{posR} = 4\Delta t/(1/(u_R + a_{SR}) + 1/(u_2(i) + a_{S2}(i))) \]

\[ \text{posS} = 2\Delta t/(1/(a_{SS} - u_S) + 1/(a_{S2}(i) - u_2(i))) \]

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

\[ P_Q = P(i-1) + \frac{(2\Delta x - \text{posQ})}{4\Delta x} [P_X - P(i-2)] + \frac{(2\Delta x - \text{posQ})^2}{8\Delta x^2} [P_X + P(i-2) - 2P(i-1)] \]

\[ P_R = P(i-1) + \frac{(2\Delta x - \text{posR})}{4\Delta x} [P_X - P(i-2)] + \frac{(2\Delta x - \text{posR})^2}{8\Delta x^2} [P_X + P(i-2) - 2P(i-1)] \]

\[ P_S = P(i+1) + \frac{(\Delta x - \text{posS})}{2\Delta x} [P(i+2) - P_Y] + \frac{(\Delta x - \text{posS})^2}{2\Delta x^2} [P(i+2) + P_Y - 2P(i+1)] \]

(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:

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

\[ \text{posR} = 2\Delta t/(1/(u(i-1) + a_s(i-1)) + 1/(u_Y + a_SY)) \]

\[ \text{posS} = 4\Delta t/(1/(a_S(i+1) - u(i+1)) + 1/(a_SX - u_X)) \]

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:

\[ P_Q = \text{posQ} \cdot P(i-1) + (1 - \text{posQ}) \cdot P_Y \]

\[ P_R = \text{posR} \cdot P(i-1) + (1 - \text{posR}) \cdot P_Y \]

\[ P_S = \text{posS} \cdot P(i+1) + (1 - \text{posS}) \cdot P_X \]
The modification to the general solution in this instance was to replace $\Delta t$ 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 \( P, u, T, z, (\partial z/\partial T)p, \varphi, a_s, \Omega \) and \( W \) at the point \( S \). For example:

\[
P_S = P((2) - (Ax - posS))
\]

\[
= P(2) - \frac{(Ax-posS)}{2Ax} [P(3) - P(1)] + \frac{(Ax-posS)^2}{2Ax^2} [P(3)+P(1)-2P(2)]
\]

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).

\[ \Delta t \]

\[ \Delta x \]

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:

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

where \( P_o \) = pressure in the high pressure zone prior to rupture
\( a_o \) = isentropic wavespeed prior to rupture \( = \sqrt{\frac{\gamma P_o}{\rho_o}} \)

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:

\[ u' = \frac{a_{AT} \left[ \left( \frac{P}{P_{AT}} \right)^{\gamma} - 1 \right]}{\sqrt{\frac{\gamma}{2} \left( \gamma + 1 \right) \left( \frac{P}{P_{AT}} \right)^{\gamma} + (\gamma - 1)}} \]  \hspace{1cm} (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 \( = \sqrt{\frac{\gamma P_{AT}}{\rho_{AT}}} \)
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:

\[ u = u' \]

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

(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:

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

(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:

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

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:

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

(4.6)

where $P_o$ is the pressure in the pipe at the break point prior to rupture and $\gamma$ 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 ($\gamma = 1.4$), the following expression is obtained:

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

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:

\[ P(j) = (P_0 - P_{eq})(1 - \frac{j}{6m})^n + P_{eq} \]

where \(P_0\) = 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).
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

<table>
<thead>
<tr>
<th>Property</th>
<th>Symbol</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specific heat</td>
<td>$C_p$</td>
<td>(kJ/kg K)</td>
</tr>
<tr>
<td>Gas Constant</td>
<td>$R$</td>
<td>(kJ/kg K)</td>
</tr>
<tr>
<td>Critical Temperature</td>
<td>$T_c$</td>
<td>(°C)</td>
</tr>
<tr>
<td>Critical Pressure</td>
<td>$P_c$</td>
<td>(kPa)</td>
</tr>
</tbody>
</table>

### Pipe Data

<table>
<thead>
<tr>
<th>Property</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Diameter of Pipe</td>
<td>(m)</td>
</tr>
<tr>
<td>Angle of Inclined Pipe</td>
<td>(degrees)</td>
</tr>
<tr>
<td>Length of Pipe upstream of break point</td>
<td>(m)</td>
</tr>
<tr>
<td>Length of Pipe downstream of break point</td>
<td>(m)</td>
</tr>
<tr>
<td>Required Grid Size near the break</td>
<td>(m)</td>
</tr>
<tr>
<td>Darcy friction factor</td>
<td></td>
</tr>
<tr>
<td>Stanton Number</td>
<td></td>
</tr>
<tr>
<td>Wall Temperature</td>
<td>(°C)</td>
</tr>
<tr>
<td>Atmospheric Temperature</td>
<td>(°C)</td>
</tr>
<tr>
<td>Atmospheric Pressure</td>
<td>(kPa)</td>
</tr>
<tr>
<td>Initial Temperature along Pipe</td>
<td>(°C)</td>
</tr>
<tr>
<td>Initial Pressure at upstream end of Pipe</td>
<td>(kPa)</td>
</tr>
<tr>
<td>Mass flow rate through Pipe</td>
<td>(kg/s)</td>
</tr>
</tbody>
</table>

### Run Data

<table>
<thead>
<tr>
<th>Property</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Length of time step required</td>
<td>(msec)</td>
</tr>
<tr>
<td>Total Run Time required</td>
<td>(secs)</td>
</tr>
</tbody>
</table>

(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](image-url)
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:-

\[ \sum F = \dot{m} (u_2 - u_1) \]
Applying this to the section of pipe shown in Figure 5.3:

\begin{figure}
\centering
\includegraphics[width=0.8\textwidth]{figure5.3.png}
\caption{Linear Momentum Equation for Steady One-Dimensional Pipe Flow}
\end{figure}

\[(P_1 - P_2)A - w\delta x - \rho A\delta x g \sin \theta = \dot{m}(u_2 - u_1)\]

Dividing through by \(A\) and re-arranging:

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

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

\[P_2 = \rho_2 RTZ\]

\[= \frac{\dot{m}}{u_2 A} RTZ \quad \text{where } \dot{m} = \text{mass flow rate}\]

Substituting this into equation (5.1):

\[\left\{P_1 - \frac{\dot{m}}{u_2} RTZ\right\} + \frac{\dot{m}}{A}(u_1 - u_2) - \frac{\rho f u^2 \delta x}{2d} \left[\frac{u_1 + u_2}{2}\right]
\]

\[\quad - \left[\frac{\rho_1 + \dot{m}/u_2}{2}\right] \delta x g \sin \theta = 0\]
This may be re-written producing a quadratic equation in \( u_2 \).

\[
\left[ \frac{f m^2 x}{4d} + \frac{\dot{m}}{A} \right] u_2^2 + \left[ \frac{f m^2 x}{4d} u_1 + \frac{\dot{P}}{2} \right] e x g \sin \theta = P_1 - \frac{\dot{m} u_1}{A} \]

\[
+ \left[ \frac{m^2 x}{2} g \sin \theta + \dot{m} R T Z \right] = 0
\]

The subroutines STEAD1 and STEAD2 solve this quadratic and can then deduce the density and pressure at the next grid point (i.e. \( \rho_2 \) 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).

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

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:-
Transfer variables at time t from main program

Define positions of points Q, R and S and calculate variables p, t, u, z, zt, ro, as, ht and w at Q, R and S

Simultaneously solve the \( C^+ \), \( C^- \) and path equations to obtain values for ppl(i), uul(i) and ttl(i)

Save these first-order values for ppl(i), uul(i) and ttl(i)

Calculate zzl(i), zzpl(i), zztl(i), rrol(i), assl(i), hhtl(i) and wvl(i)

Define new positions of points Q, R and S and calculate variables p, t, u, z, zt, ro, as, ht and w at Q, R and S

Simultaneously solve the \( C^+ \), \( C^- \) and path equations to obtain new values for ppl(i), uul(i) and ttl(i)

Compare the new values for ppl(i) and ttl(i) with previously obtained values

Is the iterative process converging?

Is the difference between the new and previously calculated values less than \( 1\% \)?

Print a warning message

Pass the new values for ppl(i), ttl(i) and uul(i) back to the main program

EXIT FROM SUBROUTINE; RETURN TO MAIN PROGRAM

Figure 5.5(a). Flow Diagram for SUB1, SUB2 and SUB5
Transfer variables at time t from main program

Define positions of points R and S and calculate variables \( p, t, u, z, zt, ro, as, ht \) and \( w \) at R and S

Check for flow reversal

Define position of point Q and calculate variables \( p, t, u, z, zt, ro, as, ht \) and \( w \) at Q

Simultaneously solve the \( C^+, C^- \) and path equations to obtain values for \( pp2(i), uu2(i) \) and \( tt2(i) \)

Save these first-order values for \( pp2(i), uu2(i) \) and \( tt2(i) \)

Calculate \( zz2Z(i), zsp2(i), zzt2(i), rro2(i), ass2(i), hht2(i) \) and \( w2(i) \)

Define new positions of points R and S and calculate variables \( p, t, u, z, zt, ro, as, ht \) and \( w \) at R and S

Check for flow reversal

Define new position for point Q and calculate variables \( p, t, u, z, zt, ro, as, ht \) and \( w \) at Q

Simultaneously solve the \( C^+, C^- \) and path equations to obtain new values for \( pp2(i), uu2(i) \) and \( tt2(i) \)

Compare the new values for \( pp2(i) \) and \( tt2(i) \) with previously obtained values

Is the iterative process converging?

Print a warning message

Pass the new values for \( pp2(i), uu2(i) \) and \( tt2(i) \) back to the main program

EXIT FROM SUBROUTINE; RETURN TO MAIN PROGRAM

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.](image)

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 $t + \Delta t$ 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.

![Diagram of pipe 1 and 2 with path and C characteristics](image)

**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 $t + \Delta t$. 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 $t + \Delta t$.

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 $t + \Delta t$. SUBUP models the upstream boundary in pipe 1 assuming a constant pressure and a constant mass flow
Transfer variables at time t from main program

Define positions of points Q, R and S (ensuring a continuous pipe) and calculate variables p, t, u, z, zt, ro, as, hs and w at Q, R and S

Simultaneously solve the path, C+ and C- equation using subroutine DMINV to obtain values for ppl(ml), ttl(ml) and uul(ml)

Save the first-order values for ppl(ml), ttl(ml) and uul(ml)

Calculate zzl(ml), zszl(ml), zztl(ml), ro1(ml), aa1(ml), hhl(ml) and wwl(ml)

Define new positions of points Q, R and S and calculate variables p, t, u, z, zt, ro, as, hs and w at Q, R and S

Using DMINV calculate new values for ppl(ml), ttl(ml) and wwl(ml) from the path, C+ and C- equations

Compare new values with previously calculated values for ppl(ml), ttl(ml) and wwl(ml)

Is the iterative process converging?

Is the difference between the new and previously calculated values less than 1%?

yes

no

Pass the first-order values for ppl(ml), ttl(ml) and wwl(ml) back to the main program with a warning message

EXIT FROM SUBROUTINE; RETURN TO MAIN PROGRAM

no

yes

Pass the new values for ppl(ml), ttl(ml) and wwl(ml) back to the main program

Figure 5.8(a) Flow Diagram for BREAK1
TRANSFER VARIABLES AT TIME T FROM MAIN PROGRAM


KNOWING PPL(ML) FROM THE MAIN PROGRAM: CALCULATE TTL(ML) AND UUL(ML) BY SIMULTANEously SOLVING THE C* AND PATH EQUATIONS

SAVE THE FIRST ORDER VALUES FOR TTL(ML) AND UUL(ML)

CALCULATE ZZL(ML), ZZPL(ML), ZZTL(ML), RROL(ML), AASL(ML), HHTL(ML) AND WML(ML)


CALCULATE NEW VALUES FOR TTL(ML) FROM THE PATH EQUATION AND UUL(ML) FROM THE C* EQUATION

COMPARE NEW VALUES FOR TTL(ML) AND UUL(ML) WITH PREVIOUSLY CALCULATED VALUES

IS THE ITERATIVE PROCESS CONVERGING?

YES

IS THE DIFFERENCE BETWEEN THE NEW AND PREVIOUSLY CALCULATED VALUES LESS THAN 1%?

YES

PASS THE FIRST-ORDER VALUES FOR TTL(ML) AND UUL(ML) BACK TO THE MAIN PROGRAM WITH A WARNING MESSAGE

EXIT FROM SUBROUTINE; RETURN TO MAIN PROGRAM

NO

PASS THE NEW VALUES FOR TTL(ML) AND UUL(ML) BACK TO THE MAIN PROGRAM

Figure 5.8(b) Flow Diagram for BREAK3
Transfer variables at time $t$ from main program

Define position of point $S$ and calculate variables $p$, $t$, $u$, $z$, $z_t$, $r_o$, $a_s$, $h_t$ and $w$ at $S$

**Is $u_2(1) > 0$?**
- **Yes**
  - Define position of point $Q$ and calculate variables $p$, $t$, $u$, $z$, $z_t$, $r_o$, $a_s$, $h_t$ and $w$ at $Q$
  - Calculate coefficients in the path equation
  - Calculate coefficients in the $C$-equation
  - **Is $u_2(1) > 0$?**
    - **Yes**
      - Simultaneously solve the path and $C$-equations to obtain values for $u_{12}(1)$ and $t_{12}(1)$
      - Using $t_{12}(1) = t_{11}(m_1)$, calculate $u_{12}(1)$ from the $C$-equation
      - Save the first-order values for $t_{12}(1)$ and $u_{12}(1)$
      - Calculate $z_{12}(1)$, $z_{22}(1)$, $z_{32}(1)$, $r_{02}(1)$, $a_{s2}(1)$, $h_{t2}(1)$, and $w_{22}(1)$
      - Define new position of point $S$ and calculate variables $p$, $t$, $u$, $z$, $z_t$, $r_o$, $a_s$, $h_t$ and $w$ at $S$
    - **No**
      - Define new position of point $Q$ and calculate variables $p$, $t$, $u$, $z$, $z_t$, $r_o$, $a_s$, $h_t$ and $w$ at $Q$
      - Calculate new value for $t_{12}(1)$ from the path equation
      - Calculate new value for $u_{12}(1)$ from the $C$-equation
      - Compare new values with previously calculated values for $u_{12}(1)$ and $t_{12}(1)$
      - **Is the iterative process converging?**
        - **Yes**
          - Pass the first-order values for $t_{12}(1)$ and $u_{12}(1)$ back to the main program with a warning message
          - **EXIT FROM SUBROUTINE; RETURN TO MAIN PROGRAM**
        - **No**
          - Is the difference between the new and previously calculated values less than 1%?
            - **Yes**
              - Pass the new values for $t_{12}(1)$ and $u_{12}(1)$ back to the main program
            - **No**
    - **No**
      - **EXIT FROM SUBROUTINE; RETURN TO MAIN PROGRAM**

**Figure 5.8(c) Flow Diagram for BREAK4**

- **Is $u_2(1) > 0$?**
  - **Yes**
    - Define $p_q = p_{21}(1)$, $t_q = t_{21}(1)$, $z_q = z_{21}(1)$, $z_{tq} = z_{t2}(1)$, $r_{0q} = r_{02}(1)$, $a_{sq} = a_{s2}(1)$, $u_q = 0$, $w_q = 0$ and $h_{tq} = 0$
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.
Enter subroutine

Transfer variables at time t from main program

Define position of point S and calculate variables p, t, u, z, zt, ro, as, ht and w at S

Calculate uu(1) from the C-characteristic assuming constant pressure

Save the first-order value for uu(1)

Calculate rrol(1) assuming constant mass flow rate and then tti(1) from the equation of state

Calculate zzl(1), zzpl(1), zzt(1), ass(1), htl(1) and wwl(1)

Define new position for point S and calculate variables p, t, u, z, zt, ro, as, ht and w at S

Calculate new value for uu(1) from the C-equation

Compare the new and previously calculated values for uu(1)

Is the iterative process converging?

Pass the first-order value for uu(1) back to the main program with a warning message

Calculate rrol(1) and tti(1)

ppl(1) = p(1)

Exit from subroutine; return to main program

Figure 5.9(a) Flow Diagram for SUBUP
IF NT
SUBROUTINE I

Transfer variables at time t from main program

Define position of point R and calculate variables p, t, u, z, st, ro, as, ht and w at R

Is u2(m2) = 0? yes

Is uu2(m2-1) < 0? yes

Define position of point Q and calculate variables p, t, u, z, st, ro, as, ht and w at Q

Calculate pp2(m2) from the C+ equation assuming uu2(m2) is equal to zero

Simultaneously solve the C+ and path equations to obtain values for uu2(m2) and pp2(m2)

Save the first-order values for pp2(m2) and uu2(m2)

Assuming tt2(m2) = t2(m2), calculate zz2(m2), zxp2(m2), zzt2(m2), rro2(m2), ass2(m2), hht2(m2) and w2(m2)

Define new position of point R and calculate variables p, t, u, z, st, ro, as, ht and w at R

Is uu2(m2) < 0? yes

Define new position of point Q and calculate variables p, t, u, z, st, ro, as, ht and w at Q

Re-define uu2(m2) = 0 and calculate new value for pp2(m2) from the C+ equation

Calculate new values for pp2(m2) and uu2(m2) from the C+ and path equations

Is the iterative process converging? yes

Is the difference between the new and previously calculated values for pp2(m2) less than 1%? no

Pass the first-order values for pp2(m2) and uu2(m2) back to the main program with a warning message

Pass the new values for pp2(m2) and uu2(m2) back to the main program

EXIT FROM SUBROUTINE; RETURN TO MAIN PROGRAM

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.
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 agreement 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.

![Diagram](image)

Δ - Pressure Transducer Positions

Diameter of Shock Tube = 0.0603 m

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 $\frac{1}{16}$" 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](image-url)
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.

![Diagram](image)

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 ($\gamma$):

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

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

$$\frac{1}{\sqrt{f}} = -0.6 \log \left( \left[ \frac{6.9}{Re} \right]^3 + \left[ \frac{k/d}{3.7} \right]^3 \cdot 3.3 \right)$$

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 $1.8 \times 10^{-2}$. The Stanton number initial values were taken to be $2.7 \times 10^{-3}$ for the methane containing systems and $2.9 \times 10^{-3}$ 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.

<table>
<thead>
<tr>
<th>Position</th>
<th>( i ) value</th>
</tr>
</thead>
<tbody>
<tr>
<td>break</td>
<td>171</td>
</tr>
<tr>
<td>transducer 0.305 m from the break</td>
<td>139</td>
</tr>
<tr>
<td>transducer 0.61 m from the break</td>
<td>107</td>
</tr>
<tr>
<td>transducer 0.914 m from the break</td>
<td>91</td>
</tr>
<tr>
<td>transducer 1.83 m from the break</td>
<td>59</td>
</tr>
<tr>
<td>transducer 2.13 m from the break</td>
<td>55</td>
</tr>
</tbody>
</table>

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.
Groves Shock Tube Test - Methane

\[ f = 0.018 \]
\[ St = 0.0027 \]

Figure 7.1. Graph for Groves' Shock Tube Test - Methane
Figure 7.2. \( P \times \omega \) Graph for Groves' Shock Tube Test - Argon

\[ f = 0.018 \]
\[ St = 0.0029 \]
Groves Shock Tube Test - Natural Gas

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

\( f = 0.018 \)
\( St = 0.0027 \)
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:
<table>
<thead>
<tr>
<th>Position</th>
<th>i value</th>
</tr>
</thead>
<tbody>
<tr>
<td>break</td>
<td>144</td>
</tr>
<tr>
<td>transducer 2 ft 4 in from the break</td>
<td>116</td>
</tr>
<tr>
<td>transducer 4 ft from the break</td>
<td>96</td>
</tr>
<tr>
<td>transducer 8 ft from the break</td>
<td>64</td>
</tr>
<tr>
<td>transducer 13 ft from the break</td>
<td>41</td>
</tr>
<tr>
<td>transducer 45 ft from the break</td>
<td>16</td>
</tr>
<tr>
<td>transducer 90 ft from the break</td>
<td>7</td>
</tr>
</tbody>
</table>

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

\[ f = 0.018 \]
\[ St = 0.0027 \]
BGC Shock Tube Test 2

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
BGC Shock Tube Test 5

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
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
Figure 7.11. P x t Graph for British Gas Shock Tube Test 2
BGC Shock Tube Test 4

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

\[ f = 0.03 \]
\[ St = 0.0027 \]
Figure 7.13. Pressure Graph for British Gas Shock Tube Test 5

BGC 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

BGC Shock Tube Test 8

![Graph showing pressure (KPa) over time (secs)]

\[ f = 0.03 \]
\[ St = 0.0027 \]
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, $\Delta t = 0.74$ s, for NABTF3, $\Delta t = 0.72$ s, for NABTF4, $\Delta t = 0.65$ s, and for NABTF5, $\Delta t = 0.75$ 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:

<table>
<thead>
<tr>
<th>Position</th>
<th>i value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.21 m west of break</td>
<td>1172</td>
</tr>
<tr>
<td>25.30 m west of break</td>
<td>1139</td>
</tr>
<tr>
<td>36.00 m west of break</td>
<td>1123</td>
</tr>
<tr>
<td>46.63 m west of break</td>
<td>1106</td>
</tr>
<tr>
<td>4.21 m east of break</td>
<td>2128</td>
</tr>
<tr>
<td>14.17 m east of break</td>
<td>2143</td>
</tr>
<tr>
<td>25.02 m east of break</td>
<td>2160</td>
</tr>
<tr>
<td>36.06 m east of break</td>
<td>2177</td>
</tr>
<tr>
<td>46.57 m east of break</td>
<td>2194</td>
</tr>
</tbody>
</table>
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 \omega Graph for Foothills Test Results NABTF1
Figure 7.19. P x t Graph for Foothills Test Results NABTF3 (West)
Foothills Test NABTF3

Figure 7.20. P x t Graph for Foothills Test Results NABTF3 (East)
Figure 7.21. P x w 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. $P \times \omega$ Graph for Foothills Test Results NABTF4
Figure 7.25. P x t Graph for Foothills Test Results NABTF5 (West)
Figure 7.26. P × t Graph for Foothills Test Results NABTF5 (East
Figure 7.27. $P \times \omega$ 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:

\[
a^2 = \left[\frac{dP}{dp}\right]_{\text{actual}}
\]

![Figure 8.1. Approximation of \( \frac{dp}{d\rho} \) on a finite grid.](image)
With reference to Figure 8.1, \( \frac{dP}{d\rho} \) 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 \( \frac{dP}{d\rho} \) 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:

<table>
<thead>
<tr>
<th></th>
<th>calculated wavespeed (m/s)</th>
<th>isentropic wavespeed (m/s)</th>
<th>isothermal wavespeed (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BGC 1</td>
<td>543</td>
<td>457</td>
<td>264</td>
</tr>
<tr>
<td>BGC 2</td>
<td>583</td>
<td>496</td>
<td>252</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Test</th>
<th>Calculated Theoretical Wavespeeds (m/s)</th>
<th>Calculated Experimental Wavespeeds (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>NABTF 1</td>
<td>472</td>
<td>471</td>
</tr>
<tr>
<td>NABTF 3</td>
<td>472</td>
<td>469</td>
</tr>
<tr>
<td>NABTF 4</td>
<td>491</td>
<td>487</td>
</tr>
<tr>
<td>NABTF 5</td>
<td>470</td>
<td>474</td>
</tr>
</tbody>
</table>

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:

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

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
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:

\[
\text{Modified} = \frac{\text{isothermal wavespeed} + \left(\frac{\text{theoretical wavespeed} - \text{isothermal wavespeed}}{2}\right)}{2}
\]

<table>
<thead>
<tr>
<th>TEST</th>
<th>Isothermal wavespeed</th>
<th>Isentropic wavespeed</th>
<th>Theoretical wavespeed</th>
<th>Modified wavespeed</th>
<th>Actual wavespeed</th>
</tr>
</thead>
<tbody>
<tr>
<td>NABTF 1</td>
<td>297</td>
<td>427</td>
<td>470</td>
<td>383.5</td>
<td>389</td>
</tr>
<tr>
<td>NABTF 3</td>
<td>266</td>
<td>441</td>
<td>468</td>
<td>367</td>
<td>354</td>
</tr>
<tr>
<td>NABTF 4</td>
<td>273</td>
<td>438</td>
<td>488</td>
<td>380.5</td>
<td>379</td>
</tr>
<tr>
<td>NABTF 5</td>
<td>286</td>
<td>405</td>
<td>469</td>
<td>377.5</td>
<td>375</td>
</tr>
</tbody>
</table>

(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:

\[
\frac{\partial p}{\partial t} + u \frac{\partial p}{\partial x} + p \frac{\partial u}{\partial x} = 0 \quad (1)
\]

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

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

To obtain \( \rho \) in terms of \( P, z \) and \( T \)

From the equation of state

\[
\rho = \frac{P}{RTz}
\]

Therefore

\[
\ln \rho = \ln P - \ln R - \ln T - \ln z
\]

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

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

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

\[
\therefore \quad \frac{1}{\rho} \frac{d\rho}{dt} = \left( \frac{1}{P} - \frac{1}{z} \left( \frac{\partial z}{\partial P} \right)_T \right) \frac{dP}{dt} - \left( \frac{1}{T} + \frac{1}{z} \left( \frac{\partial z}{\partial T} \right)_P \right) \frac{dT}{dt}
\]
Substituting this into equation (1):

\[
\left[ \frac{1}{P} - \frac{1}{z} \left( \frac{\partial^2 z}{\partial T \partial P} \right) \right] \frac{dP}{dt} - \left[ \frac{1}{T} + \frac{1}{z} \left( \frac{\partial^2 z}{\partial T \partial P} \right) \right] \frac{dT}{dt} + \frac{\partial u}{\partial x} = 0 \tag{4}
\]

To obtain \( h \) in terms of \( P, z \) and \( T \)

From Zemansky [1968]:

\[
\frac{dh}{dt} = C_P \frac{dT}{dt} + \left\{ \frac{T}{P} \left( \frac{\partial C_P}{\partial T} \right) + 1 \right\} \frac{1}{P} \frac{dP}{dt}
\]

Substituting this into equation (3):

\[
\rho C_P \frac{dT}{dt} + \left\{ \frac{T}{P} \left( \frac{\partial C_P}{\partial T} \right) + 1 \right\} \frac{dP}{dt} = \frac{\Omega + wu}{A} \tag{5}
\]

Solving equations (4) and (5) simultaneously

\[
\begin{bmatrix}
\left[ \frac{1}{P} - \frac{1}{z} \left( \frac{\partial^2 z}{\partial T \partial P} \right) \right] \frac{dP}{dt} - \left[ \frac{1}{T} + \frac{1}{z} \left( \frac{\partial^2 z}{\partial T \partial P} \right) \right] \frac{dT}{dt} = - \frac{\partial u}{\partial x} \\
\left[ \frac{T}{P} \left( \frac{\partial C_P}{\partial T} \right) \right] \frac{dP}{dt} + \left[ \rho C_P \right] \frac{dT}{dt} = \frac{\Omega + wu}{A}
\end{bmatrix}
\]

Solving for \( \frac{dP}{dt} \):

\[
\rho C_P \left[ \frac{1}{P} - \frac{1}{z} \left( \frac{\partial^2 z}{\partial T \partial P} \right) \right] \frac{dP}{dt} + \left[ \frac{1}{T} + \frac{1}{z} \left( \frac{\partial^2 z}{\partial T \partial P} \right) \right] \frac{T}{P} \left( \frac{\partial C_P}{\partial T} \right) \frac{dP}{dt}
\]

\[
= - \rho C_P \frac{\partial u}{\partial x} + \left[ \frac{1}{T} + \frac{1}{z} \left( \frac{\partial^2 z}{\partial T \partial P} \right) \right] \frac{\Omega + wu}{A} \tag{6}
\]

But from the equation of state:

\[
\ln P = \ln P - \ln R - \ln T - \ln z
\]
Differentiating this with respect to temperature $T$, keeping pressure $P$ constant:

$$\frac{1}{\rho} \left( \frac{\partial \rho}{\partial T} \right)_P = - \frac{1}{T} \left( \frac{\partial T}{\partial T} \right)_P - \frac{1}{z} \left( \frac{\partial z}{\partial T} \right)_P$$

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

Substituting this into equation (6):

$$\left\{ \rho C_p \left[ \frac{1}{P} - \frac{1}{z} \left( \frac{\partial z}{\partial T} \right)_T \right] \right\} \left[ \frac{1}{T} + \frac{1}{z} \left( \frac{\partial z}{\partial T} \right)_P \right]^2 \frac{dT}{dt} + \rho C_p \frac{\partial u}{\partial x} = \frac{1}{T} \left[ 1 + \frac{1}{z} \left( \frac{\partial z}{\partial T} \right)_P \right] \frac{\Omega + \omega u}{A}$$

Dividing through by $C_p$:

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

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

(7)

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

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

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

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

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

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

(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:

\[
0 = \left( \frac{\partial s}{\partial P} \right) \rho \cdot \rho + \left( \frac{\partial s}{\partial \rho} \right) P
\]

\[
\therefore \left( \frac{\partial P}{\partial \rho} \right)_s = -\left( \frac{\partial s}{\partial P} \right) P / \left( \frac{\partial s}{\partial \rho} \right)_P
\]

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

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

\[
dT = \left( \frac{\partial T}{\partial P} \right) \rho dP + \left( \frac{\partial T}{\partial \rho} \right) P d\rho
\]

\[
\Rightarrow \left( \frac{\partial T}{\partial P} \right) P / \left( \frac{\partial T}{\partial \rho} \right) \rho = -\frac{1}{\left( \frac{\partial P}{\partial T} \right)}
\]

Therefore:

\[
\left( \frac{\partial P}{\partial \rho} \right)_s = \left[ \left( \frac{\partial s}{\partial T} \right)_P / \left( \frac{\partial s}{\partial \rho} \right)_P \right] \cdot \frac{1}{\left( \frac{\partial P}{\partial T} \right)}
\]

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,

\[
\left( \frac{\partial P}{\partial T} \right)_V = -\left( \frac{\partial V}{\partial T} \right)_P / \left( \frac{\partial V}{\partial \rho} \right)_T
\]

Therefore:

\[
\left( \frac{\partial s}{\partial T} \right) \rho = \frac{C_P}{T} + \left( \frac{\partial V}{\partial T} \right)^2 / \left( \frac{\partial \rho}{\partial T} \right)_T
\]

\[
\quad = \frac{C_P}{T} + \frac{1}{\rho} \left[ \frac{\partial \rho}{\partial T} \right] P / \rho^2 \left[ \frac{\partial P}{\partial T} \right]_T
\]
\[
\left( \frac{\partial \rho}{\partial T} \right)_P = \frac{C_P}{T} \left( \frac{\partial \rho}{\partial P} \right)_T - \frac{1}{\rho^2} \left( \frac{\partial \rho}{\partial T} \right)_P^2
\]

But it has already been proved that:

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

And from the equation of state:

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

Substituting these identities into equation (9):

\[
\left( \frac{\partial P}{\partial \rho} \right)_s \frac{C_P}{T} \left[ \frac{\rho}{P} \left[ 1 - \frac{P}{z} \left( \frac{\partial \rho}{\partial P} \right)_T \right] - \frac{1}{C_P T} \left[ 1 + \frac{T}{z} \left( \frac{\partial \rho}{\partial P} \right)_P \right]^2 \right]^{-1} = \frac{\rho}{P} \left[ 1 + \frac{P}{z} \left( \frac{\partial \rho}{\partial P} \right)_T \left[ 1 + \frac{T}{z} \left( \frac{\partial \rho}{\partial P} \right)_P \right] \right]^{-1}
\]

and \( \left( \frac{\partial P}{\partial \rho} \right)_s \) can be defined as the ISENTROPIC WAVE SPEED \( \alpha_s \).

Substituting this into equations (7) and (8):

\[
\frac{1}{\alpha_s} \frac{dP}{dt} + \rho \frac{d\rho}{d\rho} = \frac{1}{C_P T} \left[ 1 + \frac{T}{z} \left( \frac{\partial \rho}{\partial P} \right)_T \right] \frac{\Omega + \omega u}{A}
\]

\[
\frac{1}{\alpha_s} \frac{dT}{dt} + \frac{1}{C_P} \left[ 1 + \frac{T_s (\partial \rho)}{z (\partial P)} \right] \frac{d\rho}{d\rho} = \frac{1}{C_P} \left[ 1 - \frac{P_s (\partial \rho)}{z (\partial P)} \right] \frac{\Omega + \omega u}{A}
\]

or alternatively, including equation (2):

\[
\frac{\partial P}{\partial t} + u \frac{\partial P}{\partial x} + \rho \frac{\partial \rho}{\partial x} = \frac{\omega u}{C_P T} \left[ 1 + \frac{T}{z} \left( \frac{\partial \rho}{\partial P} \right)_T \right] \frac{\Omega + \omega u}{A} \quad (11)
\]
\[
\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} + \frac{1}{\rho} \frac{\partial p}{\partial x} = -\frac{W}{\rho A} - g \sin \theta \quad (12)
\]

\[
\frac{\partial T}{\partial t} + u \frac{\partial T}{\partial x} + \frac{a_s^2}{C_p} \left[ 1 + \frac{T}{z} \left( \frac{\partial z}{\partial T} \right)_p \right] \frac{\partial u}{\partial x} = \frac{a_s^2}{C_p P} \left[ 1 - \frac{P}{z} \left( \frac{\partial z}{\partial P} \right)_T \right] \frac{\Omega + \omega u}{A} \quad (13)
\]
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 $\kappa$ and size $d$. The rough pipe law is of the form:

$$\frac{1}{\sqrt{f}} = A_1 \log \left( \frac{d}{\kappa} \right) + B_1$$

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:

$$\frac{1}{\sqrt{f}} = A_2 \log \left( \frac{\operatorname{Re}}{\sqrt{f}} \right) + B_2$$

and the Blasius form of the friction relationship is:

$$f = A_3 \operatorname{Re}^{B_3}$$

$A$ 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 ($W$):

$$W = \frac{A}{d} \varphi f \frac{u|u|}{2}$$

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 $10^5$: 

207
\[ f = 0.3160 \text{Re}^{-0.28} \]

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

\[ \text{Re} > 2 \times 10^3, \text{defined by:-} \]

\[ f = 0.046 \text{Re}^{-0.2} \]

Another equation that can be written in Blasius form is the Panhandle 'A' equation:

\[ \frac{1}{\sqrt{f}} = 3.39 \text{Re}^{0.073} \text{E} \]

\[ \Rightarrow f = 0.087 \text{Re}^{-0.146} \frac{1}{\text{E}^2} \]

where \( \text{E} \) 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:

\[ \frac{1}{\sqrt{f}} = 2 \log \left( \frac{\text{Re}}{\sqrt{f}} \right) - 0.3 \]
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:

$$\frac{1}{\sqrt{f}} = 2 \log \left( \frac{1}{k/3.7d + 2.5 \left(1/\sqrt{f}\right)/Re} \right)$$

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:

\[
\frac{1}{\sqrt{f}} = -2 \log \left( \frac{k}{3.7d} + \frac{5.74}{Re^{0.5}} \right)
\]

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

\[
\frac{1}{\sqrt{f}} = -2 \log \left( \frac{k}{3.7} + \frac{13}{Re} \right)
\]

Shacham

\[
\frac{1}{\sqrt{f}} = -2 \log \left( \frac{k}{3.7} + \frac{14.5}{Re} \right)
\]
Haaland [1983] re-examined the basis for Colebrook's equation and developed the explicit relation:

\[
\frac{1}{\sqrt{f}} = -1.8 \log \left\{ \frac{6.9}{\text{Re}} + \left( \frac{k/d}{3.7} \right)^{1.11} \right\}
\]

Haaland then generalised this equation to:

\[
\frac{1}{\sqrt{f}} = \frac{-1.8}{n} \log \left\{ (\frac{6.9}{\text{Re}})^{n} + \left( \frac{k/d}{3.7} \right)^{1.11n} \right\}
\]

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].
Taylor's expansion around a point \( x \) is given by:-

\[
\begin{align*}
\text{For a standard internal point with equidistant adjacent points, using the notation shown in Figure 4.2(i), these equations may be written:-}
\end{align*}
\]

\[
\begin{align*}
u(i + 1) &= u(i) + \Delta x u'(i) + \frac{(\Delta x)^2}{2} u''(i) + \frac{(\Delta x)^3}{6} u'''(i) + \ldots \ldots \ldots (1) \\
u(i - 1) &= u(i) - \Delta x u'(i) + \frac{(\Delta x)^2}{2} u''(i) - \frac{(\Delta x)^3}{6} u'''(i) + \ldots \ldots \ldots (2)
\end{align*}
\]

Adding equations (1) and (2) produces:

\[
\begin{align*}
u(i + 1) + u(i - 1) &= 2u(i) + 2 \frac{(\Delta x)^2}{2} u''(i) \text{ neglecting higher order terms} \\
\Rightarrow \quad u''(i) &= \frac{1}{\Delta x^2} \left\{ u(i + 1) + (u - 1) - 2u(i) \right\} \quad (3)
\end{align*}
\]

Subtracting equation (2) from equation (1) produces:

\[
\begin{align*}
u(i + 1) - u(i - 1) &= 2\Delta x u'(i) \text{ neglecting higher order terms} \\
\Rightarrow \quad u'(i) &= \frac{1}{2\Delta x} \left\{ u(i + 1) - u(i - 1) \right\} \quad (4)
\end{align*}
\]
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:

$$u_Q = u(i - \text{pos}Q)$$

$$= u(i) - \frac{\text{pos}Q}{2\Delta x} \{u(i+1) - u(i-1)\} + \frac{\text{pos}Q^2}{2\Delta x^2} \{u(i+1) + u(i-1) - 2u(i)\}$$

(5)

Similarly for point R:

$$u_R = u(i) - \frac{\text{pos}R}{2\Delta x} \{u(i+1) - u(i-1)\} + \frac{\text{pos}R^2}{2\Delta x^2} \{u(i+1) + u(i-1) - 2u(i)\}$$

(6)

And for point S:

$$u_S = u(i) + \frac{\text{pos}S}{2\Delta x} \{u(i+1) - u(i-1)\} + \frac{\text{pos}S^2}{2\Delta x^2} \{u(i+1) + u(i-1) - 2u(i)\}$$

(7)

Equations (5), (6) and (7) are used to define each of the variables $P, u, T, z, \partial z/\partial T, \phi, a_s, \Omega$, 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:

$$u''(i + 1) = \frac{1}{(\Delta x)^2} \{u(i + 2) + u(i) - 2u(i + 1)\}$$

$$u'(i + 1) = \frac{1}{2\Delta x} \{u(i + 2) - u(i)\}$$
\[ u''(i - 1) = \frac{1}{4\Delta x^2} \left[u(i) + u(i - 2) - 2u(i - 1)\right] \]

\[ u'(i - 1) = \frac{1}{4\Delta x} \left[u(i) - u(i - 2)\right] \]

Therefore,

\[ u_Q = u((i - 1) + (2\Delta x - \text{pos}_Q)) = u(i - 1) + \frac{(2\Delta x - \text{pos}_Q)}{4\Delta x} \left[u(i) - u(i - 2)\right] \]

\[ + \frac{(2\Delta x - \text{pos}_Q)^2}{8\Delta x^2} \left[u(i) + u(i - 2) - 2u(i - 1)\right] \] (8)

For point R:

\[ u_R = u((i - 1) + (2\Delta x - \text{pos}_R)) = u(i - 1) + \frac{(2\Delta x - \text{pos}_R)}{4\Delta x} \left[u(i) - u(i - 2)\right] \]

\[ + \frac{(2\Delta x - \text{pos}_R)^2}{8\Delta x^2} \left[u(i) + u(i - 2) - 2u(i - 1)\right] \] (9)

and for point S:

\[ u_S = u((i + 1) - (\Delta x - \text{pos}_S)) = u(i + 1) - \frac{(\Delta x - \text{pos}_S)}{2\Delta x} \left[u(i + 2) - u(i)\right] \]

\[ + \frac{(\Delta x - \text{pos}_S)^2}{2\Delta x^2} \left[u(i + 2) + u(i) - 2u(i + 1)\right] \] (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:


\[ u_Q = u((i - 1) + (\Delta x - \text{pos}_Q)) \]

\[ = u(i - 1) + \frac{(\Delta x - \text{pos}_Q)}{2\Delta x} \{u(i) - u(i - 2)\} \]

\[ + \frac{(\Delta x - \text{pos}_Q)^2}{2\Delta x^2} \{u(i) + u(i - 2) - 2u(i - 1)\} \quad (11) \]

\[ u_R = u((i - 1) + (\Delta x - \text{pos}_R)) \]

\[ = u(i - 1) + \frac{(\Delta x - \text{pos}_R)}{2\Delta x} \{u(i) - u(i - 2)\} \]

\[ + \frac{(\Delta x - \text{pos}_R)^2}{2\Delta x^2} \{u(i) + u(i - 2) - 2u(i - 1)\} \quad (12) \]

\[ u_S = u((i + 1) - (2\Delta x - \text{pos}_S)) \]

\[ = u(i + 1) - \frac{(2\Delta x - \text{pos}_S)}{4\Delta x} \{u(i + 2) - u(i)\} \]

\[ + \frac{(2\Delta x - \text{pos}_S)^2}{8\Delta x^2} \{u(i) + u(i + 2) - 2u(i + 1)\} \quad (13) \]

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:

\[ u_Q = u((i + 1) - (2\Delta x - \text{pos}_Q)) \]

\[ = u(i + 1) - \frac{(2\Delta x - \text{pos}_Q)}{4\Delta x} \{u(i + 2) - u(i)\} \]

\[ + \frac{(2\Delta x - \text{pos}_Q)^2}{8\Delta x^2} \{u(i + 2) + u(i) - 2u(i + 1)\} \quad (14) \]

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:

\[ u_Q = u(i - 1) + \frac{(2\Delta x - \text{posQ})}{4\Delta x} \left[ u_X - u(i - 2) \right] \]

\[ + \frac{(2\Delta x - \text{posQ})^2}{8\Delta x^2} \left[ u_X + u(i - 2) - 2u(i - 1) \right] \]  \hspace{1cm} (15)

\[ u_R = u(i - 1) + \frac{(2\Delta x - \text{posR})}{4\Delta x} \left[ u_X - u(i - 2) \right] \]

\[ + \frac{(2\Delta x - \text{posR})^2}{8\Delta x^2} \left[ u_X + u(i - 2) - 2u(i - 1) \right] \]  \hspace{1cm} (16)

\[ u_S = u(i + 1) - \frac{(\Delta x - \text{posS})}{2\Delta x} \left[ u(i + 2) - u_Y \right] \]

\[ + \frac{(\Delta x - \text{posS})^2}{2\Delta x^2} \left[ u(i + 2) + u_Y - 2u(i + 1) \right] \]  \hspace{1cm} (17)

Similarly, for this situation occurring downstream of the break, Figure 4.2(v):

\[ u'_Q = u(i - 1) + \frac{(\Delta x - \text{posQ})}{2\Delta x} \left[ u_X - u(i - 2) \right] \]

\[ + \frac{(\Delta x - \text{posQ})^2}{2\Delta x^2} \left[ u_X + u(i - 2) - 2u(i - 1) \right] \]  \hspace{1cm} (18)

\[ u'_R = u(i - 1) + \frac{(\Delta x - \text{posR})}{2\Delta x} \left[ u_X - u(i - 2) \right] \]

\[ + \frac{(\Delta x - \text{posR})^2}{2\Delta x^2} \left[ u_X + u(i - 2) - 2u(i - 1) \right] \]  \hspace{1cm} (19)
\[ u_s = u(i + 1) - \frac{(2\Delta x - \text{pos})}{4\Delta x} \left( u(i + 2) - u_Y \right) \]
\[ + \frac{(2\Delta x - \text{pos})^2}{8\Delta x^2} \left\{ u(i + 2) + u_Y - 2u(i + 1) \right\} \quad (20) \]

and if flow reversal occurs, equation (18) is replaced by:

\[ u_Q = u(i + 1) - \frac{(2\Delta x - \text{pos})}{4\Delta x} \left( u(i + 2) - u_Y \right) \]
\[ + \frac{(2\Delta x - \text{pos})^2}{8\Delta x^2} \left\{ u(i + 2) + u_Y - 2u(i + 1) \right\} \quad (21) \]
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:

\[ \frac{dy}{dt} = F \left( \frac{dy}{dx} \right) \]  \hspace{1cm} (1)

where \( F \) is a function of a form to be determined.

Differentiating equation (1) with respect to time produces:

\[ \frac{d^2y}{dt^2} = \left( F \left( \frac{dy}{dx} \right) \right)^2 \frac{d^2y}{dx^2} \]  \hspace{1cm} (2)

From the conservation of mass, using the notation shown in Figure A.1.

\[ \rho_0 \, dx = \rho \, dy \]

\[ \Rightarrow \rho = \rho_0 \left( \frac{dy}{dx} \right)^{-1} \]  \hspace{1cm} (3)
Similarly, from the conservation of linear momentum:

\[ P\text{A} - \left[ P + \frac{\partial P}{\partial x} \text{d}x \right] \text{A} = \varphi_o \text{A} \text{d}x \frac{d^2 y}{dt^2} \]

\[ \Rightarrow - \frac{\partial P}{\partial x} = \varphi_o \frac{d^2 y}{dt^2} \]  

(4)

If the flow in the tube can be assumed to be isentropic, then:

\[ p = \frac{P_o}{\varphi_o^\gamma} \cdot \varphi^\gamma \]

\[ \therefore \frac{\partial P}{\partial x} = \frac{P_o}{\varphi_o^\gamma} \cdot \gamma \varphi_o^{\gamma-1} \cdot \frac{\partial \varphi}{\partial x} \]  

(5)

But by differentiating equation (3) with respect to \( x \), the following expression for \( \frac{\partial \varphi}{\partial x} \) is obtained:

\[ \frac{\partial \varphi}{\partial x} = -\varphi_o \left[ \frac{dy}{dx} \right]^{-2} \frac{d^2 y}{dx^2} \]  

(6)

Substituting equations (6) and (3) into equation (5) produces:

\[ \frac{\partial P}{\partial x} = \frac{P_o}{\varphi_o^\gamma} \cdot \gamma \varphi_o^{\gamma-1} \left[ \frac{dy}{dx} \right]^{-\gamma+1} \cdot \left( -\varphi_o \right) \cdot \left[ \frac{dy}{dx} \right]^{-2} \cdot \frac{d^2 y}{dx^2} \]

\[ = -P_o \gamma \left[ \frac{dy}{dx} \right]^{-\gamma-1} \cdot \frac{d^2 y}{dx^2} \]  

(7)

Equation (4) can then be re-written using equation (7):

\[ \frac{P_o}{\varphi_o} \cdot \gamma \left[ \frac{dy}{dx} \right]^{-\gamma-1} \cdot \frac{d^2 y}{dx^2} = \frac{d^2 y}{dt^2} \]  

(8)
Comparing equations (2) and (8):

\[
\left[ F' \left( \frac{dy}{dx} \right) \right]^2 = \frac{\gamma P_0}{\rho_0} \left/ \left[ \frac{dy}{dx} \right] \right]^{\gamma+1}
\]

\[
\therefore \quad F' \left( \frac{dy}{dx} \right) = \pm \sqrt{\frac{P_0}{\rho_0}} \left/ \left[ \frac{dy}{dx} \right] \right]^{\frac{\gamma+1}{2}}
\]

Integrating equation (9):

\[
F \left( \frac{dy}{dx} \right) = \pm \sqrt{\frac{\gamma P_0}{\rho_0}} \cdot \frac{2}{\gamma+1} \left[ \frac{dy}{dx} \right]^{\frac{1-\gamma}{2}} + C
\]

where C is a constant of integration.

Substituting for \( F \left( \frac{dy}{dx} \right) \) and \( \frac{dy}{dx} \) using equations (1) and (3) respectively:

\[
\frac{dv}{dt} = C \pm \sqrt{\frac{\gamma P_0}{\rho_0}} \cdot \frac{2}{1-\gamma} \left[ \frac{\rho_0}{\rho} \right]^{\frac{1-\gamma}{2}}
\]

\[
= C \pm \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_0}{\rho_0}} \left( \frac{\rho_0}{\rho} \right)^{\frac{1-\gamma}{2}}
\]

(10)

Taking boundary conditions as:

\[
u = \frac{dv}{dt} = 0 \text{ when } \rho = \rho_0
\]

then from equation (10):

\[
0 = C \pm \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_0}{\rho_0}}
\]

\[
\therefore C = \pm \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_0}{\rho_0}}
\]

Substituting this back into equation (10):

\[
u = \pm \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_0}{\rho_0}} \pm \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_0}{\rho_0}} \left( \frac{\rho_0}{\rho} \right)^{\frac{1-\gamma}{2}}
\]

\[
= \pm \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_0}{\rho_0}} \left( \frac{\rho_0}{\rho} \right)^{\frac{1-\gamma}{2}}
\]

\[
= \pm \frac{2}{\gamma-1} \sqrt{\frac{\gamma P_0}{\rho_0}} \left[ \left( \frac{\rho_0}{\rho} \right)^{\frac{1-\gamma}{2}} - 1 \right]
\]
but since isentropic flow has been assumed:
\[
\frac{\rho}{\rho_0} = \left(\frac{P}{P_0}\right)^\gamma
\]

and
\[
a_o = \sqrt{\frac{\gamma P_0}{\rho_0}}
\]

Therefore:
\[
u = -\frac{2}{\gamma - 1} a_o \left\{ \left[\frac{P}{P_0}\right]^{\frac{\gamma-1}{2\gamma}} - 1 \right\}
\]

\[
= \frac{2}{\gamma - 1} a_o \left\{ 1 - \left[\frac{P}{P_0}\right]^{\frac{\gamma-1}{2\gamma}} \right\}
\]

(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.
<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>SHOCK TUBE</th>
<th>FULL SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GROVES</td>
<td>BGC (S/T)</td>
</tr>
<tr>
<td></td>
<td>Test 7</td>
<td>Test 8</td>
</tr>
<tr>
<td>OXYGEN O₂</td>
<td>0.0008</td>
<td>0.022</td>
</tr>
<tr>
<td>NITROGEN N₂</td>
<td>1.498</td>
<td>0.91</td>
</tr>
<tr>
<td>CARBON DIOXIDE CO₂</td>
<td>1.073</td>
<td>0.62</td>
</tr>
<tr>
<td>METHANE CH₄</td>
<td>83.266</td>
<td>82.4</td>
</tr>
<tr>
<td>ETHANE C₂H₆</td>
<td>9.608</td>
<td>7.89</td>
</tr>
<tr>
<td>PROPANE C₃H₈</td>
<td>3.597</td>
<td>5.2</td>
</tr>
<tr>
<td>ISO-BUTANE iC₄H₁₀</td>
<td>0.3414</td>
<td>0.45</td>
</tr>
<tr>
<td>N-BUTANE nC₄H₁₀</td>
<td>0.4581</td>
<td>1.21</td>
</tr>
<tr>
<td>ISO-PENTANE iC₅H₁₂</td>
<td>0.0403</td>
<td>0.25</td>
</tr>
<tr>
<td>N-PENTANE nC₅H₁₂</td>
<td>0.0342</td>
<td>0.35</td>
</tr>
<tr>
<td>N-Hexane nC₆H₁₄</td>
<td>0.0046</td>
<td>0.64</td>
</tr>
<tr>
<td>N-HEPTANE nC₇H₁₆</td>
<td>0.0003</td>
<td>0.003</td>
</tr>
<tr>
<td>N-OCTANE nC₈H₁₈</td>
<td>0.0001</td>
<td>-</td>
</tr>
<tr>
<td>N-NONANE nC₉H₂₀</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

TABLE A1. NATURAL GAS MOLAR COMPOSITIONS (%)
<table>
<thead>
<tr>
<th>COMPONENT</th>
<th>SHOCK TUBE</th>
<th>FULL SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GROVES</td>
<td>BGC (S/T)</td>
</tr>
<tr>
<td></td>
<td>Test 7</td>
<td>Test 8</td>
</tr>
<tr>
<td>OXYGEN O₂</td>
<td>-</td>
<td>0.03</td>
</tr>
<tr>
<td>NITROGEN N₂</td>
<td>2.18</td>
<td>1.25</td>
</tr>
<tr>
<td>CARBON DIOXIDE CO₂</td>
<td>2.45</td>
<td>1.34</td>
</tr>
<tr>
<td>METHANE CH₄</td>
<td>69.40</td>
<td>64.89</td>
</tr>
<tr>
<td>ETHANE C₂H₆</td>
<td>15.01</td>
<td>11.65</td>
</tr>
<tr>
<td>PROPANE C₃H₈</td>
<td>8.24</td>
<td>11.26</td>
</tr>
<tr>
<td>ISO-BUTANE iC₄H₁₀</td>
<td>1.03</td>
<td>1.28</td>
</tr>
<tr>
<td>N-BUTANE nC₄H₁₀</td>
<td>1.38</td>
<td>3.45</td>
</tr>
<tr>
<td>ISO-PENTANE iC₅H₁₂</td>
<td>0.15</td>
<td>0.89</td>
</tr>
<tr>
<td>N-PENTANE nC₅H₁₂</td>
<td>0.13</td>
<td>1.24</td>
</tr>
<tr>
<td>N-HExANE nC₆H₁₄</td>
<td>0.02</td>
<td>2.71</td>
</tr>
<tr>
<td>N-HEPTANE nC₇H₁₆</td>
<td></td>
<td>0.01</td>
</tr>
<tr>
<td>N-OCTANE nC₈H₁₈</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-NONANE nC₉H₂₀</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE A2: NATURAL GAS MASS COMPOSITIONS (%)**
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ARGON Ar</td>
<td>39.948</td>
<td>87.3</td>
<td>150.8</td>
<td>4870</td>
<td>74.9</td>
<td>0.291</td>
<td>208.13</td>
<td>520</td>
<td>1.668</td>
</tr>
<tr>
<td>NITROGEN N2</td>
<td>28.013</td>
<td>77.4</td>
<td>126.2</td>
<td>3390</td>
<td>89.8</td>
<td>0.290</td>
<td>296.84</td>
<td>1037</td>
<td>1.401</td>
</tr>
<tr>
<td>CARBON DIOXIDE CO₂</td>
<td>44.010</td>
<td>316.5</td>
<td>304.1</td>
<td>3730</td>
<td>93.9</td>
<td>0.274</td>
<td>188.92</td>
<td>819</td>
<td>1.300</td>
</tr>
<tr>
<td>METHANE CH₄</td>
<td>16.043</td>
<td>111.6</td>
<td>190.4</td>
<td>4600</td>
<td>99.2</td>
<td>0.288</td>
<td>518.36</td>
<td>2174</td>
<td>1.313</td>
</tr>
<tr>
<td>ETHANE C₂H₆</td>
<td>30.070</td>
<td>184.6</td>
<td>305.4</td>
<td>4880</td>
<td>148.3</td>
<td>0.285</td>
<td>276.51</td>
<td>1533</td>
<td>1.220</td>
</tr>
<tr>
<td>PROPANE C₃H₉</td>
<td>44.094</td>
<td>231.1</td>
<td>369.8</td>
<td>4250</td>
<td>203.0</td>
<td>0.281</td>
<td>188.54</td>
<td>1639</td>
<td>1.130</td>
</tr>
<tr>
<td>ISO-BUTANE iC₄H₁₀</td>
<td>58.124</td>
<td>261.4</td>
<td>408.2</td>
<td>3650</td>
<td>263</td>
<td>0.283</td>
<td>143.03</td>
<td>1633</td>
<td>1.096</td>
</tr>
<tr>
<td>N-BUTANE nC₄H₁₀</td>
<td>58.124</td>
<td>272.7</td>
<td>425.2</td>
<td>3800</td>
<td>255</td>
<td>0.274</td>
<td>143.03</td>
<td>1633</td>
<td>1.096</td>
</tr>
<tr>
<td>ISO-PENTANE iC₅H₁₂</td>
<td>72.151</td>
<td>282.6</td>
<td>433.8</td>
<td>3200</td>
<td>303</td>
<td>0.269</td>
<td>115.24</td>
<td>1455</td>
<td>1.086</td>
</tr>
<tr>
<td>N-PENTANE nC₅H₁₂</td>
<td>72.151</td>
<td>309.2</td>
<td>469.7</td>
<td>3370</td>
<td>304</td>
<td>0.263</td>
<td>115.24</td>
<td>1455</td>
<td>1.086</td>
</tr>
<tr>
<td>N-HEXANE nC₆H₁₄</td>
<td>86.178</td>
<td>341.9</td>
<td>507.5</td>
<td>3010</td>
<td>370</td>
<td>0.264</td>
<td>96.48</td>
<td>1302</td>
<td>1.080</td>
</tr>
</tbody>
</table>

**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

notation:-

aat isentropic wavespeed for atmosphere
ao initial isentropic wavespeed at the break point
ar cross-sectional area of the pipeline
as isentropic wavespeed
cp specific heat at constant pressure
d diameter of pipeline
dt variable time step used in subroutines
dt1 specified time step
dx variable section length used in subroutines
dx1 specified elemental pipe section length
dd grid length near the break (= dx/64)
e flow rate / area
f friction factor
g acceleration due to gravity
ga ratio of specific heats
ht heat transfer
l1 length of pipe before break
l2 length of pipe after break
m1 number of i values before break
m2 number of i values after break
n1 number of elemental sections of length dx before break
n2 number of elemental sections of length dx after break
p pressure at a point
pat atmospheric pressure
pc critical pressure of the gas
epc critical equalisation pressure
pec critical equalisation pressure
pi pi (=3.14159)
r specific gas constant
ro density at a point
st stanton number
t temperature at a point
tat atmospheric temperature
tc critical temperature of the gas
th angle of inclination of the pipe
ti time after break
tm total run time
tw pipe wall temperature
u flow velocity at a point
w frictional force
z compressibility factor
zp (dz/dp) at constant temperature
zt (dz/dt) at constant pressure

character*1 quest
implicit double precision (a-h, o-z)
double precision 11, 12

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), rx(6), asx(6), htx(6),

226
& p(6), u(6), tz(6), z(6), w(6), tz(6), roi(6), asi(6), htz(6),
& wave1(300), wave2(300), at1(300), at2(300)
pi=3.141592654

g=9.81

100 format(v)
ti=0.0

Read in initial gas data

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

Read in initial pipeline data

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, 11, 12, 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*, 11
print*, 'length of pipe downstream of the break point (m) ?'
read*, 12
print*, 'required grid size near the break (m) ?'
if(12. eq. 0. 0) goto 111
dx2=min0(11, 12)/576.0
goto 112

111 dx2=11/576.0
112 write (6, B) 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

227
Calculate initial isothermal flow conditions along the pipe

4 dx1=dd*64
Ga=cp/((cp-r)
Aat=dsqrt(1.4*287*tat)
N1=ifix(11/dx1+0.5)
N2=ifix(12/dx1+0.5)
M1=M1+121
M2=M2+121
Ar=0.25*pi*dx1
call stead1 (pl, tl, u1, z1, ro1, r, e, d, pc, tc, th, f, g, pi, dx1, m1, n1)
ao=dsqrt((Ga*pl(m1)/rol(m1))
if(12. eq. 0.0) goto 113
p2(1)=pl(m1)
t2(1)=tl(m1)
u2(1)=u1(m1)
z2(1)=z1(m1)
ro2(1)=rol(m1)
call stead2 (p2, t2, u2, z2, ro2, r, e, d, pc, tc, th, f, g, dx1, m2, n2)

Calculate wavespeed at time to at i=1 in pipe 1

113 zp1(1)=(z1(1)-1)/pl(1)
ztl(1)=(81*tc**3/(64*t1(1)**4)-9*tc/(128*t1(1)**2))*pl(1)/pc
Asl(1)=((1+ztl(1)*t1(1)/z1(1))**2)*pl(1)/(rol(1)*t1(1)*cp)
Asl(1)=1-zp1(1)*pl(1)/(z1(1)-as1(1))*rol(1)/pl(1)
Asl(1)=1/dsrt(dabs(asl(1))}

Read in the run data

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

print*, '**********************************************************************'
print*, ' Transient Analysis Results'
print*, '**********************************************************************'

Set number of transducer points for printing out results at

nint=12
write(42,99)nint
99 format(14)

Transient calculations for pipe 1 (upstream of the break)
53 DO 70 J=1,64
         TIT=TI+DT1/64
         DO 5 I=1,M1
             Z1(I)=7*TC/(128*T1(I))-27*TC**3/(64*T1(I)**3)
             Z2(I)=Z1(I)*P1(I)/PC+1
             ZP1(I)=Z1(I)-1/P1(I)
             ZTI(I)=81*TC**3/(64*T1(I)**4)-9*TC/(128*T1(I)*T1(I))
             ZTL(I)=ZTI(I)*P1(I)/PC
             R01(I)=P1(I)/((R*T1(I)*Z1(I))
             W1(I)=DABS(ABS*RO1(I)*F*U1(I)*U1(I))/(2*D)
             AS1(I)=(1+Z1(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)=AS1(I)*R01(I)/P1(I)
             AS1(I)=AS1(I)/DSQRT(DABS(AS1(I)))
             HT1(I)=PI*CP*ST*D*RO1(I)*U1(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))
         CONTINUE
      DO 10 I=1,M2
         IF(12.EQ.0.0) GO TO 10
         Z2(I)=7*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
         R02(I)=P2(I)/((R*T2(I)*Z2(I))
         W2(I)=DABS(ABS*RO2(I)*F*U2(I)*U2(I))/(2*D)
         AS2(I)=(1+Z2(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)=AS2(I)/DSQRT(DABS(AS2(I)))
         HT2(I)=PI*CP*ST*D*RO2(I)*U2(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))
      CONTINUE
      IF(JK.EQ.0) GO TO 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.297) GO TO 881
         IF(I.EQ.172) GO TO 881
         IF(I.EQ.157) GO TO 881
         IF(I.EQ.140) GO TO 881
         IF(I.EQ.123) GO TO 881
         IF(I.EQ.106) GO TO 881
      GO TO 52
881 I=I+1000
      WRITE(42,55)I,PPS1(I),U1(I),TTS1(I),WAVE1(I),AT1(I)
      CONTINUE
      IF(12.EQ.0.0) GO TO 116
      DO 46 I=1,M2
52 CONTINUE
tts2(i) = t2(i) - 273.16
wave2(i) = a2(i) - dabs(u2(i))
if(i.eq.1) goto 882
if(i.eq.129) 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, pp2(i), u2(i), tts2(i), wave2(i), at2(i)
continue
116
jk = 0
dx = dx1/64
dt = dt1/64
Do 37 i = 1, ml
pp1(i) = p1(i)
tt1(i) = t1(i)
uu1(i) = u1(i)
37
continue
Do 40 i = 1, m2
if(i2.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(i1.1t.(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, rol, w1, as1, hti, th)
goto 12
13 if(ti.gt.(81*dt1)) goto 58
if(ti.gt.dt1)) goto 92
if(i2.eq.0.0) goto 120
call break1(p1, t1, u1, p2, t2, u2, pp1, tt1, uu1, z1, zt1, rol, as1, hti, & w1, dx, tc, pc, r, ar, f, d, cp, pi, st, tw, g, dt, i, th, z2, zt2, rol, as2, & hti, tw)
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 = (1-(pec/pp1(m1))**((ga-1)/(2*ga))
y2 = dsqrt(14*(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
pe = pe
y1 = (2/(ga-1))*as1(m1)*(1-(pec/pp1(m1))**((ga-1)/(2*ga))
goto 71
72 pe = pe
ueq = y1
teq = tt1(m1)*((pe/pp1(m1))**((ga-1)/(2*ga))
goto 12
55 format(i4, 1x, 5f14.4)
pp1(ml)=p(leave-peq)*((1.0-kk/5120.0)**2)+peq
58 call break3(pl, ti, ui, pp1, ttl, uu1, zi, zt1, rol, as1, htl, 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, ttl, uu1, tc, pc, r, ar, f, d, cp, pi, st, & tw, g, dt, i, zi, zp1, zt1, rol, w1, as1, htl, th)
py(1)=pp1(m1-64)
ty(1)=tt1(m1-64)
uy(1)=u1(m1-64)
goto 11
14 j1=j1/2
dx=dx*2
dt=dt*2
dol=
ii=(m1-64)
i=ii
if (i.ge.(m1-65)) goto 15
if (i.gt.(m1-65)) goto 16
call sub1(p1, t1, ui, dx, pp1, ttl, uu1, tc, pc, r, ar, f, d, cp, pi, st, & tw, g, dt, i, zi, zp1, zt1, rol, w1, as1, htl, th)
goto 15
16 if (i.gt.(m1-65)) goto 81
pl(i+1)=px(k)
u1(i+1)=ux(k)
t1(i+1)=tx(k)
z1(i+1)=zx(k)
w1(i+1)=wx(k)
z1(i+1)=zt1(x(k))
rol(i+1)=roxy(k)
asi(i+1)=asx(k)
h1(i+1)=htx(k)
call sub1(p1, t1, ui, dx, pp1, ttl, uu1, tc, pc, r, ar, f, d, cp, pi, st, & tw, g, dt, i, zi, zp1, zt1, rol, w1, as1, htl, th)
goto 15
81 dt=dt/2
dx=dx/2
zy(k)=(9*tc/(128*tx(k))-27*tc**3/(64*tx(k)**3))*px(k)/pc+1
zpy(k)=(izy(k)-1)/py(k)
zty(k)=(81*tc**3/(64*ty(k)**4)-9*tc/(128*ty(k)**3))*py(k)/pc
roz(k)=px(k)/(r*tx(k)*zx(k))
wx(k)=dabs(ar*roxy(k)*f*ux(k)*ux(k))/2*2
asx(k)=((1+tx(k)*ntx(k)/tx(k))*py(k)/(rov(k)*tx(k)*pc)
asx(k)=asx(k)+zpy(k)*py(k)/zx(k)-1
asx(k)=1/dsqrt(dabs(asx(k)*roxy(k)/px(k)))
hxy(k)=pi*cp*st*d*roxy(k)*uzx(k)*(tw-tx(k))
if (i.gt.(m1-65)) goto 81
pl(i+1)=px(k)
u1(i+1)=ux(k)
t1(i+1)=tx(k)
z1(i+1)=zx(k)
w1(i+1)=wx(k)
z1(i+1)=zt1(x(k))
rol(i+1)=roxy(k)
asi(i+1)=asx(k)
h1(i+1)=htx(k)
call sub5(p1, t1, ui, dx, pp1, ttl, uu1, tc, pc, r, ar, f, d, cp, pi, st, & tw, g, dt, i, zi, zp1, zt1, rol, w1, as1, htl, th, px, ux, tx, wx, xz, & ztx, rox, asx, htx, py, uy, ty, wy, zty, roy, asy, hty, k)
\[ dx = dx \times 2 \]

15 continue

\[ k = 2 \]

\[ j2 = (j1/2) \times 2 \]

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, zi, zp1, zti, rol, wi, asi, hti, th)

\[ py(2) = pp1(m1-96) \]

\[ ty(2) = tt1(m1-96) \]

\[ uy(2) = uu1(m1-96) \]

goto 11

17 \[ j1 = j1/2 \]

\[ dx = dx \times 2 \]

\[ dt = dt \times 2 \]

do 18 ii = 1, (m1-96)

\[ i = ii \]

if \((i, lt, (m1-11))\) 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, zi, zp1, zti, rol, wi, asi, hti, th)

goto 18

19 \[ zx(k) = \left(9 \times tc \div (128 \times tx(k)) - 27 \times tc \times 3 / (64 \times tx(k) \times 3)\right) \times px(k) / pc + 1 \]

\[ zpx(k) = (zx(k) - 1) / px(k) \]

\[ ztx(k) = (1 + tc \times 3 / (64 \times tx(k) \times 4) - 9 \times tc / (128 \times tx(k) \times tx(k))) \times px(k) / pc \]

\[ rox(k) = px(k) / (r \times tx(k) \times zx(k)) \]

\[ wx(k) = \text{dabs(ar \times rox(k) \times fx \times u(k) \times u(k))} / (2 \times d) \]

\[ asx(k) = ((1 + tx(k) \times zx(k) / zx(k)) \times 2) \times px(k) / (rox(k) \times tx(k) \times cp) \]

\[ asx(k) = asx(k) + zpx(k) \times px(k) / zx(k) - 1 \]

\[ asx(k) = asx(k) + zpx(k) \times px(k) / zx(k) - 1 \]

\[ htx(k) = \pi \times cp \times st \times d \times rox(k) \times u(k) \times (tw - tx(k)) \]

\[ if \((i, gt, (m1-97))\) goto 82 \]

pl(i+1)=px(k)

ul(i+1)=ux(k)

tl(i+1)=tx(k)

zl(i+1)=zx(k)

wl(i+1)=wx(k)

zt1(i+1)=ztx(k)

rol(i+1)=rox(k)

asl(i+1)=asx(k)

htl(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, zi, zp1, zti, rol, wi, asi, hti, th)

goto 18

82 \[ dt = dt \times 2 \]

\[ dx = dx \times 2 \]

\[ zy(k) = \left(9 \times tc / (128 \times ty(k)) - 27 \times tc \times 3 / (64 \times ty(k) \times 3)\right) \times py(k) / pc + 1 \]

\[ zpy(k) = (zy(k) - 1) / py(k) \]

\[ zty(k) = (1 + tc \times 3 / (64 \times ty(k) \times 4) - 9 \times tc / (128 \times ty(k) \times ty(k))) \times py(k) / pc \]

\[ roy(k) = py(k) / (r \times ty(k) \times zy(k)) \]

\[ wy(k) = \text{dabs(ar \times roy(k) \times fx \times u(k) \times u(k))} / (2 \times d) \]

\[ asy(k) = ((1 + ty(k) \times zty(k) / zy(k)) \times 2) \times py(k) / (roy(k) \times ty(k) \times cp) \]

\[ asy(k) = asy(k) + zpy(k) \times py(k) / zy(k) - 1 \]

\[ asy(k) = asy(k) + zpy(k) \times py(k) / zy(k) - 1 \]

\[ hty(k) = \pi \times cp \times st \times d \times roy(k) \times u(k) \times (tw - ty(k)) \]

call sub5(p1, t1, u1, dx, pp1, tt1, uu1, tc, pc, r, ar, f, d, cp, pi, st, & tw, g, dt, i, zi, zp1, zti, rol, wi, asi, hti, th, px, ux, tx, wx, zx, & ztx, rox, asx, htx, py, uy, ty, wy, zty, roy, asy, hty, k)

\[ dt = dt \times 2 \]

\[ dx = dx \times 2 \]
k=3
j2=(j1/2)*2
if (j2. eq. j1) goto 20
i=m1-112
px(3)=pi(m1-112)
tx(3)=ti(m1-112)
ux(3)=ui(m1-112)
call sub2(pi, ti, ui, dx, pp1, tt1, uu1, tc, pc, r, ar, f, d, cp, pi, st, & tw, g, dt, i, z1, zp1, zt1, ro1, w1, as1, htl, 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 i=i=1, m1-112
i=i
if(i. lt. (m1-117)) goto 21
if(i. ge. (m1-113)) goto 22
call sub1(pi, ti, ui, dx, pp1, tt1, uu1, tc, pc, r, ar, f, d, cp, pi, st, & tw, g, dt, i, z1, zp1, zt1, ro1, w1, as1, htl, th)
goto 21
22
zx(k)=(9*tc/(128*tx(k))*27*tc**3/(64*tx(k)**3))*px(k)/pc+1
zp(k)=(zx(k)-1)/px(k)
zt(k)=(11*tx(k)*zt(k)/tx(k)**2)*px(k)/(roz(k)*tx(k)*cp)
roz(k)=roz(k)/tx(k)*pz(k)*cy(k)
as(k)=as(k)+roz(k)*cy(k)/cy(k)**2)*px(k)/(roz(k)*tx(k)*cp)
ht(k)=roz(k)*cy(k)/cy(k)**2)*px(k)/(roz(k)*tx(k)*cp)
if(i. gt. (m1-113)) goto 83
pl(i+1)=px(k)
ul(i+1)=ux(k)
tl(i+1)=tx(k)
zl(i+1)=zx(k)
wl(i+1)=wx(k)
ztl(i+1)=zt(k)
rozl(i+1)=roz(k)
asl(i+1)=as(k)
htl(i+1)=ht(k)
call sub1(pi, ti, ui, dx, pp1, tt1, uu1, tc, pc, r, ar, f, d, cp, pi, st, & tw, g, dt, i, z1, zp1, zt1, ro1, w1, as1, htl, 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)=(1*ty(k)-1)/py(k)
zty(k)=(11*ty(k)*zt(k)/ty(k)**2)*py(k)/(roy(k)*ty(k)*pc)
roy(k)=roy(k)/ty(k)*pz(k)*py(k)
as(k)=as(k)+roy(k)*py(k)/ty(k)**2)*py(k)/(roy(k)*ty(k)*cp)
hty(k)=roy(k)*py(k)/ty(k)**2)*py(k)/(roy(k)*ty(k)*cp)
call sub5(pi, ti, ui, dx, pp1, tt1, uu1, tc, pc, r, ar, f, d, cp, pi, st, & tw, g, dt, i, z1, zp1, zt1, ro1, w1, as1, htl, 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,ui,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,ui,as1,ht1,th)
goto 24
25
if(i.gt.(m1-121)) goto 84
p1(i+1)=px(k)
u1(i+1)=ux(k)
tl(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,ui,as1,ht1,th)
goto 24
84
dx=dx/2
zy(k)=(9*tc/(128*tx(k)))-27*tc**3/(64*tx(k)**3))py(k)/pc+1
zpy(k)=(pz(k))/(tx(k)-1)/px(k)
ztx(k)=(6*tc**3/(64*tx(k)**4))-9*tc/(128*tx(k)**3))*py(k)/pc
zro(k)=px(k)/(r*tx(k)+zx(k))
wx(k)=dabs(ar*rox(k)+ux(k))/2d)
asx(k)=((1+tx(k))/zx(k)(**2)*px(k)/(roz(k)**(tx(k))*cp)
asx(k)=asx(k)+zpx(k)/px(k))/zx(k)-1
asx(k)=1/dsqrt(dabs(asx(k)+roz(k)/px(k)))
htx(k)=pi*cp*st*dt*roz(k)**(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,ui,as1,ht1,th)
goto 24
24
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, wi, as1, hti1, 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=i+1
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)=(81*tc**3/(64*tx(k)**4)-9*tc/(128*tx(k)*tx(k)))*px(k)/pc
zty(k)=(81*tc**3/(64*tx(k)**4)-9*tc/(128*tx(k)*tx(k)))*py(k)/pc
ztx(k)=dabs(ar*rox(k)*f*ux(k)*ux(k))/(2*d)

85

if (i. gt. (m1-125)) goto 85

px(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)

pl(i+1)=px(k)
tl(i+1)=tx(k)
u1(i+1)=ux(k)
z1(i+2) = tz(x(5))
wl(i+2) = wx(x(5))
zt1(i+2) = ztx(x(5))
rol(i+2) = rox(x(5))
asl(i+2) = asx(x(5))
htl(i+2) = htx(x(5))
call sub2(p1, t1, u1, dx, ppl, tt1, uu1, tc, pc, r, ar, f, d, cp, pi, st, & tw, g, dt, i, z1, zp1, zt1, rol, wl, asl, htl, th)
py(6) = ppl(m1-126)
ty(6) = ttl(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, rol, wl, asl, htl, th, ppl, ttl, 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
zp(k) = (zx(k) - 1)/px(k)
ztx(k) = (61*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) + zp(k)*px(k)/zx(k)-1
asx(k) = l/sqrt(dabs(asx(k)*rox(k)*ux(k)*(tw-tx(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)
zti(i+1) = ztx(k)
rol(i+1) = rox(k)
asl(i+1) = asx(k)
htl(i+1) = htx(k)
90
call sub1(p1, t1, u1, dx, ppl, tt1, uu1, tc, pc, r, ar, f, d, cp, pi, st, & tw, g, dt, i, z1, zp1, zt1, rol, wl, asl, htl, 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)
zt(k) = (61*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)*zt(k)/zy(k)**2) * py(k)/(roy(k)*ty(k)*cp)
asy(k) = asy(k) + zpy(k)*py(k)/zy(k)-1
asy(k) = l/sqrt(dabs(asy(k)*roy(k)*uy(k)*(tw-ty(k)))
hty(k) = pi*cp*st*d*roy(k)*uy(k)*(tw-ty(k))
call sub5(p1, t1, u1, dx, ppl, tt1, uu1, tc, pc, r, ar, f, d, cp, pi, st, & tw, g, dt, i, z1, zp1, zt1, rol, wl, asl, htl, 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

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(ppl,tt1,uu1,pp2,tt2,uu2,m1,i)
   \ goto \ 33
95 \ pp2(1)=ppl(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,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(ii.le.66) \ goto \ 36
     \ call \ sub4(p2,t2,u2,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,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,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
32  

i=97
pz(2)=p2(i)
uz(2)=u2(i)
tz(2)=t2(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.1e.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)
continue

39 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)
ztz(i-1)=ztz(2)
roz(i-1)=roz(2)
asz(i-1)=asz(2)
htz(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)
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, & zt2, 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)=z2(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)
z2(4)=z2(i)
ztz(4)=ztz(i)
roz(4)=roz(i)
asz(4)=asz(i)
htz(4)=htz(i)
wz(4)=wz(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 i=1, 124
i=ii
if(i.1e.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, & zt2, 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)=z2(4)
w2(i-1)=wz(4)
zt2(i-1)=ztz(4)
as2(i-1)=as2(i)
h2(i-1)=h2(i)
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, h2, th)
k=5
j2=(j1/2)*2
if(j2, eq, j1) goto 47
i=125
p2(5)=p2(i)
u2(5)=u2(i)
t2(5)=t2(i)
z2(5)=z2(i)
z2(5)=z2(i)
ro2(5)=ro2(i)
as2(5)=as2(i)
h2(5)=h2(i)
w2(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, h2, 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, h2, th)
dx=dx*2
dt=dt*2
i=126
p2(i-1)=p2(5)
t2(i-1)=t2(5)
u2(i-1)=u2(5)
z2(i-1)=z2(5)
w2(i-1)=w2(5)
z2(i-1)=z2(5)
ro2(i-1)=ro2(5)
as2(i-1)=as2(5)
h2(i-1)=h2(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, h2, th)
k=6
j2=(j1/2)*2
if(j2, eq, j1) goto 50
i=127
p2(6)=p2(i)
u2(6)=u2(i)
t2(6)=t2(i)
z2(6)=z2(i)
z2(6)=z2(i)
ro2(6)=ro2(i)
as2(6)=as2(i)
h2(6)=h2(i)
w2(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, h2, 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, h2, 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)
w2(i-1)=uz(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
pl(i)=pp1(i)
t1(i)=tt1(i)
u1(i)=uu1(i)
60 continue
do 59 i=1,m2
if(12.eq.0.0) goto 59
p2(i)=pp2(i)
t2(i)=tt2(i)
u2(i)=uu2(i)
59 continue
if(ti.1t.(6*dt1)) goto 70
70 continue
jk=1
if(tis.le.tm) goto 53
print*, '-'
print*, '-'
print*, '-'
stop
end
This subroutine calculates steady isothermal flow along pipe 1. (upstream of the break)

Implicit double precision (a-h, o-z)

dimension pl(300), u1(300), rol(300), zl(300), t1(300)

Print*, 'Initial conditions required:--'
Print*, 'Initial temperature along the pipe (celcius)'
Read*, ti(1)
ti(1)=ti(1)+273.16
Print*, 'Initial pressure at upstream end of pipe (kpa)'
Read*, pi(1)
p1(1)=pi(1)*1000
Print*, 'Mass flow rate through the pipe (kg/s)'
Read*, flow

zl(1)=9*tc/(128*t1(1))-27*tc**3/(64*t1(1)**3)
zl(1)=zl(1)*pi(1)/pc +1
rol(1)=pi(1)/{zl(1)*r*t1(1)}
e=flow/(.25*pi*d**2)

Do 2 i=1, ml-1
If (i.ge.(ml-126)) goto 3
Dx=dxi
1 If((flow eq 0.0) goto 9
aa=f*e*dx/(4.0d0*d)+e
bb=f*e*dx*ul(i)/(4.0d0*d)+dx*rol(i)*g*d*sin(th)/2.0d0
cc=e*dx*g*d*sin(th)/2.0d0+e*r*t1(i)*zl(i)
ul(i+1)=(-bb-dsqrt(bb**2-4*aa*cc))/(2*aa)
rol(i+1)=e/ul(i+1)
p1(i+1)=rol(i+1)*zl(i+1)*r*t1(i)
z1(i+1)=9*tc/(128*t1(i))-27*tc**3/(64*t1(i)**3)
zl(i+1)=zl(i+1)*p1(i+1)/pc +1
t1(i+1)=t1(i)
goto 2
2
9 ul(i+1)=0.0
p1(i+1)=p1(i)-dx1*d*sin(th)
t1(i+1)=t1(i)
zl(i+1)=9*tc/(128*t1(i))-27*tc**3/(64*t1(i)**3)
zl(i+1)=zl(i+1)*p1(i+1)/pc +1
rol(i+1)=p1(i+1)/{zl(i+1)*r*t1(i+1)}
goto 2
3
3 If (i.ge.(ml-124)) goto 4
Dx=dxi/2.
goto 1
4 If (i.ge.(ml-120)) goto 5
Dx=dxi/4.
goto 1
5 If (i.ge.(ml-112)) goto 6
Dx=dxi/8.
goto 1
6 If (i.ge.(ml-96)) goto 7
Dx=dxi/16.
goto 1
7 If (i.ge.(ml-64)) goto 8
goto 1
8 \text{dx=dx1/64.}
goto 1
2 \text{continue}
c
\text{return}
end
This subroutine calculates steady isothermal flow along pipe 2. (downstream of the break)

dimension p2(300), u2(300), ro2(300), z2(300), t2(300)

do 2 i=1,m2-1
  if (i.lt.127) goto 3
  dx=dx1
  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
  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)=z2(i+1)*p2(i+1)/pc + 1
  t2(i+1)=t2(i)
  goto 2
9  u2(i+1)=0.0
  p2(i+1)=p2(i)-dx1*dsin(th)
  t2(i+1)=t2(i)
  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
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
continue
return
end
subroutine sub1(pl, tl, ul, dx, ppl, ttl, wul, tc, pc, r, & ar, f, d, cp, pi, st, tw, g, dt, i, zi, zpl, zti, rol, w1, as1, & htl, th)

This subroutine calculates p, t and u at normal internal points upstream of the break.

implicit double precision (a-h, o-z)
parameter(ni = 300)
dimension pl(ni), tl(ni), ul(ni), zl(ni), & zttl(ni), wul(ni), asl(ni), htl(ni), ppl(ni), & rrol(ni), wwl(ni), asl(ni), htl(ni), & a(9), b(3), rol(ni), zpl(ni)
integer 11(3), mm(3), count

First order approximation

if(ul(i).eq.0.0) goto 20
if(ul(i-1).eq.0.0) goto 20
posq=2*dt/(1/ul(i)+1/ul(i-1))
goto 22

20 posq=dt*(ul(i)+ul(i-1))/2

22 posr=dt*2/(1/(ul(i)+asl(i-1))+1/(ul(i-1)+asl(i-1)))
poss=dt*2/(1/(asl(i)-ul(i-1))+1/(asl(i-1)-ul(i)))

pq=posq/dx*pl(i-1)+(1-posq/dx)*pl(i)
tq=posq/dx*tl(i-1)+(1-posq/dx)*tl(i)
uq=posq/dx*ul(i-1)+(1-posq/dx)*ul(i)
zq=posq/dx*zl(i-1)+(1-posq/dx)*zl(i)
ztq=posq/dx*ztl(i-1)+(1-posq/dx)*ztl(i)
roq=posq/dx*rol(i-1)+(1-posq/dx)*rol(i)
asq=posq/dx*asl(i-1)+(1-posq/dx)*asl(i)
htq=posq/dx*htl(i-1)+(1-posq/dx)*htl(i)
wrq=posq/dx*wul(i-1)+(1-posq/dx)*wul(i)

pr=posr/dx*pl(i-1)+(1-posr/dx)*pl(i)
tr=posr/dx*tl(i-1)+(1-posr/dx)*tl(i)
ur=posr/dx*ul(i-1)+(1-posr/dx)*ul(i)
zr=posr/dx*zl(i-1)+(1-posr/dx)*zl(i)
ztz=posr/dx*ztl(i-1)+(1-posr/dx)*ztl(i)
ror=posr/dx*rol(i-1)+(1-posr/dx)*rol(i)
asr=posr/dx*asl(i-1)+(1-posr/dx)*asl(i)
htz=posr/dx*htl(i-1)+(1-posr/dx)*htl(i)
wsr=posr/dx*wul(i-1)+(1-posr/dx)*wul(i)

x1=asr*dt*(1+tr*ztz/zr)/((ror*cp*tr*ar)
x2=ass*dt*(1+ts*ztz/ztz)/(ros*cp*ts*ar)
a(1)=-(1+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)-wq*ur*x1/(2*pr)+wr*dt/(2*ar*ror*pr)
a(5)=wr*ur*x1/(2*tr)-wr*dt/(2*ar*ror*tr)
a(6)=1/(ros*ass)+ws*us*x2/(2*ps)+ws*dt/(2*ar*ros*ps)
If(Ur.eq.0.0)goto 2
if(uu1.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

b(1)=htq*dt/(roq*cp*ar)+tq-pq*(1+tq*ztq/zq)/(roq*cp)
b(2)=tr*x1+ur+pr/(ror*ass)-g*dt*dsin(th)
b(3)=-hts*x2+us-ps/(ros*ass)-g*dt*dsin(th)
call dminv(a,3, det, 11, mm)
if(det.ne.0.0) goto 9
write(6,30) i
30 format('pipe1 i=',3)
stop "no inverse"
9 ppl(i)=a(1)*b(1)+a(2)*b(2)+a(3)*b(3)
ttl(i)=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
uul(i)=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
count=0
psave=ppl(i)
tsave=ttl(i)
usave=uul(i)
pdif=ppl(i)*1000
tdif=ttl(i)*1000
10 zzi(i)=9*tc/(128*tt1(i))-27*tc**3/(64*tt(i)**3)
zzi(i)=zzi(i)*ppl(i)/pc+1
zzi(i)=zzi(i)-1)/ppl(i)
zzi(i)=zzi(i)+3/(64*tt1(i)**4)-9*tc/(128*tt1(i)**tt1(i))
zzi(i)=zzi(i)-1)/ppl(i)/pc
ror(i)=ppl(i)/(1(tt1(i)*zzi1(i))
ww1(i)=dabs(ar*rrol(i)**#*uul(i)*uul(i))/(2*3)
aasl(i)=(1+zzi1(i)**#tt1(i)/zzi1(i))**2*ppl(i)
aasl(i)=aasl1(i)/(ror(i)*tt1(i)*cp)+zzi1(i)*ppl(i)/zzi1(i)
aasl(i)=(1-aasl1(i))**ror(i)/ppl(i)
aasl(i)=1/dsqrt(dabs(aasl1(i)))
ht1(i)=pi*cp*st*d*ror1(i)*uul(i)*(tw-tt1(i))
count=count + 1
21

Second order procedure

if(uu1.eq.0.0) goto 21
if(uu1(i).eq.0.0) goto 21
posq=2*dt/(1/uu1+1/uul(i))
goto 23
21 posq=dt*(uu+uu1(i))/2
23 posq=2*dt/(1/(asr+asr)+1/(uu1(i)+aasl1(i)))+
posq=2*dt/(1/(ass-us)+1/(aasl1(i)-uu1(i)))

pq=p1(i)-posq*(p1(i+1)-p1(i-1))/(2*dx)
tq=tt1(i)-posq*(tt1(i+1)-tt1(i-1))/(2*dx)
uq=uul(i)-posq*(uul(i+1)-uul(i-1))/(2*dx)
qz=zzi1(i)-posq*(zzi1(i+1)-zzi1(i-1))/(2*dx)
itz=zt1(i)-posq*(zt1(i+1)-zt1(i-1))/(2*dx)
ror=ror1(i)-posq*(ror1(i+1)-ror1(i-1))/(2*dx)
ass=as1(i)-posq*(as1(i+1)-as1(i-1))/(2*dx)
ht=ht1(i)-posq*(ht1(i+1)-ht1(i-1))/(2*dx)
wq=wul(i)-posq*(wul(i+1)-wul(i-1))/(2*dx)
pr=p1(i)-posq*(u1(i+1)-u1(i-1))/(2*dx)
tr=tt1(i)-posq*(tt1(i+1)-tt1(i-1))/(2*dx)
ur=u1(i)-posq*(u1(i+1)-u1(i-1))/(2*dx)
zr=zzi(i)-posq*(zzi(i+1)-zzi(i-1))/(2*dx)
ror=rol(i)-posr*(rol(i+1)-rol(i-1))/(2*dx)
asr=asl(i)-posr*(asl(i+1)-asl(i-1))/(2*dx)
htr=htl(i)-posr*(htl(i+1)-htl(i-1))/(2*dx)
wr=wl(i)-posr*(wl(i+1)-wl(i-1))/(2*dx)
ps=p(i)+posq*(p(i+1)-p(i-1))/(2*dx)
rs=rs+posq*pq*(rs+pq*(pl(i+1)+pl(i-1)-2*pl(i))/(2*dx*dx))

a(1)=-(1+tt(i)*zz(i)/zl(i))/(rrol(i))
a(1)=(a(1)-(1+aosl(i)/2))/(2*cp)
a(2)=1.0
a(3)=0.0
a(4)=(1/(ror*asr))/1/(rrol(i)*aasl(i))/2
a(5)=0.0
a(6)=1.0
a(7)=1/(ror*ass)/1/(rrol(i)*as(i))/2
a(8)=0.0
a(9)=1.0
b(1)=(htq+wq)/(2*cp)
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)
   UUT = A(7) * B(1) + A(8) * B(2) + A(9) * B(3)
   DIFP = DABS(PPI(I) - PIT) / PIT
   DIFT = DABS(TTI(I) - TIT) / TIT
   IF(COUNT.GT.200) GOTO 15
   IF(DIFP.GT.0.01) GOTO 13
   IF(DIFT.LT.0.01) GOTO 14
13 PPI(I) = PIT
   TTI(I) = TIT
   UUI(I) = UUT
   PDIF = DIFP
   TDIF = DIFT
   GOTO 10
15 WRITE(6, 16) I
16 FORMAT('SUB1 - NO ITERATION FOR I=', I4, ' IN PIPE 1')
14 PPI(I) = PIT
   TTI(I) = TIT
   UUI(I) = UUT
   RETURN
END
subroutine sub2(pl, t1, Ul, dx, ppi, ttl, Ul1, tt, pc, c, r, & ar, f, d, cp, pi, st, tw, g, dt, i, zl, zp1, ztl, rol, w1, as1, & htl, th)

this subroutine calculates p, t and u at internal boundary points between different grid sizes.

implicit double precision (a-h, o-z)
dimension pl(300), t1(300), Ul(300), zl(300), & ztl(300), w1(300), as1(300), htl(300), ppi(300), & ttl(300), Uul(300), zzl(300), zzp1(300), zzt1(300), & rrol(300), wwl(300), aas1(300), hhtl(300), & a(9), b(3), rol(300), ppi(300)
integer ii(3), mm(3), count

first order approximation

if(Ul(i).eq.0.0) goto 20
if(Ul(i-1).eq.0.0) goto 20
posq=2*dt/(Uul(i)+1/Uul(i-1))
goto 22
20 posq=dt*(Uul(i)+Uul(i-1))/2
22 posr=dt*2/(1/(Uul(i-1)+as1(i-1))
poss=dt*2/(1/(as1(i-1)-Uul(i-1)))
pq=posq/(2*dx)*pl(i-1)+(1-posq/(2*dx))*pl(i)
tq=posq/(2*dx)*t1(i-1)+(1-posq/(2*dx))*t1(i)
uq=posq/(2*dx)*Ul(i-1)+(1-posq/(2*dx))*Ul(i)
zq=posq/(2*dx)*zl(i-1)+(1-posq/(2*dx))*zl(i)
zq=posq/(2*dx)*zt1(i-1)+(1-posq/(2*dx))*zt1(i)
roq=posq/(2*dx)*rol(i-1)+(1-posq/(2*dx))*rol(i)
asq=posq/(2*dx)*as1(i-1)+(1-posq/(2*dx))*as1(i)
htq=posq/(2*dx)*htl(i-1)+(1-posq/(2*dx))*htl(i)
wq=posq/(2*dx)*wwl(i-1)+(1-posq/(2*dx))*wwl(i)
pr=posr/(2*dx)*pl(i-1)+(1-posq/(2*dx))*pl(i)
tr=posr/(2*dx)*t1(i-1)+(1-posq/(2*dx))*t1(i)
ur=posr/(2*dx)*Ul(i-1)+(1-posq/(2*dx))*Ul(i)
zr=posr/(2*dx)*zl(i-1)+(1-posq/(2*dx))*zl(i)
rtr=posr/(2*dx)*zt1(i-1)+(1-posq/(2*dx))*zt1(i)
ror=posr/(2*dx)*rol(i-1)+(1-posq/(2*dx))*rol(i)
asr=posr/(2*dx)*as1(i-1)+(1-posq/(2*dx))*as1(i)
htr=posr/(2*dx)*htl(i-1)+(1-posq/(2*dx))*htl(i)
wtr=posr/(2*dx)*wwl(i-1)+(1-posq/(2*dx))*wwl(i)
ps=poss/dx*pl(i+1)+(1-poss/dx)*pl(i)
ts=poss/dx*t1(i+1)+(1-poss/dx)*t1(i)
us=poss/dx*Ul(i+1)+(1-poss/dx)*Ul(i)
zs=poss/dx*zl(i+1)+(1-poss/dx)*zl(i)
zt=poss/dx*zt1(i+1)+(1-poss/dx)*zt1(i)
ro=poss/dx*rol(i+1)+(1-poss/dx)*rol(i)
aso=poss/dx*as1(i+1)+(1-poss/dx)*as1(i)
h=poss/dx*htl(i+1)+(1-poss/dx)*htl(i)
w=poss/dx*wwl(i+1)+(1-poss/dx)*wwl(i)

x1=asr*dt*(1+tr*zt1/zr)/(ror*cp+ts*ar)
x2=asr*dt*(1+ts*zt1/zs)/(ros*cp+ts*ar)
a(1)=-(1+q*zq/ztq)/roq*cp-wq*udt/(2*roq*cp*ar*pq)
a(2)=1+wq*udt/(2*roq*cp*ar*tq)
a(3)=-wq*udt/(2*roq*cp*ar)
a(4)=1/(ror*asr)-wr*ur*x1/(2*pr)+wr*udt/(2*ar*ror*pr)
a(5)=wr*ur*x1/(2*tr)-wr*udt/(2*ar*ror*tr)
a(7)=-1/(ros*as)/ws*uts*x2/(2*ps)+ws*udt/(2*ar*ros*ps)
a(8)=-ws*uts*x2/(2*ts)-ws*udt/(2*ar*ros*ts)
if(us. eq. 0.0) goto 2
a(6)=1-w*r*x1+w*r*d*t/(a*r*w*r*u)
a(7)=1+w*s*x2+w*s*d*t/(a*r*w*s*u)
goto 1
2 a(6)=1
a(9)=1
c
b(1)=h*tq*d*t/(r*o*q*c*p*a*r)+t*q-p*q*(1+t*q*z*tq/zq)/(r*o*q*c*p)
b(2)=t*r*x1+u*r+p*r/(r*o*r*a*s*r)-g*d*t*d*sin(th)
b(3)=-h*t*s*x2+v*u-p*s/(r*o*s*a*s*r)-g*d*t*d*sin(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)
ttl(i)=a(4)*b(1)+a(5)*b(2)+a(6)*b(3)
ul1(i)=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
count=0
psave=pp1(i)
tsave=ttl(i)
usave=ul1(i)
pdif=pp1(i)*1000
tdif=ttl(i)*1000
10 zz1(i)=9*t*c/(128*t*t*i(i))-27*t*c**3/(64*t*t*i(i)**3)
zz1(i)=zz1(i)*pp1(i)/pc+1
zzp1(i)=(zz1(i)-1)/pp1(i)
zzt1(i)=81*t*c**3/(64*t*t*i(i)**3)-9*t*c/(128*t*t*i(i)*tt1(i))
zzt1(i)=zzt1(i)*pp1(i)/pc
rrol(i)=pp1(i)/(r*t*t*i(i)**zz1(i))
wwul(i)=dabs(a*r*w*r*wul1(i)*f*uul1(i)*uul1(i))/(2*d)
aasl1(i)=(1+zz1(i)*tt1(i)/zz1(i))**2*pp1(i)
aasl1(i)=(1-aasl1(i))*rrol(i)**tt1(i)*cp)+zzp1(i)*pp1(i)/zz1(i)
aasl1(i)=1/dl*n*d*r*r*w*r*wul1(i)*uul1(i)*tt1(i)
count=count+1

second order procedure

if(uq. eq. 0.0) goto 21
if(uu1(i), eq. 0.0) goto 21
posq=2*d*t/(1/uq+1/uu1(i))
goto 23
21 posq=d*t*(uq+uu1(i))/2
23 posr=2*d*t/(1/(u*r+a*s*r)+1/(uu1(i)+aasl1(i)))
poss=2*d*t/(1/(a*s-s-u)+1/(aasl1(i)-uu1(i)))

pq=pp1(i-1)+2*d*x-posq)*pp1(i-1-p1(i-2))/(4*d*x)
uq=uu1(i-1)+2*d*x-posq)*uu1(i-1-u(i-2))/(4*d*x)
tq=tt1(i-1)+2*d*x-posq)*tt1(i-1-t(i-2))/(4*d*x)
zq=z1(i-1)+2*d*x-posq)*z1(i-1-z1(i-2))/(4*d*x)
wq=wl1(i-1)+2*d*x-posq)*wl1(i-1-w1(i-2))/(4*d*x)
ztt=zt1(i-1)+2*d*x-posq)*zt1(i-1-t1(i-2))/(4*d*x)
roq=rol1(i-1)+2*d*x-posq)*rol1(i-1-rol(i-2))/(4*d*x)
asq=asl1(i-1)+2*d*x-posq)*asl1(i-1-asl1(i-2))/(4*d*x)
htq=ht1(i-1)+2*d*x-posq)*ht1(i-1-ht1(i-2))/(4*d*x)
pr=pp1(i-1)+2*d*x-posq)*pp1(i-1-p1(i-1))/(4*d*x)
uu=uu1(i-1)+2*d*x-posq)*uu1(i-1-u(i-2))/(4*d*x)
ttt=tt1(i-1)+2*d*x-posq)*tt1(i-1-t1(i-2))/(4*d*x)
ztt=z1(i-1)+2*d*x-posq)*z1(i-1-z1(i-2))/(4*d*x)
wtt=wl1(i-1)+2*d*x-posq)*wl1(i-1-w1(i-2))/(4*d*x)

250
\[ \frac{\text{asr} = \text{asl}(i-1) + (2*\text{dx} \cdot \text{posr}) \cdot (\text{rol}(1-i) - \text{rol}(i-2))}{(4*\text{dx})} \]

\[ \frac{\text{htr} = \text{htl}(1-i) + (2*\text{dx} \cdot \text{posr}) \cdot (\text{asl}(1-i) - \text{asl}(2-i))}{(4*\text{dx})} \]

\[ p = \frac{1+i}{\text{dx} \cdot \text{posr}} \cdot \left[ p(i+1) - p(i) \right] / (2*\text{dx}) \]

\[ \frac{\text{us} = \text{ul}(i) - (2*\text{dx} \cdot \text{posr}) \cdot \left[ u(i+1) - u(i) \right] / (2*\text{dx})}{(2*\text{dx})} \]

\[ \text{ts} = \frac{t(i+2) - t(i)}{(2*\text{dx})} \]

\[ z = z(i+1) - (2*\text{dx} \cdot \text{posr}) \cdot \left[ zl(i+2) - zl(i) \right] / (2*\text{dx}) \]

\[ w = w(i) + \left[ (u(i+1) - 2*\text{dx} \cdot \text{posr}) \cdot \left[ \text{ul}(i+2) - \text{ul}(i) \right] / (2*\text{dx}) \right] / (2*\text{dx}) \]

\[ zt = z(i+1) + (2*\text{dx} \cdot \text{posr}) \cdot \left[ zl(i+2) - zl(i) \right] / (2*\text{dx}) \]

\[ htr = \frac{1+i}{\text{dx} \cdot \text{posr}} \cdot \left[ htl(1-i) + \text{htl}(i) \right] / (2*\text{dx}) \]

\[ \text{ps} = \frac{p(i+1) + p(i)}{(2*\text{dx} \cdot \text{posr})} \cdot \left[ u(i+1) + u(i) \right] / (2*\text{dx}) \]

\[ t = \frac{t(i+1) - t(i)}{(2*\text{dx} \cdot \text{posr})} \cdot \left[ u(i+1) - u(i) \right] / (2*\text{dx}) \]

\[ z = z(i+1) \]

\[ w = w(i) \]

\[ zt = z(i+1) \]

\[ htr = \frac{1+i}{\text{dx} \cdot \text{posr}} \cdot \left[ htl(1-i) + \text{htl}(i) \right] / (2*\text{dx}) \]

\[ \text{ps} = \frac{p(i+1) + p(i)}{(2*\text{dx} \cdot \text{posr})} \cdot \left[ u(i+1) + u(i) \right] / (2*\text{dx}) \]

\[ \text{us} = \frac{u(i+1) + u(i)}{(2*\text{dx} \cdot \text{posr})} \cdot \left[ u(i+1) + u(i) \right] / (2*\text{dx}) \]

\[ ts = \frac{t(i+1) + t(i)}{(2*\text{dx} \cdot \text{posr})} \cdot \left[ u(i+1) + u(i) \right] / (2*\text{dx}) \]

\[ \text{zs} = \frac{z(i+1) + z(i)}{(2*\text{dx} \cdot \text{posr})} \cdot \left[ zl(i+1) + zl(i) \right] / (2*\text{dx}) \]

\[ w = \frac{w(i+1) + w(i)}{(2*\text{dx} \cdot \text{posr})} \cdot \left[ u(i+1) + u(i) \right] / (2*\text{dx}) \]

\[ \text{zt} = \frac{z(i+1) + z(i)}{(2*\text{dx} \cdot \text{posr})} \cdot \left[ zl(i+1) + zl(i) \right] / (2*\text{dx}) \]
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 (tit(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
tit(i) = tit
uit(i) = uit
pdif = difp
tdif = dift
goto 10
15 write (6, 16) i
16 format ('sub2 - no iteration for i=', i4, ' in pipe 1 (sub2)')
14 pp1(i) = pit
tit(i) = tit
uit(i) = uit
return
end
This subroutine calculates \( p, t \) and \( u \) at internal boundary points between different grid sizes in pipe 2.

**First order approximation**

\[
\begin{align*}
\text{posq} &= \frac{dt}{u_2(i)} + \frac{1}{u_2(i-1) + u_2(i+1)} \\
\text{pq} &= \text{posq}/dx * p_2(i-1) + (1 - \text{posq}/dx) * p_2(i) \\
\text{tq} &= \text{posq}/dx * t_2(i-1) + (1 - \text{posq}/dx) * t_2(i) \\
\text{uq} &= \text{posq}/dx * u_2(i-1) + (1 - \text{posq}/dx) * u_2(i) \\
\text{zq} &= \text{posq}/dx * z_2(i-1) + (1 - \text{posq}/dx) * z_2(i)
\end{align*}
\]

\[
\begin{align*}
\text{ps} &= \text{posq}/(2*dx) * p_2(i-1) + (1 - \text{posq}/(2*dx)) * p_2(i) \\
\text{ts} &= \text{posq}/(2*dx) * t_2(i-1) + (1 - \text{posq}/(2*dx)) * t_2(i) \\
\text{us} &= \text{posq}/(2*dx) * u_2(i-1) + (1 - \text{posq}/(2*dx)) * u_2(i) \\
\text{zs} &= \text{posq}/(2*dx) * z_2(i-1) + (1 - \text{posq}/(2*dx)) * z_2(i)
\end{align*}
\]

if \( u_2(i) \leq 0.0 \) goto 7
if \( u_2(i-1) \leq 0.0 \) goto 17
\]

if \( u_2(i) \leq 0.0 \) goto 1
if \( u_2(i-1) \leq 0.0 \) goto 7
goto 17
\]

7 \[
\begin{align*}
\text{pq} &= \text{posq}/dx * p_2(i-1) + (1 - \text{posq}/dx) * p_2(i) \\
\text{tq} &= \text{posq}/dx * t_2(i-1) + (1 - \text{posq}/dx) * t_2(i) \\
\text{uq} &= \text{posq}/dx * u_2(i-1) + (1 - \text{posq}/dx) * u_2(i) \\
\text{zq} &= \text{posq}/dx * z_2(i-1) + (1 - \text{posq}/dx) * z_2(i)
\end{align*}
\]
\begin{verbatim}
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 p4=posq/(2*dx)*p2(i+1)+(1-posq/(2*dx))*p2(i)
t4=posq/(2*dx)*t2(i+1)+(1-posq/(2*dx))*t2(i)
u4=posq/(2*dx)*u2(i+1)+(1-posq/(2*dx))*u2(i)
z4=posq/(2*dx)*z2(i+1)+(1-posq/(2*dx))*z2(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)
rt4=posq/(2*dx)*ro2(i+1)+(1-posq/(2*dx))*ro2(i)
as4=posq/(2*dx)*as2(i+1)+(1-posq/(2*dx))*as2(i)
ht4=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)
     goto 3
3 a(1)=-(1+tt4*zt4/z4)/(roq*ct)-wq*uq*dt/(2*roq*cp*ar*pq)
a(2)=1+wq*uq*dt/(2*roq*cp*ar*tq)
a(3)=-wq*dt/(roq*ct*ar)
b(1)=htq*dt/(roq*cp*ar)+tq-pq*(1+tt4*zt4/z4)/(roq*cp)
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
10 z22(i)=9*tc/(128*tt2(i))-27*tc**3/(64*tt2(i)**3)
z22(i)=z22(i)*pp2(i)/pc+1
z22(i)=z22(i)/pp2(i)
zt2(i)=81*tc*3/(64*tt2(i)**4)-9*tc/(128*tt2(i)**2)*tt2(i)
zt2(i)=zt2(i)/pp2(i)/pc
ro2(i)=pp2(i)/(rt2(i)*z22(i))
ru2(i)=dabs(ar*ro2(i)**fu22(i)*uu2(i))/(2*dd)
aas2(i)=(1+z22(i)*tt2(i)/z22(i)**2)**2*pp2(i)
aas2(i)=aas2(i)/(ro2(i)**tt2(i)*cp)*zzp2(i)*pp2(i)/z22(i)
aas2(i)=(1-aas2(i))**ro2(i)/pp2(i)
aas2(i)=1/dsqrt(dabs(aas2(i)))
ht2(i)=pi*cp*st*d*ro2(i)***uu2(i)**(tw-tt2(i))
count=count+1
end

Second order procedure

if(uq.eq.0.0) goto 8
if(uu2(i).eq.0.0) goto 8
posq=dabs(dt*2/(1/uq+a2+1/uu2(i)))
goto 11
8 posr=2*dt/(1/(ur+asr)+1/(uu2(i)+aas2(i)))
11 posr=2*dt/(1/(ur+asr)+1/(uu2(i)+aas2(i)))
end
\end{verbatim}
\[\begin{align*}
pr &= p_2(i-1) + (dx - pos) * (p_2(i) - p_2(i-2)) / (2*dx) \\
uq &= u_2(i-1) + (dx - pos) * (u_2(i) - u_2(i-2)) / (2*dx) \\
tq &= t_2(i-1) + (dx - pos) * (t_2(i) - t_2(i-2)) / (2*dx) \\
qz &= z_2(i-1) + (dx - pos) * (z_2(i) - z_2(i-2)) / (2*dx) \\
wr &= w_2(i-1) + (dx - pos) * (w_2(i) - w_2(i-2)) / (2*dx) \\
zt &= z_2(i-1) + (dx - pos) * (zt_2(i) - z_2(i-2)) / (2*dx) \\
rro &= r_2(i-1) + (dx - pos) * (ro_2(i) - ro_2(i-2)) / (2*dx) \\
asr &= as_2(i-1) + (dx - pos) * (as_2(i) - as_2(i-2)) / (2*dx) \\
ht &= h_2(i-1) + (dx - pos) * (h_2(i) - h_2(i-2)) / (2*dx) \\
ps &= p_2(i-1) - (2*dx - pos) * (p_2(i+2) - p_2(i)) / (4*dx) \\
us &= u_2(i+1) - (2*dx - pos) * (u_2(i+2) - u_2(i)) / (4*dx) \\
ts &= t_2(i+1) - (2*dx - pos) * (t_2(i+2) - t_2(i)) / (4*dx) \\
zs &= z_2(i+1) - (2*dx - pos) * (z_2(i+2) - z_2(i)) / (4*dx) \\
wzs &= w_2(i+2) - w_2(i-2) / (4*dx) \\
zt &= z_2(i+1) - (2*dx - pos) * (zt_2(i+2) - z_2(i)) / (4*dx) \\
rzs &= r_2(i+1) - (2*dx - pos) * (ro_2(i+2) - ro_2(i)) / (4*dx) \\
aszs &= as_2(i+1) - (2*dx - pos) * (as_2(i+2) - as_2(i)) / (4*dx) \\
hts &= h_2(i+1) - (2*dx - pos) * (h_2(i+2) - h_2(i)) / (4*dx) \\
pr &= p_2(i+1) - (2*dx - pos) * (p_2(i+2) - p_2(i-1)) / (2*dx) \\
uq &= u_2(i+1) - (2*dx - pos) * (u_2(i+2) - u_2(i-1)) / (2*dx) \\
tq &= t_2(i+1) - (2*dx - pos) * (t_2(i+2) - t_2(i-1)) / (2*dx) \\
qz &= z_2(i+1) - (2*dx - pos) * (z_2(i+2) - z_2(i-1)) / (2*dx) \\
wr &= w_2(i+1) - (2*dx - pos) * (w_2(i+2) - w_2(i-1)) / (2*dx) \\
zt &= z_2(i+1) - (2*dx - pos) * (zt_2(i+2) - z_2(i-1)) / (2*dx) \\
rro &= r_2(i+1) - (2*dx - pos) * (ro_2(i+2) - ro_2(i-1)) / (2*dx) \\
asr &= as_2(i+1) - (2*dx - pos) * (as_2(i+2) - as_2(i-1)) / (2*dx) \\
ht &= h_2(i+1) - (2*dx - pos) * (h_2(i+2) - h_2(i-1)) / (2*dx) \\
ps &= p_2(i+1) - (2*dx - pos) * (p_2(i+2) - p_2(i-1)) / (2*dx) \\
us &= u_2(i+1) - (2*dx - pos) * (u_2(i+2) - u_2(i-1)) / (2*dx) \\
ts &= t_2(i+1) - (2*dx - pos) * (t_2(i+2) - t_2(i-1)) / (2*dx) \\
zs &= z_2(i+1) - (2*dx - pos) * (z_2(i+2) - z_2(i-1)) / (2*dx) \\
wzs &= w_2(i+2) - w_2(i-2) / (4*dx) \\
zt &= z_2(i+1) - (2*dx - pos) * (zt_2(i+2) - z_2(i-1)) / (2*dx) \\
rzs &= r_2(i+1) - (2*dx - pos) * (ro_2(i+2) - ro_2(i-1)) / (2*dx) \\
aszs &= as_2(i+1) - (2*dx - pos) * (as_2(i+2) - as_2(i-1)) / (2*dx) \\
hts &= h_2(i+1) - (2*dx - pos) * (h_2(i+2) - h_2(i-1)) / (2*dx) \\
if((uu2(i)+uq).lt.0.0) goto 4 \\
pq &= p_2(i+1) - (dx - pos) * (p_2(i+2) - p_2(i-2)) / (2*dx) \\
uq &= u_2(i+1) - (dx - pos) * (u_2(i+2) - u_2(i-2)) / (2*dx) \\
tq &= t_2(i+1) - (dx - pos) * (t_2(i+2) - t_2(i-2)) / (2*dx) \\
qz &= z_2(i+1) - (dx - pos) * (z_2(i+2) - z_2(i-2)) / (2*dx) \\
wr &= w_2(i+1) - (dx - pos) * (w_2(i+2) - w_2(i-2)) / (2*dx) \\
zt &= z_2(i+1) - (dx - pos) * (zt_2(i+2) - z_2(i-2)) / (2*dx) \\
rro &= r_2(i+1) - (dx - pos) * (ro_2(i+2) - ro_2(i-2)) / (2*dx) \\
asr &= as_2(i+1) - (dx - pos) * (as_2(i+2) - as_2(i-2)) / (2*dx) \\
ht &= h_2(i+1) - (dx - pos) * (h_2(i+2) - h_2(i-2)) / (2*dx) \\
ps &= p_2(i+1) - (dx - pos) * (p_2(i+2) - p_2(i-1)) / (2*dx) \\
us &= u_2(i+1) - (dx - pos) * (u_2(i+2) - u_2(i-1)) / (2*dx) \\
tq &= t_2(i+1) - (dx - pos) * (t_2(i+2) - t_2(i-1)) / (2*dx) \\
qz &= z_2(i+1) - (dx - pos) * (z_2(i+2) - z_2(i-1)) / (2*dx) \\
wr &= w_2(i+1) - (dx - pos) * (w_2(i+2) - w_2(i-1)) / (2*dx) \\
zt &= z_2(i+1) - (dx - pos) * (zt_2(i+2) - z_2(i-1)) / (2*dx) \\
rro &= r_2(i+1) - (dx - pos) * (ro_2(i+2) - ro_2(i-1)) / (2*dx) \\
asr &= as_2(i+1) - (dx - pos) * (as_2(i+2) - as_2(i-1)) / (2*dx) \\
ht &= h_2(i+1) - (dx - pos) * (h_2(i+2) - h_2(i-1)) / (2*dx) \\
goto 4 \\
pq &= p_2(i+1) - (2*dx - pos) * (p_2(i+1) - p_2(i-2)) / (2*dx) \\
uq &= u_2(i+1) - (2*dx - pos) * (u_2(i+2) - u_2(i-2)) / (2*dx) \\
tq &= t_2(i+1) - (2*dx - pos) * (t_2(i+2) - t_2(i-2)) / (2*dx) \\
qz &= z_2(i+1) - (2*dx - pos) * (z_2(i+2) - z_2(i-2)) / (2*dx) \\
wr &= w_2(i+1) - (2*dx - pos) * (w_2(i+2) - w_2(i-2)) / (2*dx) \\
zt &= z_2(i+1) - (2*dx - pos) * (zt_2(i+2) - z_2(i-2)) / (2*dx) \\
rro &= r_2(i+1) - (2*dx - pos) * (ro_2(i+2) - ro_2(i-2)) / (2*dx) \\
asr &= as_2(i+1) - (2*dx - pos) * (as_2(i+2) - as_2(i-2)) / (2*dx) \\
ht &= h_2(i+1) - (2*dx - pos) * (h_2(i+2) - h_2(i-2)) / (2*dx) \\
goto 6 \\
\end{align*}\]
\[\begin{align*}
roq &= ro2(i+1)-(2*dx-psq)*(ro2(i+2)-ro2(i))/(4*dx) \\
asq &= as2(i+1)-(2*dx-psq)*(as2(i+2)-as2(i))/(4*dx) \\
h tq &= h t2(i+1)-(2*dx-psq)*(ht2(i+2)-ht2(i))/(4*dx) \\
pq &= pq+(2*dx-psq)**2*(p2(i+2)+p2(i)-2*p2(i+1))/(8*dx*dx) \\
q &= q+(2*dx-psq)**2*(z2(i+2)+z2(i)-2*z2(i+1))/(8*dx*dx) \\
wq &= wq+(2*dx-psq)**2*(w2(i+2)+w2(i)-2*w2(i+1))/(8*dx*dx) \\
roq &= roq+(2*dx-psq)**2*(ro2(i+2)+ro2(i)-2*ro2(i+1))/(8*dx*dx) \\
asq &= asq+(2*dx-psq)**2*(as2(i+2)+as2(i)-2*as2(i+1))/(8*dx*dx) \\
htq &= htq+(2*dx-psq)**2*(ht2(i+2)+ht2(i)-2*ht2(i+1))/(8*dx*dx) \\
\end{align*}\]

\[\begin{align*}
a(1) &= -(1+tt2(i)*zt2(i))/ro2(i) \\
a(1) &= a(1)-(1+tt2(i)/i)/ro2(i)/(2*cp) \\
a(2) &= 1.0 \\
a(3) &= 0.0 \\
a(4) &= (1/(ror*asr)+1/(ror2(i)*as2(i)))/2 \\
a(5) &= 0.0 \\
a(6) &= 1.0 \\
a(7) &= (1/(ros*ass)-1/(ror2(i)*as2(i)))/2 \\
a(8) &= 0.0 \\
a(9) &= 1.0 \\
\end{align*}\]

\[\begin{align*}
b(1) &= (htq+wq*sq)/roq+hht2(i)+ww2(i)/uu2(i))/ro2(i) \\
b(1) &= b(1)*dt/(2*cp*ar+a)*bq+ttq \\
b(2) &= (1+tt2(i)*zt2(i))/ro2(i)+wtt2(i)+ww2(i)*uu2(i)) \\
b(2) &= b(2)*as2(i)/(ror2(i)*t2(i)) \\
b(2) &= (b(1)*qs+i*tz2(i))/zt2(i) \\
b(2) &= b(2)*as2(i)/(ror2(i)*zt2(i)) \\
b(2) &= b(2)+a*gt2*dsin(th)+a(4)*pr+ur \\
b(3) &= (1+tt2(i)*zt2(i))/zt2(i) \\
b(3) &= b(3)*as2(i)/(ror2(i)*zt2(i)) \\
b(3) &= (b(1)*qs+i*tz2(i))/zt2(i) \\
b(3) &= (b(1)-w2(i)/ro2(i))/ro2(i)) \\
b(3) &= b(3)-g*dsin(th)+a(7)*ps+us \\
\end{align*}\]

call dminv(a, 3, det, 11, mm) 
if(det.ne.0.0) goto 12 
write(6, 31) 
31 format('pipe 2 (2nd order) i=', i, 3) 
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(difpt.gt.0.01) goto 13 
if(diftl.t.0.01) goto 14 
13 pp2(i) = pit 
tt2(i) = tit 
uu2(i) = uit 
difp = difp 
tdif = dift 
goto 10 
c 
15 write(6, 16) 
16 format('sub3 - no iteration for i= ', i, ' in pipe 2') 
14 pp2(i) = pit 
tt2(i) = tit 
uu2(i) = uit 
return 
end
This subroutine calculates $p$, $t$ and $u$ at normal internal points downstream of the break.

First order approximation

$$\begin{align*}
\text{pr} &= \text{posr}/dx*p2(i-1)+1/\text{posr}/dx*p2(i) \\
\text{tr} &= \text{posr}/dx*t2(i-1)+1/\text{posr}/dx*t2(i) \\
\text{ur} &= \text{posr}/dx*u2(i-1)+1/\text{posr}/dx*u2(i) \\
\text{zr} &= \text{posr}/dx*z2(i-1)+1/\text{posr}/dx*z2(i) \\
\text{ztr} &= \text{posr}/dx*zt2(i-1)+1/\text{posr}/dx*zt2(i) \\
\text{rpr} &= \text{posr}/dx*ro2(i-1)+1/\text{posr}/dx*ro2(i) \\
\text{apr} &= \text{posr}/dx*as2(i-1)+1/\text{posr}/dx*as2(i) \\
\text{htr} &= \text{posr}/dx*ht2(i-1)+1/\text{posr}/dx*ht2(i) \\
\text{wr} &= \text{posr}/dx*w2(i-1)+1/\text{posr}/dx*w2(i) \\
\text{ps} &= \text{poss}/dx*p2(i+1)+1/\text{poss}/dx*p2(i) \\
\text{ts} &= \text{poss}/dx*t2(i+1)+1/\text{poss}/dx*t2(i) \\
\text{us} &= \text{poss}/dx*u2(i+1)+1/\text{poss}/dx*u2(i) \\
\text{zs} &= \text{poss}/dx*z2(i+1)+1/\text{poss}/dx*z2(i) \\
\text{zt} &= \text{poss}/dx*zt2(i+1)+1/\text{poss}/dx*zt2(i) \\
\text{ros} &= \text{poss}/dx*ro2(i+1)+1/\text{poss}/dx*ro2(i) \\
\text{ass} &= \text{poss}/dx*as2(i+1)+1/\text{poss}/dx*as2(i) \\
\text{hts} &= \text{poss}/dx*ht2(i+1)+1/\text{poss}/dx*ht2(i) \\
\text{ws} &= \text{poss}/dx*w2(i+1)+1/\text{poss}/dx*w2(i) \\
\text{x1} &= \text{asr}/dt*(1+tr*zt2/zr)/(ror*cp*tr*ar) \\
\text{x2} &= \text{ass}/dt*(1+ts*zt2/zs)/(ros*cp*ts*ar) \\
a(4) &= 1/(ror*asr)-w*r*x1/(2*pr)+w*r*dt/(2*ar*ror*pr) \\
a(5) &= w*r*x1/(2*tr)-w*r*dt/(2*ar*ror*tr) \\
a(7) &= -1/(ros*ass)+w*s*x2/(2*ps)+w*s*dt/(2*ar*ros*ts) \\
a(8) &= w*s*x2/(2*ts)-w*s*dt/(2*ar*ros*ts) \\
\text{if} (\text{ur} . \text{eq.} . 0.0) \text { goto 20 } \\
\text{if} (\text{us} . \text{eq.} . 0.0) \text { goto 20 } \\
a(6) &= 1-w*r*x1+w*r*dt/(ar*ror*ur) \\
a(9) &= 1+w*s*x2+w*s*dt/(ar*ros*us) \\
\text{goto 21} \\
\text{20} \\
a(6) &= 1 \\
a(9) &= 1 \\
\text{21} \\
b(2) &= \text{htr}+x1+ur+pr/(\text{ror}*\text{asr})-g*dt*dsin(th) \\
b(3) &= -\text{hts}+xs+us-ps/(\text{ros}*-\text{ass})-g*dt*dsin(th) \\
\text{if} (u2(i) . \text{le.} 0.0) \text { goto 1 } \\
\text{if} (u2(i-1) . \text{eq.} . 0.0) \text { goto 1 } \\
\text{posq} &= \text{psq}/dx*p2(i-1)+1/\text{posq}/dx*p2(i) \\
\text{tq} &= \text{psq}/dx*t2(i-1)+1/\text{posq}/dx*t2(i) \\
uq &= \text{psq}/dx*u2(i-1)+1/\text{posq}/dx*u2(i) \\
\text{zq} &= \text{psq}/dx*z2(i-1)+1/\text{posq}/dx*z2(i) \\
ztr &= \text{psq}/dx*zt2(i-1)+1/\text{psq}/dx*zt2(i) \\
\text{roq} &= \text{psq}/dx*ro2(i-1)+1/\text{psq}/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

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)

3 a(1) = -(1+ztq*ztq/zq)/(roq*cp) - wq*uq*dt/(roq*cp*ar*pq)
a(2) = 1+wq*uq*dt/(zo*cp*ar*tq)
a(3) = -wq*dt/(roq*cp*ar)
b(1) =httq*dt/(roq*cp*ar)+tq-pq*(1+ztq/ztq/zq)/(roq*cp)

5 call dminv(a, 3, det, 11, mm)
   if(det.ne.0.0) goto 9
   write(6, 30) i
   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)
tssave = tt2(i)
usave = uu2(i)
pdif = pp2(i)-1000
tdif = tt2(i)*1000

10 zzo(i) = 9*tc/((128*tt2(i))**3)-(27*tc**3)/(64*tt2(i)**3)
    zzo(i) = zzo(i)*pp2(i)/pc+1
    zzp2(i) = (zzo(i)-1)/pp2(i)
    zzt2(i) = 0.1*tc**3/(4*tt2(i)**4)-9*tc/(128*tt2(i)**tt2(i))
    zzt2(i) = zzt2(i)*pp2(i)/pc
    rro2(i) = pp2(i)/(rtt2(i)+zzo(i))
    uu2(i) = dabs(ar*rro2(i)*f*uu2(i)*uu2(i))/(2*d)
    aas2(i) = ((1+zzo(i)*tt2(i)/zzo2(i)**2)*pp2(i)
    aas2(i) = aas2(i)/(rro2(i)*tt2(i)*cp)+zzp2(i)*pp2(i)/zzo2(i)
    aas2(i) = (1-aas2(i))*rro2(i)/pp2(i)
    aas2(i) = 1/dsqrt(dabs(aas2(i)))
    htt2(i) = pi*cp*st*d*rro2(i)*uu2(i)*(tw-tt2(i))
count = count+1

second order procedure

if(uq.eq.0.0) goto 8
if(uu2(i).eq.0.0) goto 8
   posq = dabs(dt**(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)
\[ ur = u2(i) + posr * (u2(i+1) - u2(i-1)) / (2*dx) \]
\[ tr = z2(i) + posr * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ ztr = z2(i) + posr * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ ror = ro2(i) + posr * (ro2(i+1) - ro2(i-1)) / (2*dx) \]
\[ asr = as2(i) + posr * (as2(i+1) - as2(i-1)) / (2*dx) \]
\[ htr = ht2(i) + posr * (ht2(i+1) - ht2(i-1)) / (2*dx) \]
\[ wr = w2(i) + posr * (w2(i+1) - w2(i-1)) / (2*dx) \]
\[ pq = p2(i) + posr * (p2(i+1) - p2(i-1)) / (2*dx) \]
\[ tq = t2(i) + posr * (t2(i+1) - t2(i-1)) / (2*dx) \]
\[ uq = u2(i) + posr * (u2(i+1) - u2(i-1)) / (2*dx) \]
\[ zq = z2(i) + posr * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ ztq = z2(i) + posr * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ rqs = ro2(i) + posr * (ro2(i+1) - ro2(i-1)) / (2*dx) \]
\[ asq = as2(i) + posr * (as2(i+1) - as2(i-1)) / (2*dx) \]
\[ hts = ht2(i) + posr * (ht2(i+1) - ht2(i-1)) / (2*dx) \]
\[ wts = w2(i) + posr * (w2(i+1) - w2(i-1)) / (2*dx) \]
\[ pq = p2(i) + posr * (p2(i+1) - p2(i-1)) / (2*dx) \]
\[ tq = t2(i) + posr * (t2(i+1) - t2(i-1)) / (2*dx) \]
\[ uq = u2(i) + posr * (u2(i+1) - u2(i-1)) / (2*dx) \]
\[ zq = z2(i) + posr * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ ztq = z2(i) + posr * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ rqs = ro2(i) + posr * (ro2(i+1) - ro2(i-1)) / (2*dx) \]
\[ asq = as2(i) + posr * (as2(i+1) - as2(i-1)) / (2*dx) \]
\[ hts = ht2(i) + posr * (ht2(i+1) - ht2(i-1)) / (2*dx) \]
\[ wts = w2(i) + posr * (w2(i+1) - w2(i-1)) / (2*dx) \]
\[ pq = p2(i) + posr * (p2(i+1) - p2(i-1)) / (2*dx) \]
\[ tq = t2(i) + posr * (t2(i+1) - t2(i-1)) / (2*dx) \]
\[ uq = u2(i) + posr * (u2(i+1) - u2(i-1)) / (2*dx) \]
\[ zq = z2(i) + posr * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ ztq = z2(i) + posr * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ rqs = ro2(i) + posr * (ro2(i+1) - ro2(i-1)) / (2*dx) \]
\[ asq = as2(i) + posr * (as2(i+1) - as2(i-1)) / (2*dx) \]
\[ hts = ht2(i) + posr * (ht2(i+1) - ht2(i-1)) / (2*dx) \]
\[ wts = w2(i) + posr * (w2(i+1) - w2(i-1)) / (2*dx) \]

if \((uu2(i) + uq) < 0.0\) goto 4

\[ pq = p2(i) + posq * (p2(i+1) - p2(i-1)) / (2*dx) \]
\[ tq = t2(i) + posq * (t2(i+1) - t2(i-1)) / (2*dx) \]
\[ uq = u2(i) + posq * (u2(i+1) - u2(i-1)) / (2*dx) \]
\[ zq = z2(i) + posq * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ ztq = z2(i) + posq * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ rqs = ro2(i) + posq * (ro2(i+1) - ro2(i-1)) / (2*dx) \]
\[ asq = as2(i) + posq * (as2(i+1) - as2(i-1)) / (2*dx) \]
\[ hts = ht2(i) + posq * (ht2(i+1) - ht2(i-1)) / (2*dx) \]
\[ wts = w2(i) + posq * (w2(i+1) - w2(i-1)) / (2*dx) \]

goto 6

\[ pq = p2(i) + posq * (p2(i+1) - p2(i-1)) / (2*dx) \]
\[ tq = t2(i) + posq * (t2(i+1) - t2(i-1)) / (2*dx) \]
\[ uq = u2(i) + posq * (u2(i+1) - u2(i-1)) / (2*dx) \]
\[ zq = z2(i) + posq * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ ztq = z2(i) + posq * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ rqs = ro2(i) + posq * (ro2(i+1) - ro2(i-1)) / (2*dx) \]
\[ asq = as2(i) + posq * (as2(i+1) - as2(i-1)) / (2*dx) \]
\[ hts = ht2(i) + posq * (ht2(i+1) - ht2(i-1)) / (2*dx) \]
\[ wts = w2(i) + posq * (w2(i+1) - w2(i-1)) / (2*dx) \]

goto 8

\[ pq = p2(i) + posq * (p2(i+1) - p2(i-1)) / (2*dx) \]
\[ tq = t2(i) + posq * (t2(i+1) - t2(i-1)) / (2*dx) \]
\[ uq = u2(i) + posq * (u2(i+1) - u2(i-1)) / (2*dx) \]
\[ zq = z2(i) + posq * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ ztq = z2(i) + posq * (z2(i+1) - z2(i-1)) / (2*dx) \]
\[ rqs = ro2(i) + posq * (ro2(i+1) - ro2(i-1)) / (2*dx) \]
\[ asq = as2(i) + posq * (as2(i+1) - as2(i-1)) / (2*dx) \]
\[ hts = ht2(i) + posq * (ht2(i+1) - ht2(i-1)) / (2*dx) \]
\[ wts = w2(i) + posq * (w2(i+1) - w2(i-1)) / (2*dx) \]
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)

a(1) = -(1 + tt2(i) * zz2(i) / zz2(i)) / rro2(i)
a(1) = (a(1) - (1 + tq * ztq / zq)) / (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

b(1) = (htq + wq * uq) / rro2(i) + (hht2(i) + w2(i) * uu2(i)) / rro2(i)
b(1) = b(1) * dt / (2 * cp * ar) + a(1) * pq + tq
b(2) = (1 + tt2(i) * zz2(i) / zz2(i)) * (hht2(i) + w2(i) * uu2(i))
b(2) = b(2) * aas2(i) / rro2(i) * tt2(i)
b(2) = b(2) / (1 + tr * ztr / zr) * (htr + wr * ur) / (ror * tr) / cp
b(2) = b(2) * (wt / ror + w2(i) / rro2(i)) * dt / (2 * ar)
b(2) = b(2) / (1 + tt2(i) * zz2(i) / zz2(i)) * (hht2(i) + w2(i) * uu2(i))
b(3) = b(3) * aas2(i) / (rro2(i) * tt2(i))
b(3) = (b(3) * asr * (1 + ts * zts / zs) * (hts + ws * us) / (ros * ts)) / cp
b(3) = b(3) - (ws / ros + w2(i) / rro2(i)) * dt / (2 * ar)
b(3) = b(3) - g * dt * dsin(th) + a(4) * pr + ur

if(det . ne. 0.0) goto 12
write (6, 31) i
31 format ('pipe 2 (2nd order) i=' ', i3)
write (6, 16) i
16 format ('sub4 - no iteration for i=' ', i4, ' in pipe 2')
This subroutine calculates p, t and u at internal boundary points linking different grid sizes.

```
c subroutine sub5(pl, tl, ul, dx, ppl, ttl, uul, tc, pc, 1', l!-! ar, T, d, cp, pi, st, tw, g, dt, i, zi, zpl, ztl, rol, wi, as1, & htl, th, px, ux, tx, wx, zx, ztx, rox, asx, htx, py, uy, ty, & wy, zy, zty, royalty, asy, hty, k)
c c This subroutine calculates p, t and u at internal boundary points linking different grid sizes.

c implicit double precision (a-h, o-z)
dimension pl(300), tl(300), ul(300), zl(300), & ztl(300), wl(300), as1(300), htl(300), pp1(300), & tti(300), uu1(300), zzi(300), ztp1(300), ztt1(300), & rr0l(300), uww1(300), asas1(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), & wy(6), zty(6), royalty(6), asy(6), hty(6), a(7), b(3), & rol(300)
integer ii(3), mm(3), count

First order approximation
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

posr=dt*4/(1/(ux(k)+asx(k))+1/(u1(i-1)+as1(i-1)))
poss=dt*2/(1/(as(k)-uy(k))+1/(as1(i+1)-u1(i+1)))

pq=posq/(2*dx)*pl(i-1)+(1-posq/(2*dx))*px(k)
tq=posq/(2*dx)*tl(i-1)+(1-posq/(2*dx))*tx(k)
uq=posq/(2*dx)*ul(i-1)+(1-posq/(2*dx))*ux(k)
zq=posq/(2*dx)*zl(i-1)+(1-posq/(2*dx))*zx(k)
ztr=posq/(2*dx)*zt1(i-1)+(1-posq/(2*dx))*ztx(k)
roq=posq/(2*dx)*ro1(i-1)+(1-posq/(2*dx))*roxt(k)
asq=posq/(2*dx)*asl(i-1)+(1-posq/(2*dx))*asx(k)
htq=posq/(2*dx)*htl(i-1)+(1-posq/(2*dx))*htx(k)
wq=posq/(2*dx)*wl(i-1)+(1-posq/(2*dx))*wx(k)
pr=posr/(2*dx)*pl(i-1)+(1-posr/(2*dx))*px(k)
tr=posr/(2*dx)*tl(i-1)+(1-posr/(2*dx))*tx(k)
ur=posr/(2*dx)*ul(i-1)+(1-posr/(2*dx))*ux(k)
zs=poss/dx*p1(i+1)+(1-poss/dx)*py(k)

x1=asr*2*dt*(1+tr*ztr/zr)/(ror*cp*tr*ar)
x2=ass*dt*(1+ts*zt1/zs)/(ros*cp*ts*ar)
a(1)=-(1+twq*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)
```
\[
a(S) = \frac{wr \times ur \times xl}{2 \times tr} - \frac{wr \times dt}{ar \times ror \times tr}
\]

\[
a(7) = -1 \times \frac{rass \times ws \times us \times x2}{2 \times ps} + \frac{ws \times dt}{2 \times ar \times rass \times ps}
\]

\[
a(S) = -\frac{ws \times us \times x2}{2 \times ts} - \frac{ws \times dt}{2 \times ar \times rass \times ts}
\]

\[
\text{if} (ur \text{ eq. 0.0}) \text{ goto 2}
\]

\[
\text{if} (us \text{ eq. 0.0}) \text{ goto 2}
\]

\[
a(6) = 1 - \frac{wr \times x1 \times wr \times 2 \times dt}{(ar \times ror \times ur)}
\]

\[
a(9) = 1 + \frac{ws \times x2 \times ws \times dt}{(ar \times rass \times us)}
\]

\[
\text{goto 1}
\]

\[
a(6) = 1
\]

\[
a(9) = 1
\]

\[
b(1) = \frac{htr \times 2 \times dt}{(roq \times cp \times ar)} + tq - p.q \times (1 + tq \times ztq / zq) / (roq \times cp)
\]

\[
b(2) = \frac{htr \times xl + ur + pr}{(ror \times asr)} - g \times 2 \times dt \times dsin(th)
\]

\[
b(3) = -\frac{hts \times x2 + ws \times ps}{(roq \times rass)} - g \times dt \times dsin(th)
\]

\[
call dminv(a, 3, det, 11, mm)
\]

\[
\text{if} (det \text{ ne. 0.0}) \text{ goto 9}
\]

\[
write(6, 30) i
\]

\[
\text{format(’pipe1 =’, i3)}
\]

\[
\text{stop “no inverse”}
\]

\[
ppl(i) = a(1) \times b(1) + a(2) \times b(2) + a(3) \times b(3)
\]

\[
ttt(i) = a(4) \times b(1) + a(5) \times b(2) + a(6) \times b(3)
\]

\[
uul(i) = a(7) \times b(1) + a(8) \times b(2) + a(9) \times b(3)
\]

\[
\text{count} = 0 -
\]

\[
\text{psave} = ppl(i)
\]

\[
\text{tsave} = ttt(i)
\]

\[
\text{usave} = uul(i)
\]

\[
pdif = ppl(i) \times 1000
\]

\[
tdf = ttt(i) \times 1000
\]

\[
zzl(i) = 9 \times tc / (128 \times tti(i)) - 26 \times tc / ((64 \times tti(i)) \times 3)
\]

\[
zzl(i) = zt1(i) / ppl(i) + 1
\]

\[
zzl1(i) = (zzi(i) + 1) / ppl(i)
\]

\[
zzt1(i) = zzi(i) \times 3 / (64 \times tti(i)) + 9 \times tc / (128 \times tti(i) \times tti(i))
\]

\[
zzt1(i) = zzi(i) / ppl(i)
\]

\[
ror1(i) = ppl(i) / (rtt(i) \times zzi(i))
\]

\[
uul(i) = dabs(ar \times ror1(i) \times fxuul(i) \times uu1(i)) / (2 \times d)
\]

\[
aas1(i) = (1 + zzt1(i) \times tti(i)) / zzi(i) \times 2 \times ppl(i)
\]

\[
aas1(i) = aas1(i) / ror1(i) \times tti(i) / cp + zzi(i) / ppl(i) / zzi(i)
\]

\[
aas1(i) = (1 - aas1(i)) \times ror1(i) / ppl(i)
\]

\[
aas1(i) = 1 / dsqrt(dabs(aas1(i)))
\]

\[
htl(i) = p \times cp \times st \times d \times ror1(i) \times uu1(i) \times (tw - tti(i))
\]

\[
\text{count} = \text{count} + 1
\]

\[
\text{Second order procedure}
\]

\[
\text{if} (uq \text{ eq. 0.0}) \text{ goto 21}
\]

\[
\text{if} (uu1(i) \text{ eq. 0.0}) \text{ goto 21}
\]

\[
\text{posq} = 4 \times dt / (1 / uq + uu1(i))
\]

\[
goto 23
\]

\[
\text{posq} = dt / (uq + uu1(i))
\]

\[
posr = 4 \times dt / (1 / ur + asr) + 1 / (uu1(i) + aas1(i))
\]

\[
poss = 2 \times dt / (1 / (ass - us) + 1 / (aas1(i) - uu1(i)))
\]

\[
pq = ppl(i - 1) + (2 \times dx - posq) \times (px(k) - ppl(i - 2)) / (4 \times dx)
\]

\[
uq = uu1(i - 1) + (2 \times dx - posq) \times (vx(k) - uu1(i - 2)) / (4 \times dx)
\]

\[
tq = tti(i - 1) + (2 \times dx - posq) \times (tx(k) - tti(i - 2)) / (4 \times dx)
\]

\[
qq = zzi(i - 1) + (2 \times dx - posq) \times (zx(k) - zzi(i - 2)) / (4 \times dx)
\]

\[
wq = wul(i - 1) + (2 \times dx - posq) \times (wx(k) - wul(i - 2)) / (4 \times dx)
\]

\[
zq = ztl(i - 1) + (2 \times dx - posq) \times (zt(k) - ztl(i - 2)) / (4 \times dx)
\]

\[
roq = rro(i - 1) + (2 \times dx - posq) \times (rox(k) - rro(i - 2)) / (4 \times dx)
\]

\[
ass = aas1(i - 1) + (2 \times dx - posq) \times (asx(k) - aas1(i - 2)) / (4 \times dx)
\]

\[
htq = htl(i - 1) + (2 \times dx - posq) \times (htx(k) - htl(i - 2)) / (4 \times dx)
\]

\[
pr = ppl(i - 1) + (2 \times dx - posq) \times (px(k) - ppl(i - 2)) / (4 \times dx)
\]
\[
\begin{align*}
tr &= zt(1-i-1) + (2*dx-psr)*t(1-i-2) / (4*dx) \\
zr &= zl(1-i-1) + (2*dx-psr)*z(1-i-2) / (4*dx) \\
wtr &= ul(1-i-1) + (2*dx-posr)*u(1-i-2) / (4*dx) \\
fr &= ro(1-i-1) + (2*dx-posr)*r(1-i-2) / (4*dx) \\
lsr &= as(1-i-1) + (2*dx-posr)*a(1-i-2) / (4*dx) \\
hrs &= hti(1-i-1) + (2*dx-posr)*h(1-i-2) / (4*dx) \\
ps &= pl(i+1) - (dx-posq)*(p(i+2)-py(k)) / (2*dx) \\
us &= ul(i+1) - (dx-psq)*(ul(i+2)-uy(k)) / (2*dx) \\
zs &= zl(i+1) - (dx-psq)*(zl(i+2)-zy(k)) / (2*dx) \\
wz &= wu(i+1) - (dx-psq)*(wu(i+2)-wy(k)) / (2*dx) \\
zs &= zl(i+1) - (dx-psq)*(zl(i+2)-zy(k)) / (2*dx) \\
pr &= pr(i-1) + (2*dx-posq)*p(i-2) / (8*dx*dx) \\
ur &= ur(i-1) + (2*dx-posq)*u(i-2) / (8*dx*dx) \\
tr &= tr(i-1) + (2*dx-posq)*t(i-2) / (8*dx*dx) \\
zr &= zr(i-1) + (2*dx-posq)*z(i-2) / (8*dx*dx) \\
wz &= wz(i-1) + (2*dx-psq)*w(i-2) / (8*dx*dx) \\
zs &= zl(i+1) + (2*dx-psq)*z(i+2) / (8*dx*dx) \\
ps &= pl(i+1) - (dx-posq)*(p(i+2)-py(k)) / (2*dx*dx) \\
qs &= qs(i+1) + (2*dx-psq)*q(i+2) / (2*dx*dx) \\
tr &= tr(i+1) - (2*dx-psq)*t(i+2) / (8*dx*dx) \\
zr &= zr(i+1) - (2*dx-psq)*z(i+2) / (8*dx*dx) \\
wz &= wz(i+1) + (2*dx-psq)*w(i+2) / (8*dx*dx) \\
zs &= zl(i-1) + (2*dx-psq)*z(i-2) / (8*dx*dx) \\
\end{align*}
\]

\[
\begin{align*}
a(1) &= -(1+ttl(i)*zztl(i)/ztl(i))/rrol(i) \\
a(1) &= (a(1)-(1+ttq*ztq/zq))/roq/(2*cp) \\
a(2) &= 1.0 \\
a(3) &= 0.0 \\
a(4) &= (1/(ror*asr)+1/(rrol(i)*aasl(i)))/2 \\
a(5) &= 0.0 \\
a(6) &= 1.0 \\
a(7) &= (-1/(ros*ass)-1/(rrol(i)*aasl(i)))/2 \\
a(8) &= 0.0 \\
a(9) &= 1.0 \\
\end{align*}
\]

\[
\begin{align*}
b(1) &= (htq+wq*uq)/roq+(hhtl(i)+wwl(i)*uul(i))/rrol(i) \\
b(1) &= b(1)*dt/(cp*ar)+a(1)*pq+tq \\
b(2) &= (1+ttl(i)*wwzi(i)/zzi(i))*(hhtl(i)+wwl(i)*uul(i)) \\
b(2) &= b(2)*aasl(i)/(rrol(i)*ttl(i)) \\
b(2) &= (b(2)+asr(1+ttt*ztt/ztz)*(httq+wwq*ur)/ror*tr)/cp \\
b(2) &= (b(2)-(wq/roq)*wwl(i))/rrol(i)))*dt/ar \\
b(2) &= (b(2)+qg2*dt*sin(th)+a(4)*pr+ur \\
b(3) &= (1+ttl(i)*wwzi(i)/zzi(i))*(hhtl(i)+wwl(i)*uul(i)) \\
\end{align*}
\]
\[ b(3) = (b(3) - \left( \frac{\text{hts} + \text{ws} + \text{us}}{(\text{ros} \times \text{ts})} \right) \times (\text{hts} + \text{ws} + \text{us}) / (\text{ros} \times \text{ts}) / \text{cp} \] \\
\[ b(3) = (b(3) - \left( \frac{\text{ws} \times \text{ros} + \text{wwl}(1)}{\text{rrol}(1)} \right) \times \text{dt} / (2 \times \text{ar}) \] \\
\[ c \]

\begin{verbatim}
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)
uul=a(7)*b(1)+a(8)*b(2)+a(9)*b(3)
difp=dabs(pp1(i)-pit)/pit
dift=dabs(ttl(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
ttl(i)=tit
uu1(i)=uul
pdif=difp
tdif=dift
goto 10
15 write(6,16) i
16 format('sub5 - no iteration for i=', i4, 'in pipe 1 (sub5)')
14 pp1(i)=pit
ttl(i)=tit
uu1(i)=uul
return
end
\end{verbatim}
This subroutine calculates $p$, $t$ and $u$ at internal boundary points linking different grid sizes in pipe 2.

First order approximation

```plaintext
posr = dt * x2 / (1/(u2(i) + as2(i)) + 1/(u2(i) - 1 + as2(i - 1)))
posq = dt * x2 / ((1 + as2(i) - u2(i - 1)) + 1/(as2(i) - uz(k)))
```

```plaintext
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)
```

```plaintext
ztr = posr / dx * zt2(i - 1) + (1 - posr / dx) * zt2(i)
```

```plaintext
ror = posr / dx * ro2(i - 1) + (1 - posr / dx) * ro2(i)
asr = posr / dx * as2(i - 1) + (1 - posr / dx) * as2(i)
```

```plaintext
htr = posr / dx * ht2(i - 1) + (1 - posr / dx) * ht2(i)
```

```plaintext
wr = posr / dx * w2(i - 1) + (1 - posr / dx) * w2(i)
pw = posq / (2 * dx) * p2(i - 1) + (1 - posq / (2 * dx)) * p2(k)
tq = posq / (2 * dx) * t2(i - 1) + (1 - posq / (2 * dx)) * t2(k)
uq = posq / (2 * dx) * u2(i - 1) + (1 - posq / (2 * dx)) * u2(k)
zq = posq / (2 * dx) * z2(i - 1) + (1 - posq / (2 * dx)) * z2(k)
```

```plaintext
zts = posq / (2 * dx) * zt2(i - 1) + (1 - posq / (2 * dx)) * zt2(k)
```

```plaintext
ros = posq / (2 * dx) * ro2(i - 1) + (1 - posq / (2 * dx)) * ro2(k)
ass = posq / (2 * dx) * as2(i - 1) + (1 - posq / (2 * dx)) * as2(k)
```

```plaintext
htrs = posq / (2 * dx) * ht2(i - 1) + (1 - posq / (2 * dx)) * ht2(k)
```

```plaintext
ws = posq / (2 * dx) * w2(i - 1) + (1 - posq / (2 * dx)) * w2(k)
```

```plaintext
x1 = asr * dt * x1 + tr * ztr / zr / (ror * cp * tr * ar)
x2 = ass * dt * x2 + 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 * xs * x2 / (2 * ps) + ws * dt / (ar * ros * ps)
a(8) = ws * xs * x2 / (2 * ts) - ws * dt / (ar * ros * ts)
```

```plaintext
if (pr eq 0.0) goto 30
if (us eq 0.0) goto 30
```

```plaintext
if (x2 = 1) goto 31
```

```plaintext
a(6) = 1
a(9) = 1
```

```plaintext
b(2) = htr * x1 + ur * pr / (ror * asr) - g * dt * dsin (th)
b(3) = - hts * x2 + us - ps / (ros * ass) - g * dt * dsin (th)
```

```plaintext
if (u2(i) .le. 0.0) goto 1
if (u2(i) .eq. 0.0) goto 2
```

```plaintext
posq = dt * x2 / (1 / u2(i) + 1 / u2(i - 1))
```

```plaintext
goto 5
```

```plaintext
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)
```

```plaintext
265
```
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))

7 posq=dabs(dt*(u2(i)+u2(i+l))

8 pq=posq/(2*dx*p2(i+1)+(1-posq/(2*dx))*p2(k)
tq=posq/(2*dx*t2(i+1)+(1-posq/(2*dx))*t2(k)
uq=posq/(2*dx*u2(i+1)+(1-posq/(2*dx))*u2(k)
ztq=posq/(2*dx*zt2(i+1)+(1-posq/(2*dx))*zt2(k)
roq=posq/(2*dx*ro2(i+1)+(1-posq/(2*dx))*ro2(k)
asq=posq/(2*dx*as2(i+1)+(1-posq/(2*dx))*as2(k)
htq=posq/(2*dx*ht2(i+1)+(1-posq/(2*dx))*ht2(k)
wq=posq/(2*dx*w2(i+1)+(1-posq/(2*dx))*w2(k)

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

call dminv(a,3,del,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
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)-1/9*tc/(128*tt2(i)*tt2(i))
zz2(i)=zz2(i)+pp2(i)
rro2(i)=pp2(i)/(r*tt2(i)**2)
ww2(i)=dabs(ar*rro2(i)*f*uu2(i)*uu2(i))/(2*d)
aas2(i)=(1+zz2(i)*tt2(i)/zz2(i)**2)*pp2(i)
aas2(i)=aas2(i)/(rro2(i)**2)
ht2(i)=p*cp*tt2(i)*(2-tt2(i))
count=count+1

Second order procedure
posr=2*dt/(1/(ur+asr)+1/(uu2(i)+aas2(i)))
\[ pr = p_2(i-1) + (dx-posr)*p_2(i) - p_2(i-2) \]  
\[ ur = u_2(i-1) + (dx-posr)*u_2(i) - u_2(i-2) \]  
\[ tr = t_2(i-1) + (dx-posr)*t_2(i) - t_2(i-2) \]  
\[ zr = z_2(i-1) + (dx-posr)*z_2(i) - z_2(i-2) \]  
\[ wr = w_2(i-1) + (dx-posr)*w_2(i) - w_2(i-2) \]  
\[ ztr = zt_2(i-1) + (dx-posr)*zt_2(i) - zt_2(i-2) \]  
\[ ro = r_2(i-1) + (dx-posr)*r_2(i) - r_2(i-2) \]  
\[ asr = as_2(i-1) + (dx-posr)*as_2(i) - as_2(i-2) \]  
\[ htr = ht_2(i-1) + (dx-posr)*ht_2(i) - ht_2(i-2) \]  
\[ ps = p_2(i+1) - (2*dx-posr)*p_2(i+2) - p_2(i) \]  
\[ us = u_2(i+1) - (2*dx-posr)*u_2(i+2) - u_2(i) \]  
\[ ts = t_2(i+1) - (2*dx-posr)*t_2(i+2) - t_2(i) \]  
\[ zs = z_2(i+1) - (2*dx-posr)*z_2(i+2) - z_2(i) \]  
\[ ws = w_2(i+1) - (2*dx-posr)*w_2(i+2) - w_2(i) \]  
\[ zts = zt_2(i+1) - (2*dx-posr)*zt_2(i+2) - zt_2(i) \]  
\[ ros = r_2(i+1) - (2*dx-posr)*r_2(i+2) - r_2(i) \]  
\[ ass = as_2(i+1) - (2*dx-posr)*as_2(i+2) - as_2(i) \]  
\[ hts = ht_2(i+1) - (2*dx-posr)*ht_2(i+2) - ht_2(i) \]  
\[ pr = pr + (dx-posr)^2*(p_2(i) + p_2(i-2) - 2*p_2(i-1)) \]  
\[ ur = ur + (dx-posr)^2*(u_2(i) + u_2(i-2) - 2*u_2(i-1)) \]  
\[ tr = tr + (dx-posr)^2*(t_2(i) + t_2(i-2) - 2*t_2(i-1)) \]  
\[ zr = zr + (dx-posr)^2*(z_2(i) + z_2(i-2) - 2*z_2(i-1)) \]  
\[ wr = wr + (dx-posr)^2*(w_2(i) + w_2(i-2) - 2*w_2(i-1)) \]  
\[ ztr = ztr + (dx-posr)^2*(zt_2(i) + zt_2(i-2) - 2*zt_2(i-1)) \]  
\[ ro = ro + (dx-posr)^2*(r_2(i) + r_2(i-2) - 2*r_2(i-1)) \]  
\[ asr = asr + (dx-posr)^2*(as_2(i) + as_2(i-2) - 2*as_2(i-1)) \]  
\[ htr = htr + (dx-posr)^2*(ht_2(i) + ht_2(i-2) - 2*ht_2(i-1)) \]  
\[ ps = ps + (2*dx-posr)*p_2(i+2) - p_2(i) \]  
\[ us = us + (2*dx-posr)*u_2(i+2) - u_2(i) \]  
\[ ts = ts + (2*dx-posr)*t_2(i+2) - t_2(i) \]  
\[ zs = zs + (2*dx-posr)*z_2(i+2) - z_2(i) \]  
\[ ws = ws + (2*dx-posr)*w_2(i+2) - w_2(i) \]  
\[ zts = zts + (2*dx-posr)*zt_2(i+2) - zt_2(i) \]  
\[ ros = ros + (2*dx-posr)*r_2(i+2) - r_2(i) \]  
\[ ass = ass + (2*dx-posr)*as_2(i+2) - as_2(i) \]  
\[ hts = hts + (2*dx-posr)*ht_2(i+2) - h_2(i) \]  
\[ if((uu2(i)+uq).lt.0.0) goto 4 \]  
\[ if(uq .eq. 0.0) goto 11 \]  
\[ posq = dabs(dt*2/(1/uu2(i)+1/uq)) \]  
\[ goto 20 \]  
\[ 20 pq = p_2(i-1) + (dx-posr)*p_2(i) - p_2(i-2) \]  
\[ uq = u_2(i-1) + (dx-posr)*u_2(i) - u_2(i-2) \]  
\[ tq = t_2(i-1) + (dx-posr)*t_2(i) - t_2(i-2) \]  
\[ zq = z_2(i-1) + (dx-posr)*z_2(i) - z_2(i-2) \]  
\[ wq = w_2(i-1) + (dx-posr)*w_2(i) - w_2(i-2) \]  
\[ zqt = zt_2(i-1) + (dx-posr)*zt_2(i) - zt_2(i-2) \]  
\[ roq = r_2(i-1) + (dx-posr)*r_2(i) - r_2(i-2) \]  
\[ asq = as_2(i-1) + (dx-posr)*as_2(i) - as_2(i-2) \]  
\[ hqt = ht_2(i-1) + (dx-posr)*ht_2(i) - ht_2(i-2) \]  
\[ pq = pq + (dx-posr)^2*(p_2(i) + p_2(i-2) - 2*p_2(i-1)) \]  
\[ uq = uq + (dx-posr)^2*(u_2(i) + u_2(i-2) - 2*u_2(i-1)) \]  
\[ tq = tq + (dx-posr)^2*(t_2(i) + t_2(i-2) - 2*t_2(i-1)) \]  
\[ zq = zq + (dx-posr)^2*(z_2(i) + z_2(i-2) - 2*z_2(i-1)) \]  
\[ wq = wq + (dx-posr)^2*(w_2(i) + w_2(i-2) - 2*w_2(i-1)) \]  
\[ zqt = zqt + (dx-posr)^2*(zt_2(i) + zt_2(i-2) - 2*zt_2(i-1)) \]  
\[ roq = roq + (dx-posr)^2*(ro_2(i) + ro_2(i-2) - 2*ro_2(i-1)) \]  
\[ asq = asq + (dx-posr)^2*(as_2(i) + as_2(i-2) - 2*as_2(i-1)) \]  
\[ hqt = hqt + (dx-posq)^2*(ht_2(i) + ht_2(i-2) - 2*ht_2(i-1)) \]  
\[ goto 6 \]
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)+pzek)-2*p2(i+1)/(8*dx*dx)
uq=uq+(2*dx-posq)**2*(u2(i+2)+uzek)-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*zz(i+1))/(8*dx*dx)
wq=wq+(2*dx-posq)**2*(w2(i+2)+wz(k)-2*wz(i+1))/(8*dx*dx)
ztq=ztq+(2*dx-posq)**2*(zt2(i+2)+ztz(k)-2*ztz(i+1))/(8*dx*dx)
roq=roq+(2*dx-posq)**2*(ro2(i+2)+roz(k)-2*roz(i+1))/(8*dx*dx)
asq=asq+(2*dx-posq)**2*(as2(i+2)+asz(k)-2*asz(i+1))/(8*dx*dx)
htq=htq+(2*dx-posq)**2*(ht2(i+2)+htz(k)-2*htz(i+1))/(8*dx*dx)

6 a(1)=-(1+tt2(i)*zzt2(i)/zz2(i)*rrro2(i))
a(1)=(a(1)*)((1+tq*ztq/zq)/roq)/(2*cp)
a(2)=1.0
a(3)=0.0
a(4)=((1/rrro*asr)+1/(rrro2(i)*as2(i)))/2
a(5)=0.0
a(6)=1.0
a(7)=(-1/(ros*ass)-1/(rrro2(i)*as2(i)))/2
a(8)=0.0
a(9)=1.0

if(uu2(i).eq.0.0) goto 17
b(1)=(htq+wq*uq)/roq+(ht2(i)+ww2(i)*uu2(i))/rrro2(i)
b(1)=b(1)*dt/(2*cp*ar)+a(1)*pq+tq
goto 18
17 b(1)=(htq+wq*uq)/roq+(ht2(i)+ww2(i)*uu2(i))/rrro2(i)
b(1)=b(1)*dt/(2*cp*ar)+a(1)*pq+tq
18 b(2)=(1+t2(i)*zzt2(i)/zz2(i)*)((h2(i)+ww2(i)*uu2(i))
b(2)=b(2)*as2(i)/rrro2(i)*tt2(i)
b(2)=(b(2)+asr*(1+tr*zt2/zz2)*htr+wr*ur)/(rrro*tr)/cp
b(2)=(b(2)-wr/rrro+ww2(i)/rrro2(i)))*dt/(2*ar)
b(2)=(b(2)-g*dt*dsin(th)+a(4)*pr+ur
b(3)=(1+t2(i)*zzt2(i)/zz2(i)*)((h2(i)+ww2(i)*uu2(i))
b(3)=b(3)*as2(i)/rrro2(i)*tt2(i)
b(3)=(-b(3)-ass*(1+ts*ztz/zz2)*(hs+ws*us)/(ros*ts))/cp
b(3)=(b(3)-(ws/rrro+ww2(i)/rrro2(i)))*dt/ar
b(3)=b(3)-g*2*dt*dsin(th)+a(7)*ps+us
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)
uti=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

write(6,16)i
format(’sub6 - no iteration for i=’,i4,’ in pipe 2’)
14 pp2(i)=pit
tt2(i)=tit
uu2(i)=uit
return
end
subroutine break1 (p1, t1, u1, p2, t2, u2, ppl, ttl, uul, z1, ztl, rol, asl, htl, wl, dx, tc, pc, r, ar, f, d, cp, pi, st, tw, g, dt, i, th, z2, zt2, ro2, as2, ht2, w2)

this subroutine calculates steady flow conditions at the break point prior to failure (pipe 1).

implicit double precision (a-h, o-z)
dimension p1(300), t1(300), u1(300), z1(300),
& t1(300), w1(300), as1(300), ht1(300),
& ppl(300), ttl(300), uul(300), p2(300), t2(300),
& u2(300), z2(300), zt2(300), ro2(300), w2(300),
& as2(300), ht2(300), z2(300), zt2(300),
& ztl(300), rol(300), uul(300), as1(300),
& htl1(300), a(9), b(3), rol(300)
integer 11(3), mm(3), count

first order approximation

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)))
pos2=dt*2/(1/(as1(1)-u2(1))+1/(as1(2)-u2(2)))
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)
qz=posq/dx*z1(i-1)+(1-posq/dx)*z1(i)
zq=posq/dx*ztl(i-1)+(1-posq/dx)*ztl(i)
roq=posq/dx*rol(i-1)+(1-posq/dx)*rol(i)
asp=posq/dx*as1(i-1)+(1-posq/dx)*as1(i)
htq=posq/dx*htl(i-1)+(1-posq/dx)*htl(i)
wq=posq/dx*wl(i-1)+(1-posq/dx)*wl(i)
pr=posr/dx*p1(i-1)+(1-posr/dx)*p1(i)
tr=posr/dx*t1(i-1)+(1-posr/dx)*t1(i)
uir=posr/dx*u1(i-1)+(1-posr/dx)*u1(i)
izr=posr/dx*z1(i-1)+(1-posr/dx)*z1(i)
zt2=posr/dx*ztl(i-1)+(1-posr/dx)*ztl(i)
ro2=posr/dx*rol(i-1)+(1-posr/dx)*rol(i)
asr=posr/dx*as1(i-1)+(1-posr/dx)*as1(i)
ht2=posr/dx*htl(i-1)+(1-posr/dx)*htl(i)
w2=posr/dx*wl(i-1)+(1-posr/dx)*wl(i)
p2=posr/dx*p2(1)+(1-posr/dx)*p2(1)
t2=posr/dx*t2(1)+(1-posr/dx)*t2(1)
u2=posr/dx*u2(1)+(1-posr/dx)*u2(1)
z2=posr/dx*z2(1)+(1-posr/dx)*z2(1)
zt2=posr/dx*ztl(2)+(1-posr/dx)*ztl(2)
ro2=posr/dx*rol(2)+(1-posr/dx)*rol(2)
as2=posr/dx*as2(2)+(1-posr/dx)*as2(2)
ht2=posr/dx*htl(2)+(1-posr/dx)*htl(2)
w2=posr/dx*wl(2)+(1-posr/dx)*wl(2)

a(1)=-(1+tq*ztq/zq)/croq*cp-tq*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)
\[
a(8) = \frac{-ws*us*x2/(2*ts) - ws*dt/(2*ar*ros*ts)}{if(\text{ur.eq.0.0}) \text{goto 20}}
\]
\[
a(6) = 1 - \frac{wr*x1 + wr*dt/(ar*ror*ur)}{a(9) = 1 + ws*x2 + ws*dt/(ar*ros*us)}
\]
goto 20
\[
a(6) = 1.0
\]
a(9) = 1.0
\[
21 b(1) = \frac{htq*dt/(roq*cp*ar) + tq - pq*(1 + t*qztq/zq)}{(roq*cp)}
\]
\[
b(2) = \frac{htr*x1 + ur + pr/(ror*asr) - g*dt*dsin(th)}{b(3) = 1 - lsr*x2 + us - ps/(ros*ass) - g*dt*dsin(th)}
\]
call dminv(a, 3, det, 1, mm)
\[
if(det.ne.0.) \text{goto 9}
\]
stop "no inverse"
\[
9 ppl(i) = a(1)*b(1) + a(2)*b(2) + a(3)*b(3)
\]
\[
tti(i) = a(4)*b(1) + a(5)*b(2) + a(6)*b(3)
\]
\[
usu(i) = a(7)*b(1) + a(8)*b(2) + a(9)*b(3)
\]
\[
\text{count}=0
\]
\[
psave=ppl(i)
\]
\[
usave=usu(i)
\]
\[
tsave=ttti(i)
\]
\[
pdif=ppl(i)*1000
\]
\[
udif=usu(i)*1000
\]
\[
tdif=ttti(i)*1000
\]
\[
10 zzi(i) = l*r*/tc/(128*ttti(i)) - 27*tc**3/(64*ttti(i)**3)
\]
\[
zzi(i) = zzi(i)*pppl(i)/pc+1
\]
\[
zzi(i) = zzi(i)/(-1)*pppl(i)
\]
\[
zdtt(i) = 81*tc**3/(64*ttti(i)**4) - 9*tc/(128*ttti(i)*ttti(i))
\]
\[
zdtt(i) = zdtt(i)*pppl(i)/pc
\]
\[
rrrol(i) = pppl(i)/(rrttti(i)*zzzi(i))
\]
\[
uuui(i) = dabs(ar*rrrol(i)*f*uul(i)*usu(i)) / (2*d)
\]
\[
aasl(i) = ((1 + zzzt(i)*ttti(i)/zzzi(i))**2)*pppl(i)
\]
\[
aasl(i) = aasl(i)/(rrrol(i)*ttti(i)*pc + zzzpl(i)*pppl(i)/zzzi(i))
\]
\[
aasl(i) = (1 - aasl(i)) * rrrol(i)/pppl(i)
\]
\[
aasl(i) = 1 / dabs(t*dqsqrt(t*dabs(aasl(i))))
\]
\[
hht(i) = pi*cp*st*d*rrrol(i)*usu(i)*(tw - ttti(i))
\]
\[
count = count + 1
\]
}

second order procedure
\[
\text{if(uq.eq.0.0) goto 3}
\]
\[
\text{if(uu1(i).eq.0.0) goto 3}
\]
\[
goto 4
\]
\[
3 posq = 2*dt/((1/uq+1/uu1(i))
\]
\[
goto 4
\]
\[
4 posr = 2*dt/(1/(ur+asr)+1/(uu1(i)+aasl(i)))
\]
\[
poss = 2*dt/(1/(ass-us)+1/(aasl(i)-uu1(i)))
\]
\[
pq = pppl(i) - posq*(p2(2) - p1(i-1))/(2*dx)
\]
\[
tq = ttti(i) - posq*(t2(2) - t1(i-1))/(2*dx)
\]
\[
uq = uu1(i) - posq*(u2(2) - u1(i-1))/(2*dx)
\]
\[
zq = zzzi(i) - posq*(z2(2) - z1(i-1))/(2*dx)
\]
\[
tqt = ttzi(i) - posq*(zzti(2) - zzzi(i-1))/(2*dx)
\]
\[
roq = ro1(i) - posq*(ro2(2) - ro1(i-1))/(2*dx)
\]
\[
qas = aasl(i) - posq*(aasl(2) - aasl(i-1))/(2*dx)
\]
\[
hqt = hht(i) - posq*(hht(2) - hht(i-1))/(2*dx)
\]
\[
qtw = uu1(i) - posq*(uw2(2) - u1(i-1))/(2*dx)
\]
\[
pr = pppl(i) - posr*(p2(2) - p1(i-1))/(2*dx)
\]
\[
tr = ttti(i) - posr*(t2(2) - t1(i-1))/(2*dx)
\]
\[
ur = uu1(i) - posr*(u2(2) - u1(i-1))/(2*dx)
\]
\[
rr = zzi(i) - posr*(z2(2) - z1(i-1))/(2*dx)
\]

271
ror=rol(i)-posr*(ro2(2)-rol(i-1))/(2*dx)
asr=asl(i)-posr*(asl2(2)-asl(i-1))/(2*dx)
ht=htl(i)-posr*(ht2(2)-htl(i-1))/(2*dx)
wr=wl(i)-posr*(w2(2)-wl(i-1))/(2*dx)
ps=p1(i)+posr*(p2(2)-p1(i-1))/(2*dx)
ts=t1(i)+posr*(t2(2)-t1(i-1))/(2*dx)
us=ul(i)+posr*(u2(2)-ul(i-1))/(2*dx)
z=zl(i)+posr*(z2(2)-zl(i-1))/(2*dx)
zts=zt1(i)+posr*(zt2(2)-zt1(i-1))/(2*dx)
ros=rol(i)+posr*(ro2(2)-rol(i-1))/(2*dx)
ass=asl(i)+posr*(asl2(2)-asl(i-1))/(2*dx)
hts=htl(i)+posr*(ht2(2)-htl(i-1))/(2*dx)
ws=wl(i)+posr*(w2(2)-wl(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)
q=q+posq*posq*(u2(2)+ul(i-1)-2*u1(i))/(2*dx*dx)
z=q+posq*posq*(z2(2)+z1(i-1)-2*z1(i))/(2*dx*dx)
zq=q+posq*posq*(zt2(2)+zt1(i-1)-2*zt1(i))/(2*dx*dx)
tr=tr+posr*posr*(p2(2)+p1(i-1)-2*p1(i))/(2*dx*dx)
th=h+tq+posq*posq*(ht2(2)+htl(i-1)-2*h1(i))/(2*dx*dx)
wp=wp+posr*posr*(w2(2)+wl(i-1)-2*wl(i))/(2*dx*dx)
p=pr+posr*posr*(p2(2)+p1(i-1)-2*p1(i))/(2*dx*dx)
t=tr+posr*posr*(t2(2)+t1(i-1)-2*t1(i))/(2*dx*dx)
ur=ur+posr*posr*(u2(2)+ul(i-1)-2*u1(i))/(2*dx*dx)
z=zs+posr*posr*(z2(2)+z1(i-1)-2*z1(i))/(2*dx*dx)
zt=zt+posr*posr*(zt2(2)+zt1(i-1)-2*zt1(i))/(2*dx*dx)
ror=ror+posr*posr*(ro2(2)-rol(i-1)-2*rol(i))/(2*dx*dx)
asr=ass+posr*posr*(asl2(2)-asl(i-1)-2*asl(i))/(2*dx*dx)
ht=hts+posr*posr*(ht2(2)-htl(i-1)-2*htl(i))/(2*dx*dx)
ws=ws+posr*posr*(w2(2)+wl(i-1)-2*wl(i))/(2*dx*dx)

a(1)=-(1+t(i)*zz1(i)/zzl(i))/(rorl(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/(rorl(i)*aasl(i)))/2
a(5)=0.0
a(6)=1.0
a(7)=(-1/(ror*ass)-1/(rorl(i)*aasl(i)))/2
a(8)=0.0
a(9)=1.0

b(1)=(htq+wq*uq)/roq+(hhtl(i)+wwl(i)*uul(i))/(rorl(i))
b(1)=(b(1)*dt/(2*cp*ar)+a(1)*pq+tq)
b(2)=(1+t(i)*zz1(i)/zzl(i))*hhtl(i)+wwl(i)*uul(i))
b(2)=(b(2)*aasl(i)/(rorl(i)*t(i))
b(2)=(b(2)+asr*(1+t*ztr/zr)*(htr+wr+u)/(ror*tr))/cp
b(2)=(b(2)-(wr/lor+wul(i))/rorl(i)))*dt/(2*ar)
b(2)=(b(2)-g*dt*sin(2h)+a(4)*pr+ur)
b(3)=(1+t(i)*zz1(i)/zzl(i))*hhtl(i)+wwl(i)*uul(i))
b(3)=(b(3)*aasl(i)/(rorl(i)*t(i))
b(3)=(-b(3)-ass*(1+t*zt1)/zt1(i))/htr+ws*us)/(ros*ts)/cp
b(3)=(b(3)-(ws/ro+wul(i))/rorl(i)))*dt/(2*ar)
call dminv(a,3, det, ll, 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*difp)) goto 15
if(dift.gt.(4*dift)) 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
uit(i)=uit
endifp=dpif
dift=dtif
goto 10

15 write(6,16)i
16 format('divergence - no iteration for i=',l4,' in pipe 1')
pit=psave
tit=tsave
uit=usave
14 pp1(i)=pit
tt1(i)=tit
uit(i)=uit
return
end
subroutine break2(pp1, tt1, uu1, pp2, tt2, uu2, m1, i)

this subroutine calculates steady flow conditions
at the break point prior to failure (pipe 2).

implicit double precision(a-h,o-z)
dimension pp1(300), tt1(300), uu1(300), pp2(300),
& tt2(300), uu2(300)

pp2(1)=pp1(m1)
tt2(1)=tt1(m1)
uu2(1)=uu1(m1)
return
end
subroutine break3(pl, tl, ui, ppl, tti, uul, zi, zti, roi, asi, hti, 
& wi, dx, tc, pc, r, ar, f, d, cp, pi, st, tw, g, dt, i, th, pat, tat)

This subroutine calculates the conditions at the break 
using the equalisation pressure as defined by Bannister 
and Mucklow.

implicit double precision(a-h, o-z)
dimension pl(300), t1(300), ui(300), zi(300), zti(300), 
& wi(300), asi(300), hti(300), ppl(300), tti(300), 
& uul(300), zzi(300), zzt1(300), zzp1(300), roi(300), 
& wwi(300), asw1(300), hht1(300), a(9), roi(300)
integer count

first order approximation

if(ui(i).eq.0.0) goto 1
if(ui(i-1).eq.0.0) goto 1
posq=dt*(ui(i)*1ui(i-1))
goto 2
1 posq=dt*(ui(i)+ui(i-1))/2
2 pr=dt2/(ui(i)+as1(i-1)+ui(i-1)+as1(i-1))
   pq=posq*ul(i)+1-posq*ul(i)
   uq=posq*ul(i)+1-posq*ul(i)
   tq=posq*ul(i)+1-posq*ul(i)
   zq=posq*ul(i)+1-posq*ul(i)
   ztq=posq*ul(i)+1-posq*ul(i)
   roq=posq*roi(i)+1-posq*roi(i)
   asq=posq*as1(i)+1-posq*as1(i)
   htq=posq*ht1(i)+1-posq*ht1(i)
   wq=posq*wl(i)+1-posq*wl(i)
   pr=posq*pl(i)+1-posq*pl(i)
   ur=posq*ul(i)+1-posq*ul(i)
   tr=posq*tl(i)+1-posq*tl(i)
   zr=posq*zl(i)+1-posq*zl(i)
   ztr=posq*ztl(i)+1-posq*ztl(i)
   roq=posq*roi(i)+1-posq*roi(i)
   asq=posq*as1(i)+1-posq*as1(i)
   htq=posq*ht1(i)+1-posq*ht1(i)
   wq=posq*wl(i)+1-posq*wl(i)
   pr=posq*pl(i)+1-posq*pl(i)
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)+hht1(i)/(roq*cp*ar)+wq*pp1(i)/(2*pq*roq*cp*ar)
a(6)=(1+tr*zt1/tr)*asr*ht1/(ror*cp*tr*ar)-g*sin(th)
   a(6)=a(6)-x1*wr*pp1(i)/(2*pr)+ur/dt-(pp1(i)-p1(i))/ror*asr*dt
uul(i)=(a(4)*a(5)-a(2)*a(6))/(a(1)*a(4)-a(2)*a(3))
tti(i)=(a(1)*a(6)-a(3)*a(5))/(a(1)*a(4)-a(2)*a(3))
count=0
  tsave=tti(i)
  usave=uul(i)
  tdif=tti(i)*1000
10 zzi(i)=7+tc/(128*tti(i))-27+tc**3/(64*tti(i)**3)
   zzi(i)=zzi(i)*pp1(i)/pc+1
   zpp1(i)=zzi(i)/pp1(i)
   zzt1(i)=8+tc**3/(64*tti(i)**4)-9+tc/(128*tti(i)**tti(i))
   zzt1(i)=zzt1(i)*pp1(i)/pc
275
uu1(i)=dabs(ar*rrol(i)*fuu1(i)*uu1(i))/(2*|d|

aasl(i)=(1+zztl(i)*ttl(i)/zzl(i))*aasl(i)+2)*pp1(i)
aasl(i)=aasl(i)/rrol(i)*ttl(i)+cp+zzpl(i)*pp1(i)/zzl(i)
aasl(i)=(1-aasl(i))*rrol(i)/pp1(i)
aasl(i)=1/dsqrt(dabs(aasl(i))
hhtl(i)=pi*cp*st*d*rrol(i)*uu1(i)*(tw-ttl(i))
count=count+1

Second order procedure

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*(uu1(i-1)+uu1(i))
4 posr=2*dt/(1/(ur+asr)+1/(uu1(i-1)+aasl(i)))

pq=p1(i-1)+(1-posq/dx)*p1(i-2)/2
tq=t1(i-1)+(1-posq/dx)*t1(i-2)/2
uq=u1(i-1)+(1-posq/dx)*u1(i-2)/2
zq=z1(i-1)+(1-posq/dx)*z1(i-2)/2
ztr=z1(i-1)+(1-posq/dx)*(z1(i)-z1(i-2))/2
roq=r1(i-1)+(1-posq/dx)*(ro1(i)-ro1(i-2))/2
asq=asl(i-1)+(1-posq/dx)*(asl(i)-asl(i-2))/2
htq=h1(i-1)+(1-posq/dx)*(h1(i)-h1(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=z1(i-1)+(1-posr/dx)*(z1(i)-z1(i-2))/2
roq=r1(i-1)+(1-posr/dx)*(ro1(i)-ro1(i-2))/2
asr=asl(i-1)+(1-posr/dx)*(asl(i)-asl(i-2))/2
htr=h1(i-1)+(1-posr/dx)*(h1(i)-h1(i-2))/2
wr=w1(i-1)+(1-posr/dx)*(w1(i)-w1(i-2))/2

pq=pq+p1(i-2)-2*p1(i-1)/2

zq=zq+z1(i-1)/2
ztr=ztr+z1(i-1)/2
roq=roq+ro1(i-2)-2*ro1(i-1)/2
asq=asq+asl(i-2)-2*asl(i-1)/2
htq=htq+h1(i-1)/2
wq=wq+w1(i-1)/2
pr=pr+p1(i-1)/2

ztr=ztr+z1(i)/2
roq=roq+ro1(i)-2*ro1(i-1)/2
asq=asq+asl(i)-2*asl(i-1)/2
htq=htq+h1(i)/2
wq=wq+w1(i)/2

pr=pr+p1(i-2)/2

ztr=ztr+z1(i)/2
roq=roq+ro1(i)-2*ro1(i-1)/2
asq=asq+asl(i)-2*asl(i-1)/2


tit=(1+tq*ztq/zq)/roq+(1+tq*ztq/zq)/z1(i)/zzl(i))/rrol(i)
tit=tit*(pp1(i)/pq)/(2*cp)+tq
tit=(htq*wwq*uq)/roq+(ht1(i)+ww1(i))*uu1(i))/rrol(i)
tit=tit*dt/(2*cp)*asr/ur*tg*dsin(th)
uti=tit(1+tit1(i)*zaasl(i))/h1(i)+ww1(i)+uu1(i)/uu1(i))/rrol(i)
uiti=ui*(rrol(i)+tit1(i))+(1+tr*ztr/zr)*htr+wr*ur)/asr/(rrol(i)
uiti=ui*asr/(2*cp)*asr/ur*asr)*ww1(i)/w1(i)/w1(i)/w1(i)
if(count.gt.200) goto 15
if(diff.lt.0.01) goto 14
ttl(i)=tit
uul(i)=uit
tdif=diff
goto 10
15 write(6,16)i
16 format('divergence - no iteration for i=',i4,' in pipe 1')
tit=tsave
uit=usave
14 ttl(i)=tit
uul(i)=uit
return
end
This subroutine calculates conditions at the point immediately downstream of the break after the break has occurred.

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), ass(300), hht2(300), a(9), tt1(300)
Integer ml, count

First order approximation

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)
u2 = poss/dx*u2(2)+(1-poss/dx)*u2(1)
z2 = poss/dx*z2(2)+(1-poss/dx)*z2(1)
zt2 = poss/dx*zt2(2)+(1-poss/dx)*zt2(1)
ros = poss/dx*ro2(2)+(1-poss/dx)*ro2(1)
as = poss/dx*as2(2)+(1-poss/dx)*as2(1)
ht2 = poss/dx*ht2(2)+(1-poss/dx)*ht2(1)
w2 = poss/dx*w2(2)+(1-poss/dx)*w2(1)

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 = poss/dx*p2(2)+(1-posq/dx)*p2(1)
uq = poss/dx*u2(2)+(1-posq/dx)*u2(1)
tq = poss/dx*t2(2)+(1-posq/dx)*t2(1)
zs = poss/dx*z2(2)+(1-posq/dx)*z2(1)
ztq = poss/dx*zt2(2)+(1-posq/dx)*zt2(1)
ros = poss/dx*ro2(2)+(1-posq/dx)*ro2(1)
as = poss/dx*as2(2)+(1-posq/dx)*as2(1)
htq = poss/dx*ht2(2)+(1-posq/dx)*ht2(1)
wq = poss/dx*w2(2)+(1-posq/dx)*w2(1)
goto 21

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

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-p2(1))/(roq*cp*dt)-htq/(roq*cp*ar)-tq/dt
a(3) = a(3) - wq*uq*p2(1)/(roq*cp*ar*2*pq)

23 x1 = ass*(1+ts*ztq/zs)/(ros*cp*ar*ts)
a(4) = ws*(-x1*ws-1/(ar*ros))/(2*ts)
a(5) = 1/dt+x1*ws+ws/(ar*ros*ws)
a(6) = (ps-p2(1))/(ros*ass*dt)-ws/dt+ws*p2(1)/(2*ar*ros*ps)
if(u2(1).le.0.0) goto 24

24   tt2(1)=tt1(m1)
uu2(1)=(-a(4)*tt2(1)-a(6))/a(5)
goto 25

if (u2(1).le.0.0) goto 24

25   psave=pp2(1)
usave=uu2(1)
count=tt2(1)*1000
tdif=tt2(1)*1000
udif=uu2(1)*1000

if(u2(1).ge.0.0) goto 26

26   poss=2*dt/(1/(ass-us)+1/(aas2(1)-uu2(1)))
ps=ps2(2)-(dx-poss)*(p2(3)-p2(1))/(2*dx)
ts=ts2(2)-(dx-poss)*(t2(3)-t2(1))/(2*dx)
us=us2(2)-(dx-poss)*(u2(3)-u2(1))/(2*dx)
zs=zs2(2)-(dx-poss)*(z2(3)-z2(1))/(2*dx)
ztz=ztz2(2)-(dx-poss)*(ztz2(3)-ztz2(1))/(2*dx)
ro2=ro2(2)-(dx-poss)*(ro2(3)-ro2(1))/(2*dx)
ass=ass2(2)-(dx-poss)*(ass2(3)-ass2(1))/(2*dx)
ht2=ht2(2)-(dx-poss)*(ht2(3)-ht2(1))/(2*dx)
w2=w2(2)-(dx-poss)*(w2(3)-w2(1))/(2*dx)
rs=rs2(2)-(dx-poss)*(rs2(3)-rs2(1))/(2*dx)
pl=p2(2)-(dx-poss)*(p(2(3)+p(2(1)-2*p2(2))/(2*dx*dx)
tq=t2(2)-(dx-poss)*(t(2(3)+t(2(1)-2*t2(2))/(2*dx*dx)
uq=u2(2)-(dx-poss)*(u(2(3)+u(2(1)-2*u2(2))/(2*dx*dx)
qz=z2(2)-(dx-poss)*(z(2(3)+z(2(1)-2*z2(2))/(2*dx*dx)
qtq=qtq2(2)-(dx-poss)*(qtq2(3)-qtq2(1)-2*qtq2(2))/(2*dx*dx)
rq= rq2(2)-(dx-poss)*(rq2(3)-rq2(1)-2*rq2(2))/(2*dx*dx)
as=as2(2)-(dx-poss)*(as2(3)-as2(1))/(2*dx*dx)
qts=qts2(2)-(dx-poss)*(qts2(3)-qts2(1))/(2*dx*dx)
wp=wp2(2)-(dx-poss)*(wp2(3)+wp2(1))/(2*dx*dx)

if(u2(1).ge.0.0) goto 26

Second order procedure

posq=dabs(2*dt/(1/(asq-us)+1/(aas2(1)-uu2(1))))

pq=pq2(2)-(dx-posq)*(p2(3)+p2(1)-2*p2(2))/(2*dx)
tq=tq2(2)-(dx-posq)*(t2(3)+t2(1)-2*t2(2))/(2*dx)
uq=uq2(2)-(dx-posq)*(u2(3)-u2(1))/(2*dx)
qz=qz2(2)-(dx-posq)*(z2(3)-z2(1))/(2*dx)
qtq=qtq2(2)-(dx-posq)*(qtq2(3)-qtq2(1))/(2*dx)
rq=rq2(2)-(dx-posq)*(rq2(3)-rq2(1))/(2*dx)
as=as2(2)-(dx-posq)*(as2(3)-as2(1))/(2*dx)
qts=qts2(2)-(dx-posq)*(qts2(3)-qts2(1))/(2*dx)
wp=wp2(2)-(dx-posq)*(wp2(3)+wp2(1))/(2*dx)
uq = uq + (dx - posq)**2*(u2(3) + u2(1) - 2*u2(2))/(2*dx*dx)  
iz = iz + (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 = htc + (dx - posq)**2*(ht2(3) + htc(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)  
iz = z2(1)  
ztq = zt2(1)  
roq = ro2(1)  
asq = as2(1)  
htq = 0.0  
wq = 0.0

27 tit = (1 + tq*ztq/iz)/roq + (1 + t2(1)*zt2(1)/iz2(1))/ro2(1)  
tit = tit + (htq + wq*uq)*dt/(roq*ar)/(2*cp) + tq

goto 29

28 tit = tit1(1)

29 uit = aas2(1)*(1 + t2(1)*zt2(1)/iz2(1))*(ht2(1) + wu2(1) + uu2(1))  
uit = uit / (ro2(1)*t2(1)) + ass*(1 + ts*zt2/iz) / (ro2*ts)  
uit = uit / (ps*(1 + ts*zt2/iz) / (ps*ro2(1)) + us / (ar*ros)) * dt  
uit = ((pp2(1) + ps)*1 / (ro2*ass) + 1 / (ro2*2(1)*aas2(1)) - uit) / 2  
uit = uit - g*dt*dsin(th) + us  
dift = dabs(tt2(1) - tit)/tit  
if (dift > 4*tdif) goto 15

10 write(5, 16)  

16 format('divergence - no iteration for i=1 in pipe 2')  
tit = tit save  
uit = usave  

13 tt2(1) = tit  
uu2(1) = uit  
return  
end
The subroutine `subup` calculates the pressure, velocity, and temperature at the upstream end of a pipe assuming constant pressure and constant mass flow rate.

**First Order Approximation**

```
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)*ztl(2)+(1-poss/dx)*ztl(1)
ros = (poss/dx)*ro1(2)+(1-poss/dx)*ro1(1)
asses = (poss/dx)*as1(2)+(1-poss/dx)*as1(1)
hts = (poss/dx)*ht1(2)+(1-poss/dx)*ht1(1)
wx = (poss/dx)*w1(2)+(1-poss/dx)*w1(1)
x1 = dt+(1+ts*zts/zs)*ws*ass/(ros*cp*ts*ar)
x2 = x2+g*sin(th)
x3 = (1+ts*zts/zs)*ass*ws*ro1(1)*u1(1)/(cp*ts)
x3 = (x3+ws*ro1(1)*u1(1))/(2*ar*rosc)
ui1(1) = (dsqrt(x2*x2-4*x1*x3)-x2)/(2*x1)
usave = uul(1)
```

**Second Order Procedure**

```
poss = dt*2/(1/(as1(1)-u1(1))+1/(as1(2)-u1(2)))
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 = (poss/dx)*(t1(3)-t1(1))/(2*dx)
t = 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)
```
\[
\begin{align*}
zs &= z_s + (dx - poss)^2 (z_l(3) + z_l(1) - 2z_l(2))/(2 dx^2) \\
ws &= w_l(1) - (dx - poss)^2 (w_l(3) - w_l(1))/(2 dx^2) \\
zts &= z_t(2) - (dx - poss)^2 (z_t(3) - z_t(1))/(2 dx^2) \\
ws &= w_l(2) + (dx - poss) w_l(1) - 2w_l(2))/(2 dx^2) \\
zts &= z_t(1) - (dx - poss)^2 (z_t(3) + z_t(1) - 2z_t(2))/(2 dx^2) \\
ros &= r_o(1) - (dx - poss)^2 (r_o(3) - r_o(1))/(2 dx^2) \\
ros &= r_o(2) - (dx - poss)^2 (r_o(3) + r_o(1) - 2r_o(2))/(2 dx^2) \\
ass &= a_s(1) - (dx - poss)^2 (a_s(3) - a_s(1))/(2 dx^2) \\
hs &= h_t(1) - (dx - poss)^2 (h_t(3) - h_t(1))/(2 dx^2) \\
hrts &= h_t(3) + h_t(1) - 2h_t(2))/(2 dx^2) \\
uit &= u_s + (1/(ros*ass) + 1/(rrol(1) + aasl(1)))*(p(1) - ps)/2 \\
x1 &= ass*(1 + ts*zts/zs)*(hs + ws*us)/(ros*ts) \\
x2 &= aasl(1) + (1 + t1(1)*zzt1(1)/zzl(1))*(hht1(1) + wwl(1)*uu1(1)) \\
x3 &= x3 + x2/(cp*ar) + ws/(ar*ros) + wwl(1)/(ar + rrol(1)) \\
uit &= uit - x3 \\
difu &= uul(1) - uit \\
\text{if (count. gt. 200) goto 1} \\
\text{if (difu lt. 0.01) goto 3} \\
uit &= uisave \\
\text{goto 4} \\
\text{format(22, 2)} \\
\text{write(22, 2)} \\
uit &= uisave \\
uul(i) &= uit \\
\text{rrol(1) = rrol(1) * u1(1)/uul(1) } \\
\text{tt1(1) = p(1)/(rrol(1) * zzl(1) * r) } \\
\text{pp1(1) = p(1) } \\
\text{return } \\
\text{end } 
\end{align*}
\]
This subroutine calculates \( p, u \) and \( t \) at the downstream boundary condition assuming a constant temperature non-return valve situation. (Valve closes if \( u<0 \) m/s)

```
subroutine downl(p2, t2, u2, dx, pp2, tt2, uu2, tc, pc, r, ar, th, f, d, cp, pi, st, tw, g, dt, i, iz, zt2, ro2, w2, as2, & ht2)

this subroutine calculates \( p, u \) and \( t \) at the downstream boundary condition assuming a constant temperature non-return valve situation. (Valve closes if \( u<0 \) m/s)

implicit double precision (a-h,o-z)
dimension p(300), u(300), t(300), z(300), ro(300), & pp(300), uu(300), tt(300), ro(300), w(300), as(300), ht(300), & zz(300), zp(300), zt(300), ww(300), as(300), ht(300)
integer count

First order approximation

\[
\text{posr} = \frac{dt \cdot 2}{1/(u(i)+as(i))+1/(u(i-1)+as(i-1))}
\]

\[
\text{pr} = \frac{\text{posr} \cdot p(i-1)+(1-\text{posr} \cdot dx) \cdot p(i)}{\text{pr} \cdot p(i-1) + (1-\text{posr} \cdot dx) \cdot p(i)}
\]

\[
\text{ur} = \frac{\text{posr} \cdot u(i-1)+(1-\text{posr} \cdot dx) \cdot u(i)}{\text{ur} \cdot u(i-1) + (1-\text{posr} \cdot dx) \cdot u(i)}
\]

\[
\text{tr} = \frac{\text{posr} \cdot t(i-1)+(1-\text{posr} \cdot dx) \cdot t(i)}{\text{tr} \cdot t(i-1) + (1-\text{posr} \cdot dx) \cdot t(i)}
\]

\[
\text{zr} = \frac{\text{posr} \cdot z(i-1)+(1-\text{posr} \cdot dx) \cdot z(i)}{\text{zr} \cdot z(i-1) + (1-\text{posr} \cdot dx) \cdot z(i)}
\]

\[
\text{ztr} = \frac{\text{posr} \cdot zt(i-1)+(1-\text{posr} \cdot dx) \cdot zt(i)}{\text{ztr} \cdot zt(i-1) + (1-\text{posr} \cdot dx) \cdot zt(i)}
\]

\[
\text{ror} = \frac{\text{posr} \cdot ro(i-1)+(1-\text{posr} \cdot dx) \cdot ro(i)}{\text{ror} \cdot ro(i-1) + (1-\text{posr} \cdot dx) \cdot ro(i)}
\]

\[
\text{asr} = \frac{\text{posr} \cdot as(i-1)+(1-\text{posr} \cdot dx) \cdot as(i)}{\text{asr} \cdot as(i-1) + (1-\text{posr} \cdot dx) \cdot as(i)}
\]

\[
\text{htr} = \frac{\text{posr} \cdot ht(i-1)+(1-\text{posr} \cdot dx) \cdot ht(i)}{\text{htr} \cdot ht(i-1) + (1-\text{posr} \cdot dx) \cdot ht(i)}
\]

\[
\text{wr} = \frac{\text{posr} \cdot w(i-1)+(1-\text{posr} \cdot dx) \cdot w(i)}{\text{wr} \cdot w(i-1) + (1-\text{posr} \cdot dx) \cdot w(i)}
\]

```
10 \( z2(i) = 9 \times tc/(128 \times t2(i) - 27 \times tc \times 3/(64 \times t2(i))^3) \)
\( z2(i) = z2(i) \times pp2(i)/pc + 1 \)
\( zp2(i) = (z2(i) - 1)/pp2(i) \)
\( zt2(i) = 81 \times tc^3/(64 \times t2(i)^4) - 9 \times tc/(128 \times t2(i) \times t2(i)) \)
\( zt2(i) = zt2(i) \times pp2(i)/pc + zp2(i)*pp2(i)/z2(i) \)
\( rr2(i) = pp2(i)/(r^2*t2(i)*z2(i)) \)
\( ww2(i) = \text{dabs}(ar^2*ro2(i) + f*uu2(i) + uu2(i))/(2*d) \)
\( aas2(i) = (1 + zt2(i) + t2(i)/z2(i))^2*pp2(i) \)
\( aas2(i) = aas2(i) + rr2(i)*t2(i)*tc + zp2(i)*pp2(i)/z2(i) \)
\( aas2(i) = (1 - aas2(i)) \times rr2(i)/pp2(i) \)
\( aas2(i) = 1/d\sqrt{dabs(aas2(i))} \)
\( hht2(i) = \pi*cp*st*d*rr2(i)*uu2(i)*(tw-tt2(i)) \)
\( count = count + 1 \)

**Second order procedure**

\( posr = dt*2/(1/(asr+ur)+1/(aas2(i)+uu2(i))) \)
\( pr = p2(i-1) + (dx-posr)*(p2(i) - p2(i-2))/(2*dx) \)
\( ur = u2(i-1) + (dx-posr)*(u2(i) - u2(i-2))/(2*dx) \)
\( tr = z2(i-1) + (dx-posr)*(z2(i) - z2(i-2))/(2*dx) \)
\( vr = zt2(i-1) + (dx-posr)*(zt2(i) - zt2(i-2))/(2*dx) \)
\( ro = ro2(i-1) + (dx-posr)*(ro2(i) - ro2(i-2))/(2*dx) \)
\( ur = u2(i-1) + (dx-posr)*(u2(i) - u2(i-2))/(2*dx) \)
\( tr = t2(i-1) + (dx-posr)*(t2(i) - t2(i-2))/(2*dx) \)
\( vr = z2(i-1) + (dx-posr)*(z2(i) - z2(i-2))/(2*dx) \)
\( ho = ht2(i-1) + (dx-posr)*(ht2(i) - ht2(i-2))/(2*dx) \)
\( wr = w2(i-1) + (dx-posr)*(w2(i) - w2(i-2))/(2*dx) \)

\( pr = pr + (dx-posr)^2*(p2(i) + p2(i-2) - 2*p2(i-1))/(2*dx*dx) \)
\( ur = ur + (dx-posr)^2*(u2(i) + u2(i-2) - 2*u2(i-1))/(2*dx*dx) \)
\( tr = tr + (dx-posr)^2*(t2(i) + t2(i-2) - 2*t2(i-1))/(2*dx*dx) \)
\( oo = oo + (dx-posr)^2*(o2(i) + o2(i-2) - 2*o2(i-1))/(2*dx*dx) \)
\( ho = ho + (dx-posr)^2*(ho2(i) + ho2(i-2) - 2*ho2(i-1))/(2*dx*dx) \)
\( wr = wr + (dx-posr)^2*(w2(i) + w2(i-2) - 2*w2(i-1))/(2*dx*dx) \)

\( x1 = (1/(ror*asr) + 1/(rr2(i)*aas2(i)))/2 \)
\( x2 = 1.0 \)
\( x3 = aas2(i) + 1 + t2(i) + zt2(i)/z2(i) + (ht2(i) + uu2(i) + uu2(i))/2 \)
\( x3 = x3/3/(h2(i) + t2(i) + zt2(i)/z2(i)) + asr*1 + tr*zt2(i)/z2(i) + (ht2(i) + uu2(i) + uu2(i))/2 \)
\( x3 = x3/3 * x1 + pr + ur \)

**if** (uu2(i) .le. 0.0) **goto** 5

**posq = dabs(d2*t2(i-1)/uu2(i)) + 1/uq**
\( pq = p2(i-1) + (dx-posq)*(p2(i) + p2(i-2))/(2*dx) \)
\( uq = u2(i-1) + (dx-posq)*(u2(i) + u2(i-2))/(2*dx) \)
\( tq = t2(i-1) + (dx-posq)*(t2(i) + t2(i-2))/(2*dx) \)
\( zq = z2(i-1) + (dx-posq)*(z2(i) + z2(i-2))/(2*dx) \)
\( ztq = zt2(i-1) + (dx-posq)*(zt2(i) + zt2(i-2))/(2*dx) \)
\( roq = ro2(i-1) + (dx-posq)*(ro2(i) + ro2(i-2))/(2*dx) \)
\( asq = as2(i-1) + (dx-posq)*(as2(i) + as2(i-2))/(2*dx) \)
\( hqt = ht2(i-1) + (dx-posq)*(ht2(i) + ht2(i-2))/(2*dx) \)
\( wq = w2(i-1) + (dx-posq)*(w2(i) + w2(i-2))/(2*dx) \)
\( pq = pq + (dx-posq)^2*(p2(i) + p2(i-2) - 2*p2(i-1))/(2*dx*dx) \)
\( uq = uq + (dx-posq)^2*(u2(i) + u2(i-2) - 2*u2(i-1))/(2*dx*dx) \)
\( tq = tq + (dx-posq)^2*(t2(i) + t2(i-2) - 2*t2(i-1))/(2*dx*dx) \)
\( zq = zq + (dx-posq)^2*(z2(i) + z2(i-2) - 2*z2(i-1))/(2*dx*dx) \)
\( ztq = ztq + (dx-posq)^2*(zt2(i) + zt2(i-2) - 2*zt2(i-1))/(2*dx*dx) \)
\( roq = roq + (dx-posq)^2*(ro2(i) + ro2(i-2) - 2*ro2(i-1))/(2*dx*dx) \)
\( asq = asq + (dx-posq)^2*(as2(i) + as2(i-2) - 2*as2(i-1))/(2*dx*dx) \)
\( hqt = hqt + (dx-posq)^2*(ht2(i) + ht2(i-2) - 2*ht2(i-1))/(2*dx*dx) \)
\( wq = wq + (dx-posq)^2*(w2(i) + w2(i-2) - 2*w2(i-1))/(2*dx*dx) \)
x4=-(1+tt2(i)*zz2(i))/rz2(i))/rro2(i)-(1+tq*zq/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-ht2(i)+tq+x4*pq

uit=(x3*x4-x1*x6)/x4
pit=x6/x4
goto 6

5 uif=0.0
pit=x3/x1
6 difp=dabs(pp2(i)-pit)/pit
   if(difp.gt.(4*pdir)) 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

7 write(6,9)
i
9 format('divergence - no iteration for i=','i4,' in pipe 2')
   pit=pdir
   uit=uit
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)

description of parameters
  a - input matrix replaced by inverse on exit
      matrix a is the whole matrix stored in a single
      dimension array columnwise.
  n - order of matrix 'a'
  d - resultant determinant
  l - work vector of length 'n'
  m - work vector of length 'n'

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-l>
   do 20 i=k,l
      ij=iz+i
     10 if(dabs(biga)>dabs(a(ij)))15,20,20
        biga=a(ij)
        l(k)=i
        m(k)=j
     20 continue

   interchange rows
   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

   interchange columns
   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

divide column by minus pivot value (-biga)

   if(biga)48,46,48
   d=0.0
   return
if(i-k)50,55,50
50 ik=nk+i
   a(ik)=a(ik)/(-biga)
55 continue

Reduce matrix

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
      if(j-k)62,65,62
      kj=ij-i+k
      a(ij)=hold*a(kj)+a(ij)
   65 continue

Divide row by pivot

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

Product of pivots

d=d*biga

Replace pivot by reciprocal

a(kk)=1.0/biga
80 continue

Final row and column interchange

k=n
100 k=k-1
   if(k)150,150,105
105 i=1(k)
   if(i-k)120,120,108
108 jq=n*(k-1)
   jr=n*(i-1)
   do,110 j=1,n
      jr=jq+j
      hold=a(jq)
      ji=ji+j
      a(jq)=-a(jq)
   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
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
1=0
20 l=1+1
read (2,*, end=400) time (1)
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)
go to 20
4 if(j.eq.nint2) goto 400
continue
go to 20
400 l=1-1
go to 500
300 l=0
10 l=1+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(l)=p(j)
2 continue
go to 10
200 l=1-1
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 chaang('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 `chain(tiset, 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 `pickle
   call `devend
stop
end
Routine to plot graphs of pressure vs wavespeed

```fortran
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
rewind (1)
rewind (2)
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(1)=p(j)
wplot(1)=wisen(j)
4 continue
goto 20
400 l=l-1
goto 700
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(1)=p(j)
wplot(1)=wisot(j)
6 continue
goto 30
300 l=0
10 l=l+1
```

291
read(1,*,end=200) time(n)
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
wplot(1)=p(j)
wplot(1)=wisen(j)
2 continue
goto 10

200 l=1-1
500 wmax=500
pmax=10000 
— Maximum Wavespeed and Pressure Values
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
APPENDIX VI. PREPARATION OF GAS DATA

i) Specific Heat at Constant Pressure \( C_p \)

From the known molar compositions (\([\text{Table A1, page 222}]\)), the mass composition of the gas mixtures could be calculated using the Method of Mixtures:-

\[
m_i = \frac{x_i M_i}{\sum_{i=1}^{n} (x_i M_i)}
\]

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:-

\[
\overline{C_p} = \sum_{i=1}^{n} m_i C_{p_i}
\]

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:

<table>
<thead>
<tr>
<th>Molecular %</th>
<th>Component</th>
<th>Molecular Weight (From Table A3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.498</td>
<td>N₂</td>
<td>28.013</td>
</tr>
<tr>
<td>1.073</td>
<td>CO₂</td>
<td>44.010</td>
</tr>
<tr>
<td>83.266</td>
<td>CH₄</td>
<td>16.043</td>
</tr>
<tr>
<td>9.608</td>
<td>C₂H₆</td>
<td>30.070</td>
</tr>
<tr>
<td>3.597</td>
<td>C₃H₈</td>
<td>44.094</td>
</tr>
<tr>
<td>0.3414</td>
<td>iC₄H₁₀</td>
<td>58.124</td>
</tr>
<tr>
<td>0.4581</td>
<td>nC₄H₁₀</td>
<td>58.124</td>
</tr>
<tr>
<td>0.0403</td>
<td>iC₅H₁₂</td>
<td>72.151</td>
</tr>
<tr>
<td>0.0342</td>
<td>nC₅H₁₂</td>
<td>72.151</td>
</tr>
<tr>
<td>0.0046</td>
<td>C₆H₁₄</td>
<td>86.178</td>
</tr>
</tbody>
</table>

The mass percentage of, for example, methane (CH₄) is

\[ m_{\text{CH}_4} = \frac{83.266 \times 16.043}{(1.498 \times 28.013) + (1.073 \times 44.010) + (83.266 \times 16.043) + \ldots} \]

\[ = 69.40\% \]

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:

$$\bar{M} = \frac{\sum_{i=1}^{n} x_i M_i}{\sum_{i=1}^{n} x_i}$$  \hspace{1cm} (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:

$$R = \frac{R^*}{\bar{M}}$$  \hspace{1cm} (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]:

$$\text{Mean Molecular Weight } \bar{M} = \frac{(1.498 \times 28.013) + (1.073 \times 44.010) + (83.266 \times 16.043) + \ldots}{(1.498 + 1.073 + 83.266 + 9.608 + 3.597 + \ldots)}$$

$$= 19.264$$

$\therefore \quad R = \frac{8314}{19.264} \quad \text{J/kg K}$

$$= 432 \text{ J/kg K}.$$
### 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.

![Coefficient Charts for use in the Method of Grieves and Thodos](image)

The following formula was then implemented to obtain values for the critical temperatures:

$$ T_{cm} = \frac{\Sigma_{i=1}^{n} \left( T_{ci} \right)}{1 + \frac{1}{\Sigma_{j=1}^{n} A_{ij} \cdot x_j}} $$

(5)

where $T_{ci} = \text{critical temperature of component i (} \degree \text{R)}$

$T_{cm} = \text{mixture critical temperature (} \degree \text{R)}$

$x_i = \text{molar fraction of component i}$

$x_j = \text{molar fraction of component j}$

$A_{ij} = \text{coefficient taken from chart}$

$n = \text{number of components.}$
EXAMPLE

For Groves' data, neglecting the N₂ and CO₂ components of the natural gas, the main components are:

<table>
<thead>
<tr>
<th></th>
<th>Molar fraction</th>
<th>T_b (°R)</th>
<th>T_c (°R)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Methane</td>
<td>0.83266</td>
<td>200.9</td>
<td>342.7</td>
</tr>
<tr>
<td>2. Ethane</td>
<td>0.09608</td>
<td>332.3</td>
<td>549.7</td>
</tr>
<tr>
<td>3. Propane</td>
<td>0.03597</td>
<td>416.0</td>
<td>665.6</td>
</tr>
</tbody>
</table>

From these three components three binary pairs exist. The ratios of the boiling point temperatures and the corresponding coefficients are listed below:

<table>
<thead>
<tr>
<th></th>
<th>T_bj/T_bi</th>
<th>Coefficients (from Figure A.2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methane-ethane</td>
<td>1.65</td>
<td>A_{12} = 1.46, A_{21} = 0.65</td>
</tr>
<tr>
<td>Methane-propane</td>
<td>2.07</td>
<td>A_{13} = 2.06, A_{31} = 0.42</td>
</tr>
<tr>
<td>Ethane-propane</td>
<td>1.25</td>
<td>A_{23} = 0.966, A_{32} = 0.986</td>
</tr>
</tbody>
</table>

Therefore

\[
T_{cm} = \frac{342.7}{1 + \frac{9.608}{83.266} (1.46) + \frac{3.597}{83.266} (2.06)} + \frac{549.7}{1 + \frac{83.266}{9.608} (0.65) + \frac{3.597}{9.608} (0.966)}
\]

\[
+ \frac{665.6}{1 + \frac{83.266}{3.597} (0.42) + \frac{9.608}{3.597} (0.986)}
\]

\[
= 400.8 °R
\]

\[
= -50.5 °C.
\]
iv) Critical Pressure $P_c$

The critical pressures were calculated from the following equation (Prausnitz and Gunn [1958]):

$$P_{cm} = \frac{R^* (\sum x_i Z_{ci}) T_{cm}}{\sum_i (x_i V_{ci})}$$

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:

<table>
<thead>
<tr>
<th>i</th>
<th>$x_i$</th>
<th>$V_{ci}$ (cm$^3$/mol)</th>
<th>$Z_{ci}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.498</td>
<td>89.8</td>
<td>0.290</td>
</tr>
<tr>
<td>2</td>
<td>1.073</td>
<td>93.9</td>
<td>0.274</td>
</tr>
<tr>
<td>3</td>
<td>83.266</td>
<td>99.2</td>
<td>0.288</td>
</tr>
<tr>
<td>4</td>
<td>9.608</td>
<td>148.3</td>
<td>0.285</td>
</tr>
<tr>
<td>5</td>
<td>3.597</td>
<td>203.0</td>
<td>0.281</td>
</tr>
<tr>
<td>6</td>
<td>0.3414</td>
<td>263</td>
<td>0.283</td>
</tr>
<tr>
<td>7</td>
<td>0.4581</td>
<td>255</td>
<td>0.274</td>
</tr>
<tr>
<td>8</td>
<td>0.0403</td>
<td>303</td>
<td>0.269</td>
</tr>
<tr>
<td>9</td>
<td>0.0342</td>
<td>304</td>
<td>0.263</td>
</tr>
<tr>
<td>10</td>
<td>0.0046</td>
<td>370</td>
<td>0.264</td>
</tr>
</tbody>
</table>
Therefore:

\[ \sum_{i} (x_i z_{ci}) = (1.498 \times 0.290) + (1.073 \times 0.274) + (83.266 \times 0.288) + \ldots = 28.701 \]

and

\[ \sum_{i} (x_i v_{ci}) = (1.498 \times 89.8) + (1.073 \times 93.9) + (83.266 \times 99.2) + \ldots = 10881 \text{ cm}^3/\text{mol} \]

\[ P_{cm} = \frac{8.3144 \times 28.701 \times 222.66 \times 10^3}{10881} \text{ kPa} \]

\[ = 4888 \text{ kPa}. \]
REFERENCES


